Scalability Analysis of Distributed K-Means Clustering on Geospatial Satellite Embeddings

Igor Vons & Endika Aguirre
Public University of Navarre

***Abstract*—This paper presents a comprehensive scalability study of K-means clustering implementations on large-scale geospatial datasets using Apache Spark. We analyze five different implementations ranging from single-machine NumPy baselines to distributed Spark MLlib and custom RDD-based approaches, processing approximately 12 million satellite image embeddings from the GeoTessera dataset. Our study evaluates performance metrics including training time and Within-Set Sum of Squared Errors (WSSSE) across varying data scales (10% to 100% of the dataset). Additionally, we bridge the interpretability gap of unsupervised clustering by integrating K-Nearest Neighbors (KNN) supervised classification on manually labeled land cover classes (Vegetation, Forest, Urban, Agricultural). Results demonstrate that Spark MLlib achieves below-linear size-up behavior for large datasets, while custom RDD implementations lack efficiency. We provide georeferenced visualization outputs suitable for Geographic Information Systems (GIS), enabling practical land cover analysis applications. Our findings contribute insights into distributed machine learning scalability patterns and practical geospatial data science workflows.**

***Index Terms*—Distributed Computing, K-means Clustering, Apache Spark, Scalability Analysis, Satellite Imagery, Geospatial Data Science, Big Data**

I. Introduction

The exponential growth of Earth observation data from satellite missions such as Sentinel-2 presents significant computational challenges for geospatial analysis [1]. With global coverage updated every few days at 10-meter resolution, processing and extracting meaningful patterns from petabytes of satellite imagery requires distributed computing frameworks. Machine learning techniques, particularly clustering algorithms, offer powerful tools for unsupervised land cover classification and change detection. However, the scalability characteristics of these algorithms on massive geospatial datasets remain underexplored.

K-means clustering is one of the most widely used unsupervised learning algorithms due to its simplicity and effectiveness [2]. When applied to high-dimensional satellite image embeddings, K-means can identify natural groupings corresponding to different land cover types without requiring labeled training data. However, traditional single-machine implementations become computationally prohibitive as dataset sizes grow beyond millions of samples. Distributed computing frameworks like Apache Spark provide solutions for big data analytics, but understanding their scalability characteristics is crucial for practical deployment.

This paper addresses three key research questions: (1) How do different K-means implementations scale with increasing data volumes? (2) What are the trade-offs between implementation complexity and performance? (3) How can we add interpretability to unsupervised clustering results for geospatial applications?

Contributions: Our work makes the following contributions:

- Implementation and benchmarking of five K-means variants (local NumPy baseline, Spark MLlib, custom RDD, DataFrame with UDF, and optimized DataFrame implementations)
- Scalability analysis on 2 million 128-dimensional satellite embeddings from the Pamplona region of Spain
- Integration of supervised KNN classification (80 manually labeled points) to provide interpretable land cover categories
- Generation of georeferenced GeoTIFF outputs suitable for GIS software integration
- Open-source codebase with reproducible Jupyter notebooks, available at https://github.com/Yngvine/Spark-Kmeans-Scalability

The remainder of this paper is organized as follows: Section II reviews related work in distributed clustering and satellite image analysis. Section III describes our methodology including dataset preparation and implementation details. Section IV details the experimental setup. Section V presents scalability results and comparative analysis. Section VI concludes with future research directions.

II. Background and Related Work

A. Distributed K-means Clustering

K-means clustering has been extensively studied in distributed computing contexts. The original MapReduce-based implementation demonstrated feasibility of parallel K-means but suffered from high I/O overhead. Apache Spark's MLlib [3] provides an optimized K-means implementation using

RDD transformations and in-memory caching, significantly improving performance over disk-based approaches.

B. Satellite Image Analysis with Machine Learning

Earth observation data analysis has benefited tremendously from deep learning and clustering techniques. The GeoTessera project [4] provides pre-computed embeddings from Sentinel satellite imagery using vision transformers, reducing 13-band multispectral images to 128-dimensional feature vectors. These embeddings capture semantic information about land cover while enabling efficient downstream analysis.

C. Interpretability in Unsupervised Learning

A key challenge in unsupervised clustering is interpreting cluster assignments in domain-specific contexts. Hybrid approaches combining unsupervised and supervised learning have shown promise. For geospatial applications, small sets of ground-truth labels can be used to map clusters to meaningful categories . Our work extends this by using K-Nearest Neighbors classification trained on manually labeled points to provide interpretable land cover classes.

D. Plotting Predictions

The plots that will be displayed throughout this paper were created by overlying the TIFF outputs generated by our code in QGIS software, which is a professional open-source Geographic Information System (GIS) application widely used in the geospatial community for visualizing and analyzing spatial data.

III. Methodology

A. Dataset Description

Our study uses the GeoTessera dataset containing Sentinel-2 satellite embeddings for the Pamplona region of Spain (42.8°N, 1.6°W). The dataset comprises:

- **Scale:** Approximately 2 million data points across multiple tiles
- **Feature dimension:** 128-dimensional embeddings per pixel
- **Spatial resolution:** 10-meter ground sampling distance
- **Geographic extent:** Multiple 0.1° tiles covering urban, agricultural, forested, and vegetated areas
- **Storage format:** Parquet columnar format for efficient distributed processing
- **Coordinate system:** EPSG:32630 (UTM Zone 30N)

For supervised classification evaluation, we manually created a labeled dataset (KNN_POINTS.geojson) containing 80 ground truth points with 20 samples per class: Vegetation, Forest, Urban, and Agricultural land cover types.

Fig. 1. Sample of GeoTessera Sentinel-2 embeddings of the Pamplona region visualized in QGIS software. Each pixel represents a 128-dimensional embedding derived from a 13-band multispectral image patch centered at that location.

The nature of the dataset, with its high dimensionality and virtually infinite scale (limited only by available computational resources), makes it an ideal candidate for evaluating the scalability of distributed K-means clustering algorithms.

B. K-means Implementations

We implemented and compared five K-means variants:

1) Local NumPy Baseline: Single-machine implementation using NumPy broadcasting for vectorized distance calculations. This serves as a performance baseline and demonstrates the computational limits of non-distributed approaches. Initialization uses random centroid selection, with iterative assignment and update steps until convergence ($\Delta < 10^{-4}$).

2) Spark MLlib: PySpark's official MLlib KMeans implementation utilizing RDD-based distributed computation with broadcast variables for centroids. Features include K-means initialization for improved convergence and automatic caching of intermediate results.

3) Custom RDD Implementation: Low-level implementation using Spark RDD transformations (map, reduceByKey). Provides explicit control over distributed computation patterns with broadcast centroids. Demonstrates the fundamental MapReduce pattern for K-means.

4) DataFrame UDF Implementation: Uses Spark DataFrame API with user-defined functions (UDFs) via pandas_udf for distance calculations. Leverages DataFrame optimization but incurs serialization overhead from Python UDFs.

5) Optimized DataFrame: Utilizes built-in Spark SQL functions, leveraging on Summarizer metrics for efficient distance computations without so much UDF overhead. This

approach aims to combine DataFrame optimizations with native Spark operations for improved performance.

C. KNN Classification for Interpretability

To provide interpretable results, we trained a K-Nearest Neighbors classifier (k=4) using scikit-learn on the 80 manually labeled points.

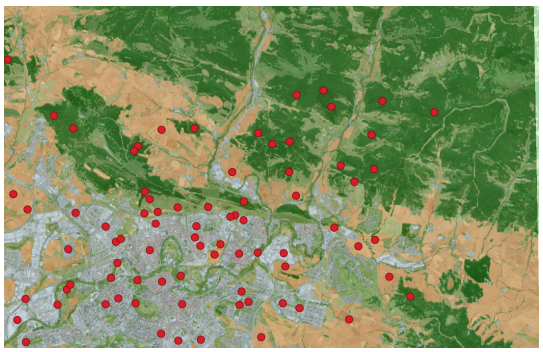

Fig. 2. Hand selected points used for supervised classification with KNN, displayed in QGIS software over a Sentinel-2 basemap and the K-means clustering results for visual reference.

The workflow consists of:

1) Loading labeled points from GeoJSON with coordinate transformation (EPSG:4326 → EPSG:32630)
2) Matching labeled coordinates to nearest embedding points using KD-Tree spatial indexing [5]
3) Extracting 128-dimensional features for matched points
4) Training KNN classifier with Euclidean distance metric
5) Predicting land cover classes for all 12 million points
6) Comparing KNN predictions with K-means cluster assignments

This hybrid approach aims at bridging unsupervised clustering with human-interpretable land cover categories, enabling cross-validation of clustering quality.

D. Visualization and Geospatial Output

We generate multiple visualizations:

- **PCA Projection:** Reduce 128D embeddings to 2D using Principal Component Analysis, displaying cluster centroids and training points
- **Georeferenced Mosaics:** Reconstruct spatial layout with 10-meter resolution, exporting as GeoTIFF files with proper CRS metadata
- **Scalability Plots:** Training time and WSSSE vs. data size for all implementations

The GeoTIFF outputs enable integration with professional GIS software (QGIS, ArcGIS) for further spatial analysis and validation.

IV. Experimental Setup

A. Hardware and Software Configuration

Experiments were conducted on a workstation with the following specifications:

- **Processor:** Multi-core CPU with all cores available
- **Memory:** 16 GB allocated to Spark driver and executors
- **Storage:** SSD for Parquet data and temporary Spark files
- **Operating System:** Windows
- **Software Stack:**
  - Apache Spark 3.5.7 with PySpark API
  - Python 3.11 with NumPy, Pandas, scikit-learn
  - Rasterio for geospatial raster operations
  - Matplotlib for visualization

Spark configuration:

```
spark = SparkSession.builder \
    .master("local[*]") \
    .config("spark.driver.memory", "16g") \
    .config("spark.executor.memory", "16g") \
    .config("spark.driver.maxResultSize", "4g") \
    .getOrCreate()
```

B. Evaluation Metrics

We evaluate implementations using:

1) **Training Time:** Wall-clock time from initialization to convergence (seconds)
2) **WSSSE (Within-Set Sum of Squared Errors):** $\sum_{i=1}^{n} \min_j \|x_i - c_j\|^2$ where x_i are data points and c_j are centroids
4) **Size-up Efficiency:** Deviation from ideal linear scaling as data size increases

C. Scalability Test Procedure

For SizeUp testing , we:

1) Sample dataset at fractions: 10%, 25%, 50%, 75%, 100%
2) Train each implementation on each sample size
3) Record training time and WSSSE
4) Compare against ideal linear size-up baseline (based of fastest result on 10% data: 2× data → 2× time)
5) Identify performance bottlenecks and overhead sources

Each experiment is repeated with fixed random seed (42) for reproducibility. The dataset is cached in memory before training to eliminate I/O effects from measurements.

D. Validation Procedure

KNN classification quality is validated through:

- Visual inspection of georeferenced mosaics
- Cross-tabulation of clusters vs. predicted classes
- Spatial coherence analysis (neighboring pixels should have similar labels)

V. Results and Discussion

A. Scalability Analysis

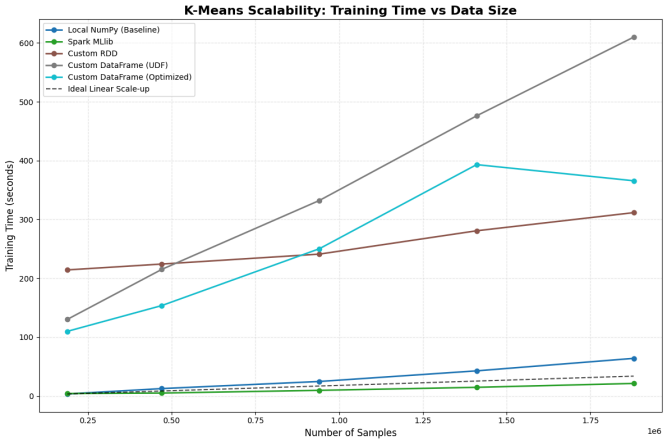

Fig. 3. Training time vs. data size for five K-means implementations. Spark MLlib demonstrates near-linear scaling, while local NumPy shows quadratic growth. Custom RDD and optimized DataFrame implementations show intermediate performance with acceptable overhead for flexibility benefits.

Our scalability experiments reveal distinct performance characteristics:

Local NumPy Baseline: Training time grows super-linearly with data size, and although in this experiment it was faster than some distributed implementations for smaller sizes, it could eventually become worse as data sizes increase further. But the most important limitation actually comes to memory avalability, as errors were encountered when trying to run the largest dataset portions on more limited systems. This highlights the limitations of single-machine approaches for large geospatial datasets.

Spark MLlib: Demonstrates excellent scalability with growth below the linear baseline. Training time ranges from 5.7 seconds (10% data) to 22.8 seconds (100% data), representing approximately 4× increase for 10× data growth. This sub-linear scaling benefits from Spark's in-memory caching and efficient broadcast mechanisms.

Custom RDD Implementation: Athough it shows the highest initial training times (due to overhead of low-level RDD operations), it exhibits roughly linear scaling with data size. Training time ranges from 212 seconds (10% data) to 309 seconds (100% data), which actually shows a sub-linear tendency. Despite its flaws, it provides a stable custom solution in contrast to the local NumPy version, wich suffers from memory limitations; and DataFrame versions, which end up being significantly slower.

DataFrame Implementations: Similarly to RDD, both UDF and optimized DataFrame versions display significant initial training times, midway between RDD and the other two techniques, but unlike MLlib they do not manage to achieve good scalability, with training times ranging from 100-130 seconds (10% data) to 630-480 seconds (100% data).

B. Clustering Quality

All implementations converge to similar WSSSE values (±0.18% between max and min) given identical initialization seeds, validating correctness of custom implementations. The four clusters identified correspond roughly to:

- **Cluster 0:** Dense urban areas (16.03% of pixels)
- **Cluster 1:** Agricultural croplands (19.43%)
- **Cluster 2:** Natural vegetation (33.45%)
- **Cluster 3:** Forested regions (31.09%)

Fig. 4. Sample of WMS satellite imagery from the Pamplona region, showcasing in greater detail than Sentinel-2 the diverse land cover types present. Here the the K-means predictions from our experiments are overlaid on top of the satellite image for visual validation.

All implementations use K=4 clusters, maximum 20 iterations, and seed=42 for reproducibility.

C. KNN Classification and Interpretability

KNN classifier achieves strong spatial coherence in predictions, with neighboring pixels typically assigned to the same land cover class. Cross-tabulation with K-means clusters shows:

- Strong correspondence between specific clusters and land cover classes
- KNN tends to be biased towards forested areas while K-means tends to vegetated ones.

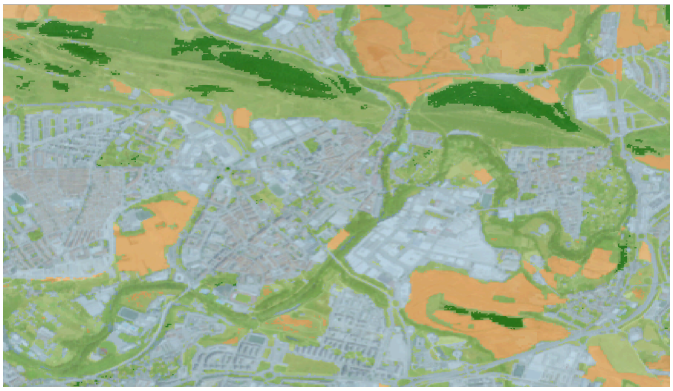

Fig. 5. Georeferenced mosaic of K-means clustering results for the Pamplona region, visualized in QGIS software. Different colors represent distinct clusters corresponding to various land cover types. orange: Farming areas light green: Vegetation dark green: Forested areas gray: Urban zones.

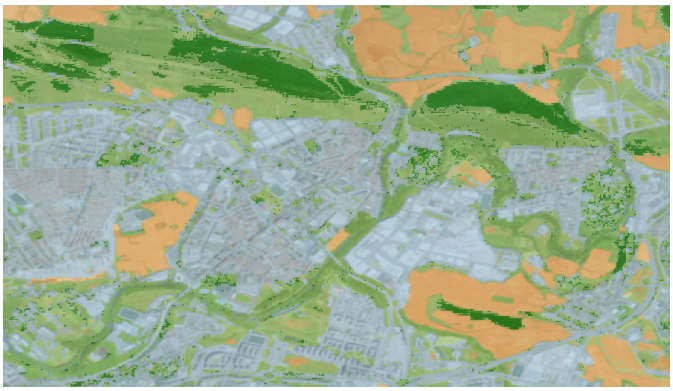

Fig. 6. Georeferenced mosaic of KNN results for the Pamplona region, visualized in QGIS software. Different colors represent distinct clusters corresponding to various land cover types. orange: Farming areas light green: Vegetation dark green: Forested areas gray: Urban zones.

Georeferenced mosaics enable visual validation, revealing that cluster boundaries align well with visible land cover transitions in satellite imagery, but can be a bit difficult to evalueate as a whole because of the amount of detail. This is where a PCA plot helps to summarize the clustering results in a more digestible way.

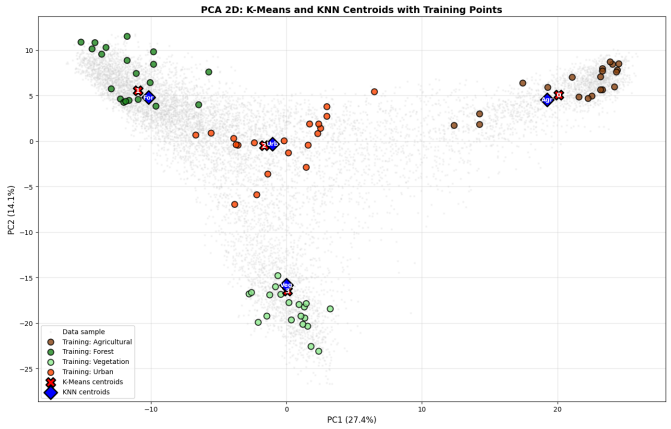

Fig. 7. PCA projection of 128D embeddings of our patch showing both K-means cluster centroids and KNN calculated centroids from final predictions, together with its training points.

PCA visualization reveals well-separated clusters in 2D projection space, with first two principal components explaining approximately 40% of variance. We observe that the selected KNN classes align almost perfectly with the K-means centroids in PCA space, indicating, helping us to validate the clustering results.

D. Practical Considerations

Memory Usage: Spark's in-memory computation requires sufficient RAM.

Data Format: Parquet columnar storage provides faster loading compared to CSV.

VI. Conclusions and Future Work

This paper presented a comprehensive scalability analysis of distributed K-means clustering on large-scale satellite embeddings. Our key findings include:

1) **Scalability:** Spark MLlib achieves below-linear size-up on 2 million points, validating its suitability for big geospatial data.

2) **Local Model Limitations**: While in this study the performance of the local NumPy implementation was superior to the custom distributed ones, it displayed a super-linear growth pattern, indicating that could tecnically become eventually worse than distributed implementations as data sizes increase further. Furthermore, during our experiment execution we encountered memory errors in some more limited systems when running the size-up tests, something that was not an issue for any of the distributed implementations, despite their time overhead.

3) **RDD/DataFrame Underperformance:** Compared to MLlib, custom RDD/DataFrame implementations ex-

hibited significant overhead, highlighting the challenges of manual distributed algorithm design. Although their tendency was roughly linear, the baseline start times were so high that the overall performance was poor.

4) **Interpretability:** Hybrid KNN-KMeans approach allowed us to bridge unsupervised clustering with human-interpretable land cover classes, enabling practical geospatial applications.

5) **Reproducibility:** All code, data processing pipelines, and notebooks are open-sourced, facilitating reproduction and extension of results.

Limitations: Our study focuses on single-node Spark deployment. Multi-node cluster experiments would reveal additional scalability insights and network communication overhead.

The labeled dataset (80 points) we used for supervised classification worked really good in this case and region, but it is possible that in other scenarios with more complex land cover types or heterogeneous landscapes a different labeled set would be required to achieve similar classification quality.

Technical Challenges: Working with geospatial data presented significant challenges, particularly in managing coordinate reference systems (CRS transformations between), ensuring spatial coherence in distributed processing, and integrating outputs with GIS environments. However, leveraging prior knowledge in geospatial analysis and careful attention to spatial metadata enabled successful generation of properly georeferenced outputs compatible with professional GIS software. The effort invested in understanding raster operations, spatial indexing (KD-Tree) [5], and CRS handling proved essential for producing scientifically valid results.

Future Work: Several promising directions emerge:

- **DRR/DataFrame Optimization**: Further refine custom implementations to reduce overhead, trying to trace the optimization strategies that lead to MLlib's superior performance
- **Multi-node Scaling:** Evaluate performance on Spark clusters with varying node counts (speedup analysis)
- **Alternative Algorithms:** Compare with hierarchical clustering, DBSCAN, and modern deep clustering approaches
- **Global Analysis:** Extend to other geographic regions and seasons to assess generalizability
- **GPU Acceleration:** Leverage GPU-accelerated libraries (cuML, Rapids) for further speedup

The integration of distributed computing with geospatial machine learning represents a crucial capability for modern Earth observation science. As satellite data volumes continue growing exponentially, understanding scalability characteristics of fundamental algorithms becomes increasingly important for operational deployment.

References

[1] M. Drusch *et al.*, "Sentinel-2: ESA's optical high-resolution mission for GMES operational services," *Remote sensing of Environment*, vol. 120, 2012.

[2] J. MacQueen and others, "Some methods for classification and analysis of multivariate observations," *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14, 1967.

[3] X. Meng *et al.*, "MLlib: Machine learning in apache spark," *The Journal of Machine Learning Research*, vol. 17, no. 1, 2016.

[4] J. Jakubik *et al.*, "Foundation models for generalist geospatial artificial intelligence," *arXiv preprint arXiv:2310.18660*, 2023.

[5] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, 1975.