Utilizing the Clay Foundation Model and Sentinel-2 Imagery for Urban Growth Monitoring in Johnston County, North Carolina

Benton Tripp

Abstract

This study addresses the challenge of infrequent updates in urban imperviousness data by integrating Sentinel-2 multispectral satellite imagery with the Clay Foundation Model, an open-source deep learning framework for Earth observation. Focusing on Johnston County, North Carolina—a region experiencing rapid urban growth—the research aims to estimate urban density more frequently than the National Land Cover Database (NLCD) updates, which occur approximately every five years. Sentinel-2 imagery, accessed via the Microsoft Planetary Computer and AWS Earth Search APIs, provides high-resolution, multitemporal data suitable for regular monitoring. The Clay Foundation Model generates spatial embeddings from the imagery, capturing detailed spectral information without the need for additional feature engineering or external indices.

To predict urban imperviousness percentages as a proxy for urbanization, various deep learning methodologies were applied, including baseline models, simple neural networks (SNN), and deep neural networks (DNN). Hyperparameter tuning and regularization techniques were employed to optimize model performance and prevent overfitting. This proof-of-concept demonstrates a scalable, data-efficient framework for monitoring urban growth, offering insights into methodological advancements and highlighting the potential of foundation models to enhance sustainable urban planning by filling gaps left by traditional datasets.

Introduction

Literature Review

Urban growth fundamentally reshapes landscapes, ecosystems, and socioeconomic structures, making it a crucial area of study within environmental and urban planning. The expansion of impervious surfaces, such as roads and buildings, serves as a key indicator of urbanization and impacts ecosystems by increasing surface runoff, reducing groundwater recharge, and altering local climates. Understanding and tracking these changes allows researchers and policymakers to manage the environmental consequences of urban growth and devise strategies to minimize negative impacts on biodiversity, water cycles, and air quality (Goetzke et al., 2008).

Remote sensing has emerged as an essential technology for monitoring urban growth, offering extensive and consistent data across time and space. Multitemporal satellite imagery, particularly from sources like Sentinel-2, enables precise tracking of land cover changes over large geographic areas and prolonged periods, making it ideal for detecting patterns of urban expansion (Ayush et al., 2021; Zhu et al., 2017). This technology facilitates not only the visualization of urban sprawl but also the quantitative analysis of changes in land use and land cover. By providing high-resolution, time-sequenced imagery, remote sensing allows for dynamic monitoring of urbanization processes, yielding insights critical for sustainable urban planning and resource management.

Advancements in deep learning have significantly enhanced the capabilities of remote sensing for urban analysis. Traditional neural network architectures, such as simple neural networks (SNNs) and deep neural networks (DNNs), have been effectively employed to model complex relationships in satellite imagery data. These models are capable of learning hierarchical feature representations, making them suitable for tasks like land cover classification and regression-based predictions of urban imperviousness (Dionelis et al., 2024; Zhu et al., 2017). However, training effective neural networks requires careful hyperparameter tuning and regularization to prevent overfitting, especially when working with high-dimensional data and limited labeled samples.

Hyperparameter tuning involves adjusting parameters such as learning rates, the number of hidden layers, the number of neurons in each layer, activation functions, and regularization techniques to optimize model performance. Regularization methods, including dropout and weight decay, help mitigate overfitting by preventing the model from becoming too specialized to the training data (He et al., 2022). Effective hyperparameter optimization can significantly improve the generalization capabilities of neural networks, making them more robust in predicting urban imperviousness from satellite imagery.

Foundation models have recently emerged as a transformative approach in artificial intelligence, particularly in fields like natural language processing with models such as BERT and GPT-3. These models are pretrained on large-scale datasets and can be fine-tuned for specific downstream tasks with minimal additional training. In Earth Observation (EO) and geospatial AI, foundation models demonstrate significant potential for applications such as land cover classification and change detection (Mai et al., 2022; Dionelis et al., 2024). By capturing universal patterns within massive datasets, foundation models enable efficient transfer learning, reducing the reliance on large labeled datasets for each specific task.

The Clay Foundation Model represents an open-source initiative to bring the advantages of foundation models to the domain of Earth observation. By pretraining on extensive satellite imagery datasets, it learns generalized representations that can be adapted to various geospatial tasks with minimal fine-tuning (Clay Foundation, 2023). This approach offers a label-efficient and scalable solution for remote sensing applications, particularly in monitoring urban growth where timely and detailed data are essential.

In this context, the integration of Sentinel-2 imagery with foundation models like the Clay Foundation Model provides a powerful framework for urban growth monitoring. The high-resolution, multitemporal data from Sentinel-2 combined with the advanced representation learning capabilities of foundation models allows for accurate and efficient estimation of urban imperviousness. This methodology addresses the challenges of data scarcity and the need for frequent updates, offering a significant improvement over traditional methods reliant on less frequent datasets like the NLCD.

By leveraging these advancements, this study aims to develop a scalable, data-efficient framework for monitoring urban growth in rapidly developing regions. The focus is on utilizing the strengths of foundation models and the optimization of simple and deep neural networks through hyperparameter tuning and regularization techniques. This approach enhances the analysis of satellite imagery, contributing to sustainable urban planning and resource management efforts.

Motivation for the Research

Despite advancements in deep learning and the introduction of foundation models, limited research has focused specifically on using these models in conjunction with optimized neural network architectures to map urban imperviousness. A significant challenge in monitoring urban growth is the infrequent updates of authoritative land cover datasets. For instance, the National Land Cover Database (NLCD) updates its urban imperviousness data approximately every five years, with recent updates in 2016 and 2021 (U.S. Geological Survey, 2021). In rapidly growing areas like Johnston County, this temporal resolution may not be sufficient to capture dynamic changes in urban density. Many locations experiencing rapid growth have a pressing need for more frequent assessments to inform timely urban planning and environmental management decisions.

Utilizing satellite imagery with frequent revisit times, such as the Sentinel-2 five-day cycle, presents an opportunity to approximate urban density more regularly. By employing the Clay Foundation Model to extract robust embeddings from Sentinel-2 imagery and applying optimized simple and deep neural networks, it is possible to predict urban imperviousness with higher temporal resolution. Hyperparameter tuning and regularization techniques play a crucial role in enhancing model performance and generalization, especially when dealing with complex, high-dimensional data. This

approach can fill the gap left by infrequent updates of traditional land cover datasets, providing a scalable and accurate means to monitor urban growth.

Research Question and Objectives

The primary objective of this study was to evaluate the potential of foundation models in enhancing the temporal resolution of urban imperviousness mapping through the application of optimized neural network architectures. Specifically, the research aimed to leverage the Clay Foundation Model, in conjunction with Sentinel-2 imagery, to predict urban imperviousness in Johnston County, North Carolina, with greater frequency than traditional datasets allow. By using the Clay model's pretrained embeddings extracted from Sentinel-2 images and applying hyperparameter-tuned simple and deep neural networks, the study sought to develop and assess different approaches for predicting urban imperviousness at high spatial and temporal resolutions, without incorporating additional feature engineering or external indices.

This proof-of-concept study intended to demonstrate the value of combining foundation models with optimized neural network architectures in filling the gap left by infrequent land cover data updates. By focusing on hyperparameter tuning and regularization techniques to improve model performance and prevent overfitting, the research contributes to the growing body of work employing foundation models for geospatial applications and highlights their potential for improving sustainable urban planning and resource management in rapidly developing regions (Clay Foundation, 2023; Goetzke et al., 2008).

Study Site

Johnston County is located in the eastern part of North Carolina, United States, covering approximately 2,050 square kilometers. It lies between latitudes 35.3°N and 35.8°N, and longitudes 78.0°W and 78.6°W. The county is part of the rapidly expanding Raleigh-Durham-Chapel Hill metropolitan area, making it a pertinent case study for urban growth analysis.

For spatial analysis, all coordinates were defined in the Universal Transverse Mercator (UTM) coordinate system, specifically UTM Zone 17N (EPSG:32617), with units in meters. This coordinate system provided spatial accuracy and consistency in measurements across the study area and was compatible with the coordinate systems used by datasets such as the Sentinel-2 imagery.

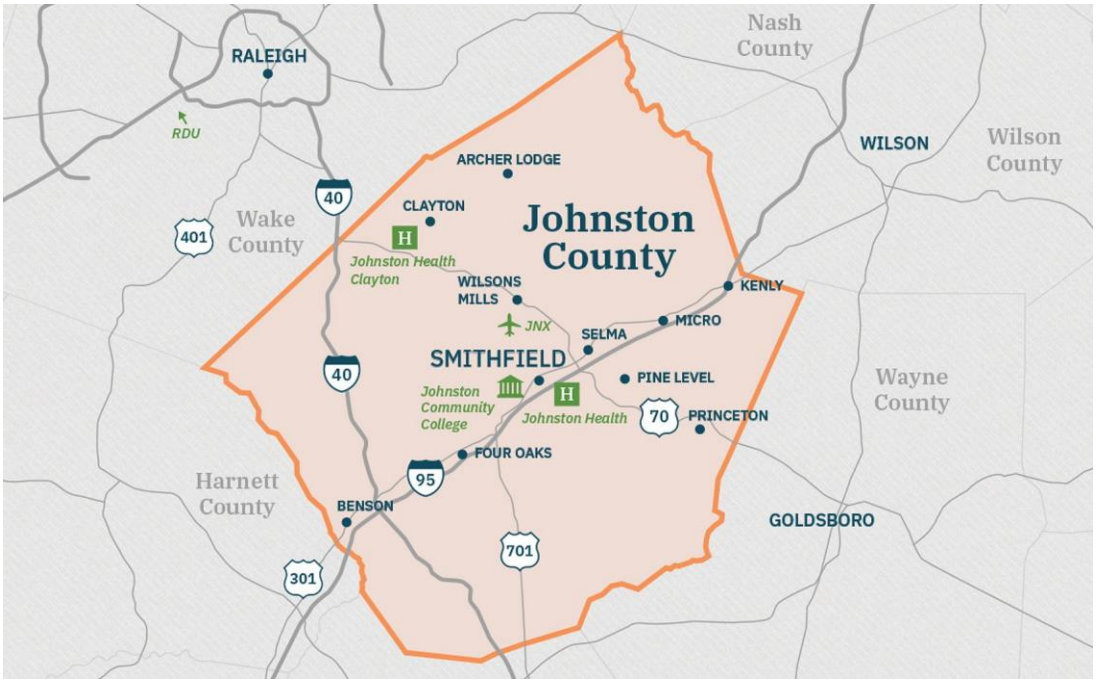

Figure 1: Johnston County, North Carolina (obtained from Johnston County Economic Development)

Data Overview

Several datasets were utilized to define spatial boundaries, analyze land cover, and estimate urban density within Johnston County, North Carolina. The Johnston County GIS data provided the official county boundary, defining the study's spatial extent (Johnston County Department of GIS, 2011). Sentinel-2 satellite imagery from the European Space Agency (ESA) was employed as the exclusive source of multispectral data. Specifically, the MSI Level-1C Top of Atmosphere (TOA) Reflectance Product, Collection 1, offering 10-meter spatial resolution and including 13 spectral bands suitable for land cover analysis, was used. Sentinel-2's revisit interval of approximately five days allowed for frequent monitoring over time (Copernicus Sentinel-2, 2021). Data access was facilitated through the `pystac_client` python library, drawing from the Microsoft Planetary Computer STAC API (Microsoft Open Source et al., 2022) and the Earth Search AWS STAC API, adhering to the SpatioTemporal Asset Catalog (STAC) API specification (STAC Specification, 2023). Urban imperviousness raster data from the National Land Cover Database (NLCD) provided 30-meter resolution raster data on land cover and impervious surfaces, which were used to calculate urban density percentages within spatial patches (U.S. Geological Survey, 2021).

Table 1 summarizes the datasets used in the study along with their key attributes, including coordinate reference systems (CRS), spatial extents, resolutions, formats, and any transformations applied during preprocessing.

Table 1: Summary of datasets and their attributes used in the study.

Dataset	**Sentinel-2 Satellite Imagery**	**Urban Imperviousness**	**Johnston County Boundary**
Original CRS	EPSG:32617 (UTM Zone 17N)	Albers Conical Equal Area (EPSG:4326)	EPSG:32617 (UTM Zone 17N)
Original Spatial Extent	Queried individually within Johnston County Region	minx: -2493045.0 miny: 177285.0 maxx: 2342655.0 maxy: 3310005.0	minx: 707750.13 miny: 3904356.29 maxx: 765959.87 maxy: 3967194.75
Original Resolution / Format	10x10 meters / Multispectral GeoTIFF	30x30 meters / GeoTIFF	Vector
Additional Data Attributes	Bands: ○ B02 (Blue) ○ B03 (Green) ○ B04 (Red) ○ B08 (NIR) Cloud Cover: < 1% Dates: ○ Approximately triannual spread (where available) ○ 2016-2024 ○ 20 dates total	-	-
Transformation Details	Used as primary spatial reference; no CRS transformation needed. Cropped to the 600x600-meter tiles covering Johnston County boundary.	Reprojected to EPSG:32617 and cropped to Johnston County. Resampled to 200m resolution using bilinear interpolation.	Converted to EPSG:32617 to match Sentinel-2 and NLCD data. Used to define study area bounds, and divided into 600x600-meter tiles for consistent spatial units.

Methods

The data processing workflow began by defining the spatial extent of Johnston County, North Carolina, using official boundary data from the Johnston County GIS. This boundary data were reprojected to UTM Zone 17N (EPSG:32617) to align with the coordinate reference system of the Sentinel-2 imagery. A grid of 600×600-meter tiles was generated to cover the county boundary, providing consistent spatial units for analysis. Each tile was stored as a polygon within a geospatial dataset to facilitate tracking and associating data files with each specific area of interest. These tiles served as the fundamental units for subsequent data extraction, processing, and urban density analysis across the study area.

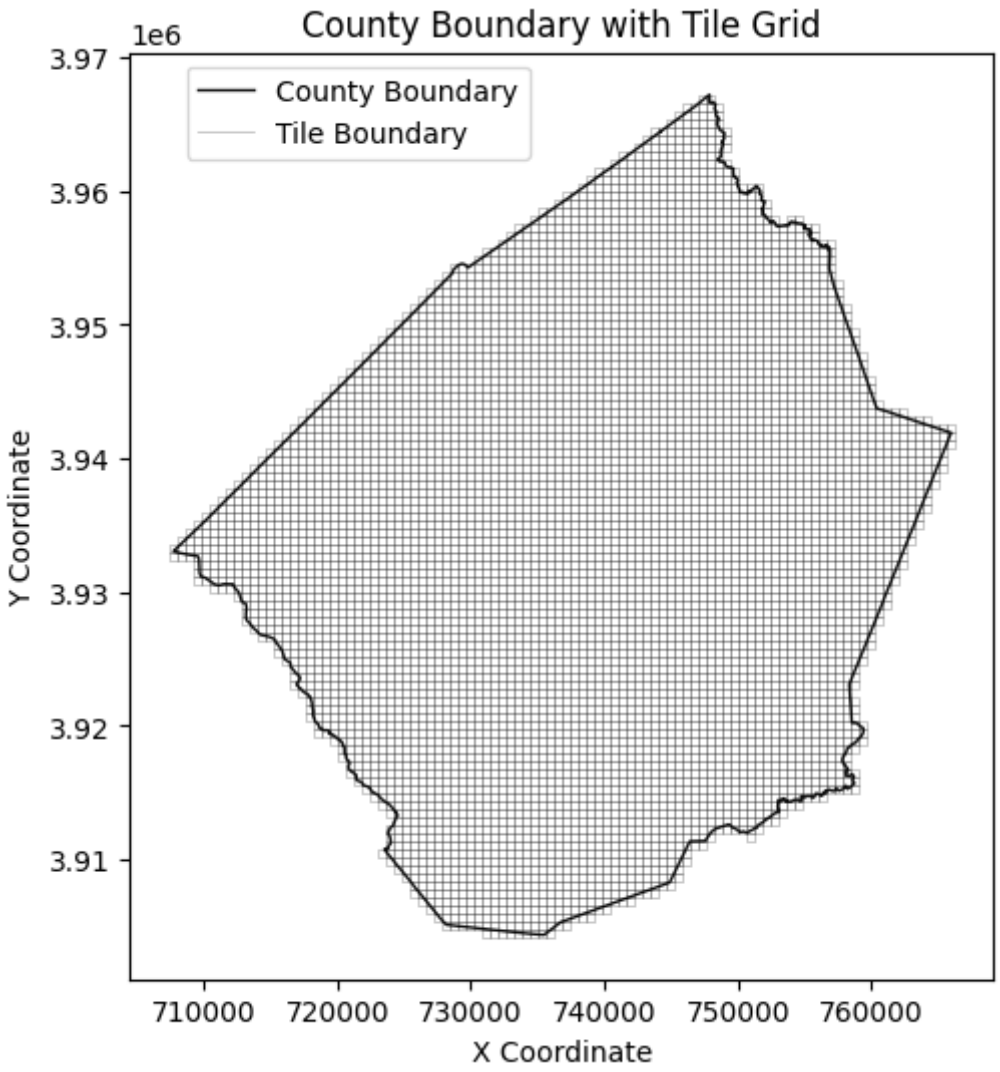

Figure 2: Map displaying the boundary of Johnston County, North Carolina, overlaid with a 600x600-meter tile grid.

Urban imperviousness data from the National Land Cover Database (NLCD) were prepared to align with the spatial framework defined by the 600×600-meter tiles across the study area. Initially, the data were clipped to the study area and reprojected to EPSG:32617 for consistency with other datasets. To maintain a balance between data granularity and computational efficiency, the urban imperviousness data was then resampled from a 30-meter to a 200-meter resolution using bilinear interpolation. This resampling method provided smooth transitions between pixel values, preserving key spatial patterns while adapting the data to a coarser resolution suitable for modeling.

Selecting a 200-meter resolution was specifically determined based on the modeling objectives. Aggregating the data to the full 600×600-meter extent of each tile would result in the loss of fine-scale details essential for accurately capturing urban density variations. Conversely, retaining the original

finer resolution would require significantly more computational resources. By selecting 200 meters, the resampling yielded a manageable 3×3 matrix for each tile, which was used as the target for the deep learning model. This level of precision struck an optimal balance, providing sufficient spatial detail for robust urban growth analysis while remaining computationally feasible for large-scale processing.

Urban imperviousness data were extracted for each tile and associated with the corresponding grid tile, enabling localized assessments of urban density within the study area. This approach facilitated the integration of urban imperviousness data with the Sentinel-2 multispectral imagery, creating a structured dataset for urban growth analysis across Johnston County.

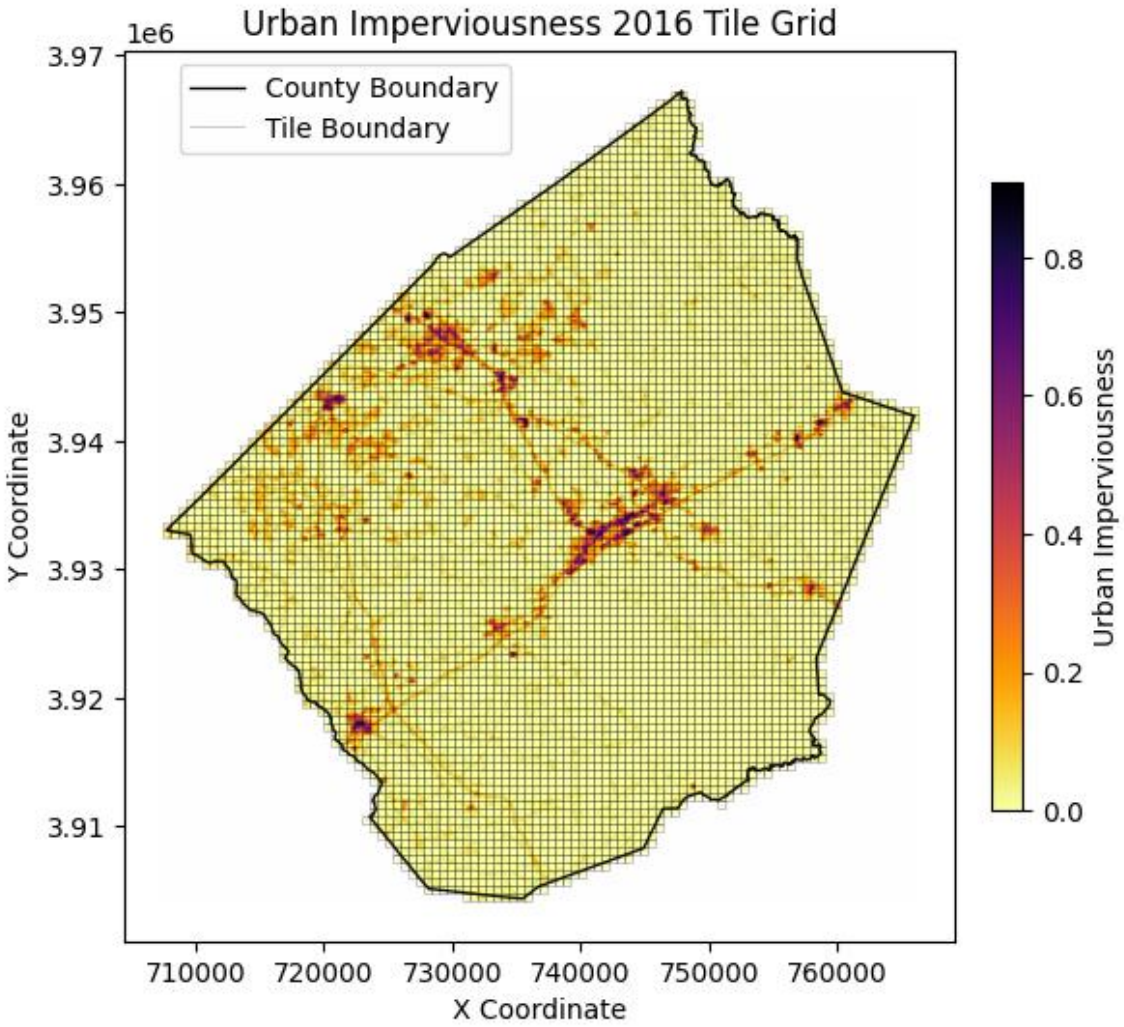

Figure 3: Urban imperviousness across Johnston County, North Carolina, in 2016, with a 600x600-meter tile grid overlay and the county boundary.

Sentinel-2 imagery was retrieved from the Microsoft Planetary Computer and AWS Earth Search APIs, providing multispectral data for the study area. Each image captured detailed spectral information relevant to urban growth analysis. Specifically, four spectral bands—Blue (B02), Green (B03), Red (B04), and Near Infrared (NIR, B08)—were downloaded at a 10-meter spatial resolution to maximize detail. Each raster was organized into 600×600-meter tiles in the EPSG:32617 coordinate reference system, facilitating manageable processing units for spatial analysis. Images were chosen based on quality criteria, primarily focusing on scenes with less than 1% cloud cover to ensure minimal interference in the analysis. The goal was to collect a consistent set of images over time, representing different seasonal conditions that could influence vegetation and other land features. While an ideal quarterly sampling interval was targeted from 2016 to 2024, certain images were unavailable due to various constraints, such as temporary data gaps or weather-related omissions. This resulted in an

adjusted schedule with images retrieved approximately three times per year, yielding a total of 20 images covering the study period.

To optimize the temporal distribution of the selected dates, a filtering and date-selection process was implemented. This process involved buffering around unavailable dates to avoid excessive temporal clustering and ensured that the selected images represented the best possible seasonal coverage for each year. Additionally, adjustments were made to avoid redundant or less informative dates, yielding a final dataset with a balanced temporal spread across the study period. Each 600×600-meter tile, aligned to EPSG:32617, was thus associated with urban imperviousness data and multitemporal spectral imagery, creating a robust, high-resolution dataset for analyzing urban growth. This selection process enabled the creation of a multispectral dataset that accounted for seasonal variability and minimized data redundancy, supporting the extraction of spatial features necessary for the deep learning model's urban growth analysis.

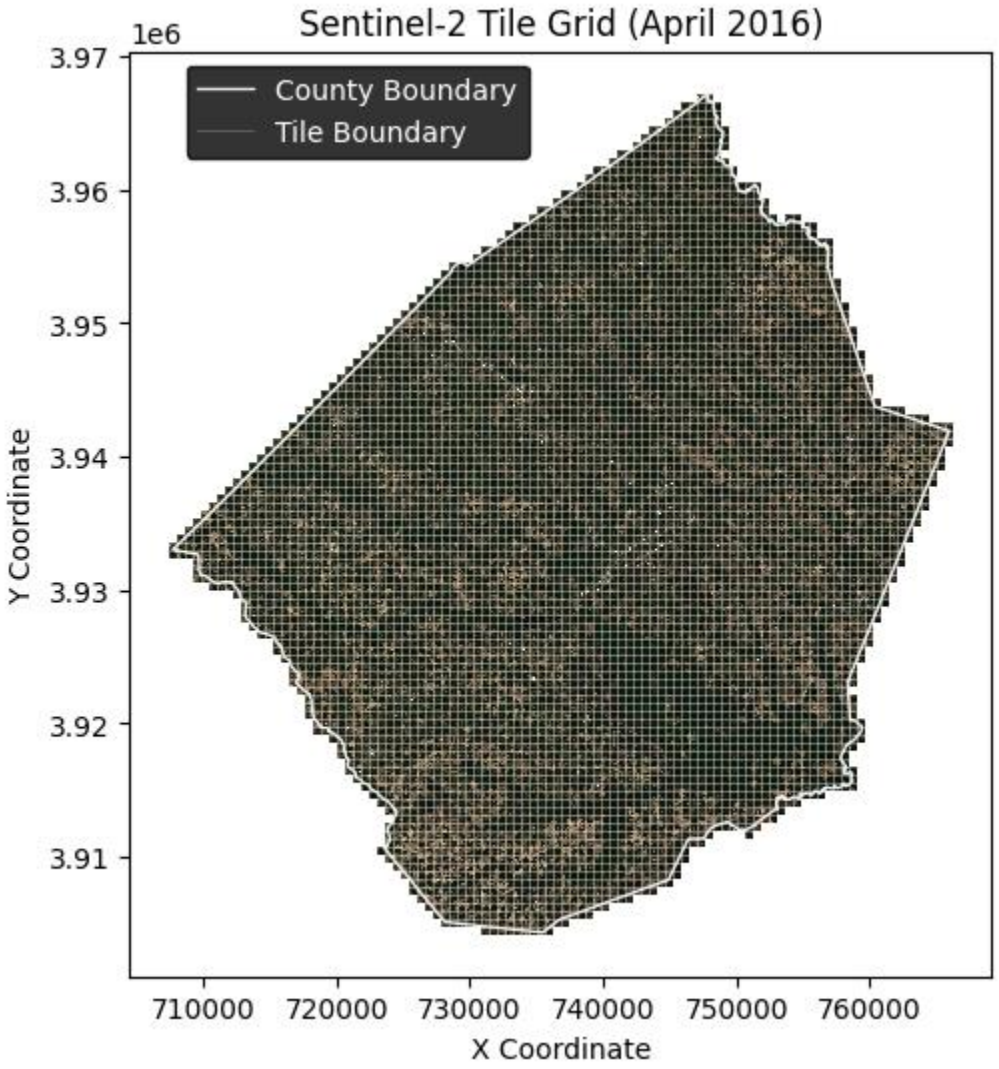

Figure 4: Sentinel-2 imagery of Johnston County in April 2016, overlaid with a 600x600-meter tile grid used and the county boundary.

To leverage the multispectral Sentinel-2 dataset for urban growth analysis, the Clay Foundation Model was employed—a pretrained deep learning framework specifically designed for Earth observation applications (Clay Foundation, 2023). This model was utilized to extract spatial embeddings from the Sentinel-2 imagery, effectively capturing complex spectral features relevant to urban imperviousness. Prior to embedding extraction, the Sentinel-2 images were preprocessed to align with the model's input

requirements, including normalization of pixel values and the integration of temporal and spatial context.

Each Sentinel-2 image was normalized by applying band-specific means and standard deviations, ensuring consistent input across the dataset. To incorporate temporal information, temporal embeddings were generated for each image date using sine and cosine transformations of the date values, capturing seasonal patterns that might influence spectral signatures. Spatial context was integrated by calculating normalized latitude and longitude embeddings based on the centroid coordinates of each 600×600-meter tile, allowing the model to account for geographic variations within the study area.

Using the preprocessed data, data cubes were constructed for each tile and image date by combining the spectral bands with the temporal and spatial embeddings. The Clay Foundation Model was then applied to these data cubes to extract spatial embeddings for each tile, resulting in a rich set of features that encapsulated both spectral and contextual information pertinent to urban imperviousness. These embeddings were stored alongside the associated dates and tile identifiers, forming a comprehensive dataset for subsequent modeling.

Concurrently, the urban imperviousness data was processed to serve as the target variable for model training. The resampled 200-meter resolution imperviousness data for each tile was normalized by dividing by 100 to represent percentage values between 0 and 1. This resulted in a 3×3 matrix of imperviousness values for each tile, providing a spatially detailed target for the model. By associating the extracted embeddings with the corresponding imperviousness matrices, a dataset suitable for supervised learning was established, where the model aimed to predict urban imperviousness based on the spectral and contextual features captured in the embeddings.

The final dataset was organized into a geospatial data structure, aligning the embeddings, temporal and spatial features, and imperviousness targets for each tile and date. This structured approach facilitated efficient data handling and allowed for the implementation of deep learning models to analyze urban growth patterns across Johnston County. The dataset was divided into training and testing subsets to evaluate the model's performance, ensuring robust assessment of its predictive capabilities.

To evaluate predictive performance, the dataset was partitioned into training, validation, and testing subsets using a combination of temporal and spatial clustering techniques to ensure representative sampling and mitigate spatial autocorrelation. Initially, the data were filtered to include only imagery and imperviousness data from before 2017 for model training and validation, reserving post-2017 data exclusively for future predictions and assessment. The centroids of each tile within the training and validation subset were extracted, and KMeans clustering was applied to group the spatial tiles into 30 clusters based on their geographic coordinates. This spatial grouping ensured that geographically proximate tiles were assigned to the same cluster, reducing the risk of spatial leakage between the splits. The clusters were then randomly shuffled, and a 70:10:20 ratio was applied to allocate clusters to the training, validation, and testing sets, respectively. This process maintained the spatial integrity of the data within each split while ensuring a representative distribution across the study area. The resulting splits were visualized to verify the geographic and proportional balance of the training, validation, and testing subsets. The features and targets for each subset were subsequently prepared as PyTorch tensors.

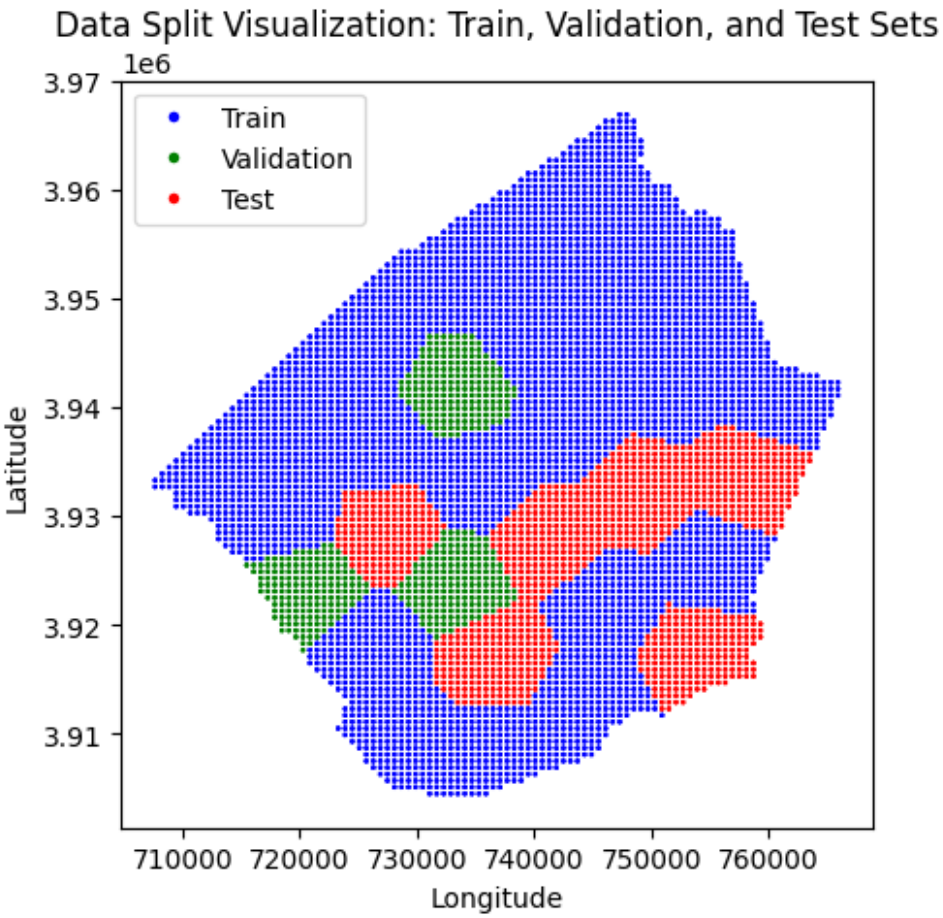

Figure 5 - Visualization of data splits for training, validation, and testing sets across Johnston County. Each tile represents a 600×600-meter area, with colors indicating the assigned data split: blue for training, green for validation, and red for testing. The clustering ensures spatial consistency and minimizes overlap between splits.

The feature set comprised the flattened spatial embeddings extracted from the Clay Foundation Model, which encapsulated the spectral and contextual information for each tile and date. The target variable was the corresponding 3×3 matrix of urban imperviousness values for each tile, reshaped into a one-dimensional array to align with the model's output requirements. Both features and targets were converted into numerical arrays and then into PyTorch tensors to facilitate efficient computation during model training. Data preprocessing steps, including normalization and reshaping, were applied consistently across all subsets to maintain compatibility with the deep learning framework and to optimize the learning process.

To establish a benchmark for predictive performance, a baseline model was developed using the mean urban imperviousness calculated across the training dataset. This baseline prediction served as a reference point against which the performance of more complex models could be compared. Evaluating the neural networks relative to this baseline allowed assessment of the extent to which they captured the spatial variability and intricate patterns of urban imperviousness beyond the average values.

Two distinct neural network architectures were implemented to predict urban imperviousness from the extracted spatial embeddings: a simple fully connected neural network (SNN) and a deep fully connected neural network (DNN). Each model was designed to explore different aspects of the data and assess how effectively the embeddings captured relevant information for predicting urban imperviousness.

Simple Neural Network (SNN)

The simple neural network served as a baseline neural model to assess the predictive capacity of the embeddings with a minimal network structure. It consisted of an input layer, a single hidden layer with nonlinear activation functions, and an output layer producing the predicted imperviousness values. This model was chosen for its simplicity and to establish whether the embeddings contained sufficient information for prediction without complex transformations.

Mathematically, the SNN can be represented as:

$$\mathbf{h} = \text{ReLU}(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$$

$$\hat{\mathbf{y}} = \sigma(\mathbf{W}_2\mathbf{h} + \mathbf{b}_2)$$

where:

- $\mathbf{x} \in \mathbb{R}^{768}$ is the input feature vector (flattened embeddings).
- $\mathbf{W}_1 \in \mathbb{R}^{128\times768}$ and $\mathbf{W}_2 \in \mathbb{R}^{9\times128}$ are weight matrices.
- $\mathbf{b}_1 \in \mathbb{R}^{128}$ and $\mathbf{b}_2 \in \mathbb{R}^9$ are bias vectors.
- ReLU is the Rectified Linear Unit activation function:

$$\text{ReLU}(\mathbf{x}) = \max(0, \mathbf{x})$$

- σ is the sigmoid activation function ensuring outputs between 0 and 1:

$$\sigma(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{x}}}$$

- $\hat{\mathbf{y}} \in \mathbb{R}^9$ is the predicted imperviousness vector.

Deep Neural Network (DNN)

The deep neural network extended the architecture by adding an additional hidden layer between the input and output layers. This increased depth allowed the model to learn more complex representations and capture nonlinear relationships within the data. By incorporating multiple layers, the DNN could hierarchically extract features from the embeddings, potentially improving predictive performance over the simpler architecture.

The DNN is mathematically expressed as:

$$\mathbf{h}_1 = \text{ReLU}(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$$

$$\mathbf{h}_2 = \text{ReLU}(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2)$$

$$\hat{\mathbf{y}} = \sigma(\mathbf{W}_3\mathbf{h}_2 + \mathbf{b}_3)$$

where the additional hidden layer $\mathbf{h}_2$ allows for deeper feature extraction.

Table 2: Summary of Neural Network Models.

Model	Architecture	Activation Function(s)	Parameters
Baseline	Mean of training data	N/A	N/A
Simple NN	Input (768) → Hidden Layer (128) → Output (9)	ReLU, Sigmoid	99,593
Deep NN	Input (768) → Hidden Layers (128, 128) → Output (9)	ReLU, Sigmoid	116,105

All models were trained using supervised learning techniques, optimizing the mean squared error (MSE) loss function to minimize the difference between the predicted ($\hat{\mathbf{y}}$) and actual ($\mathbf{y}$) imperviousness values:

$$\mathrm{MSE} = \frac{1}{\mathbf{n}}\sum_{i=1}^{n}(\hat{\mathbf{y}}_i - \mathbf{y}_i)^2,$$

where **n** is the number of samples. Additionally, hyperparameter tuning was conducted to identify the optimal configurations for each neural network architecture. This involved systematically adjusting parameters such as learning rates, the number of neurons in hidden layers, dropout rates, and weight decay factors to enhance model performance and generalization. Regularization techniques, including dropout and weight decay, were employed to mitigate overfitting by preventing the models from becoming overly specialized to the training data.

Early stopping was utilized as a regularization technique to prevent overfitting; training was halted if the validation loss did not improve over a predefined number of epochs (patience). This approach helped in maintaining the model's generalizability to unseen data by ensuring that the models did not continue training once they ceased to show improvements on the validation set.

Model performance was evaluated on the test dataset using standard regression metrics, including MSE and mean absolute error (MAE). The MAE is calculated as:

$$\mathrm{MAE} = \frac{1}{\mathbf{n}}\sum_{i=1}^{n}|\hat{\mathbf{y}}_i - \mathbf{y}_i|,$$

Additionally, residual analyses were conducted to examine the distribution and magnitude of prediction errors, providing insights into model biases and areas for potential improvement. This involved plotting residuals and assessing whether errors were randomly distributed or exhibited patterns indicating model deficiencies.

By comparing the performance of these models against the baseline, the study aimed to determine which neural network architecture most effectively leveraged the embeddings to predict urban imperviousness. The inclusion of optimized simple and deep neural networks allowed for a comprehensive evaluation of different modeling strategies in capturing the complex relationships within the data.

Results

Model Accuracy and Feature Representation

Visualization of the estimates revealed that smaller impervious features, such as roads, are more challenging for the models to predict accurately. This limitation likely stems from the spatial granularity of the data and the representation of such features within the Sentinel-2 imagery. As roads and similar features occupy smaller proportions of each 600×600-meter tile, their spectral signatures may be diluted in the aggregated input. Addressing this issue in future research could involve refining the data splits to ensure sufficient examples of smaller impervious features or employing additional data augmentation techniques.

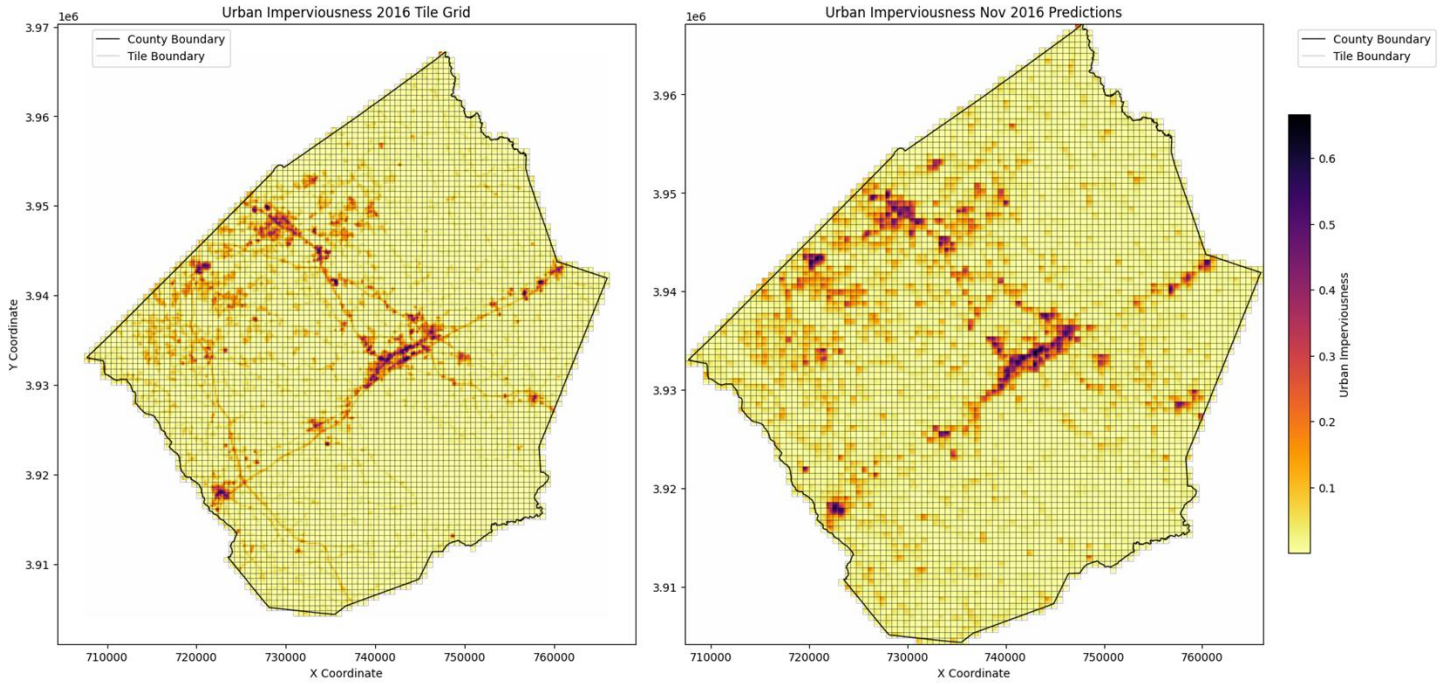

Figure 6 –Comparison of urban imperviousness for 2016. (Left) Actual NLCD imperviousness values. (Right) Predicted imperviousness using the top-performing model (DNN).

Urban Growth Trends

Between 2016 and 2023, Johnston County experienced an average annual population growth of 3.4%, according to U.S. Census data. Over the same period, the model estimated an average increase of 2.44% in urban imperviousness per year. While population growth and urban density are not expected to exhibit a one-to-one relationship, the observed trend aligns with expectations, given the county's rapid urbanization. The discrepancy underscores the importance of further investigating factors influencing imperviousness growth, including development patterns and land-use policies.

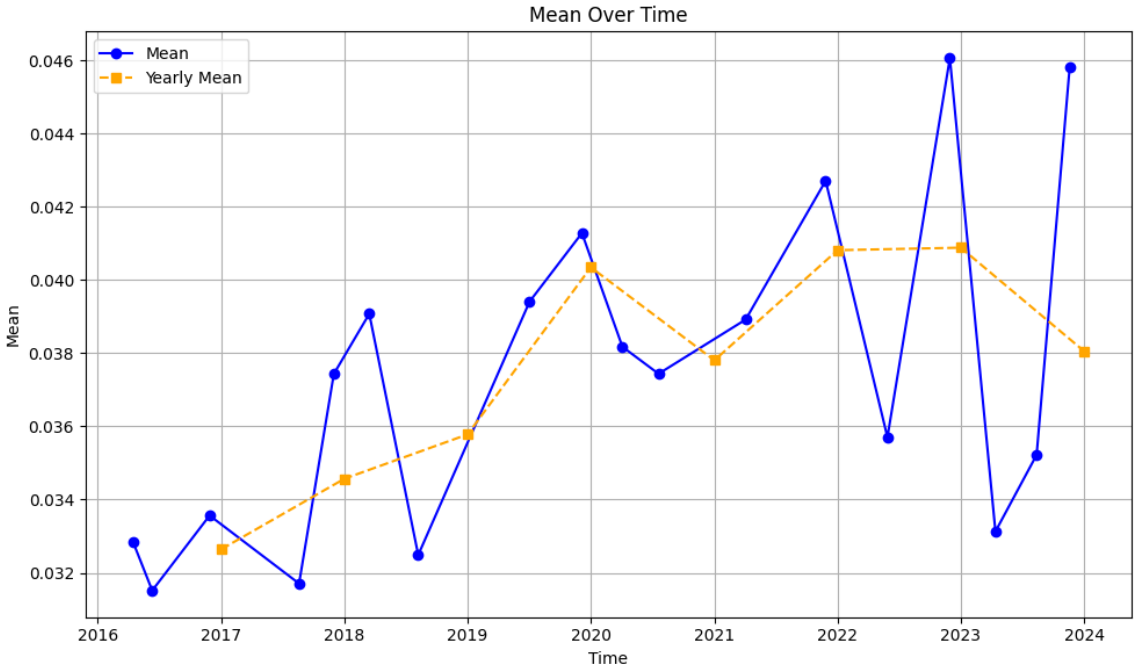

Figure 7 – Mean imperviousness for Johnston County from 2016 to 2023, as estimated by the top-performing model. Blue markers represent temporal variations in imperviousness, while orange markers denote annual averages.

Seasonal Variation in Imperviousness Estimates

The model consistently produced higher urban imperviousness estimates during winter months compared to spring and fall. This pattern likely results from reduced vegetation cover in winter, which minimizes obstruction of urban features in the imagery. Conversely, increased vegetation during spring and fall may obscure impervious surfaces, leading to lower estimates. While these results are consistent with the anticipated effects of seasonal vegetation dynamics, addressing this variability in future work through advanced temporal modeling techniques could further improve predictive accuracy.

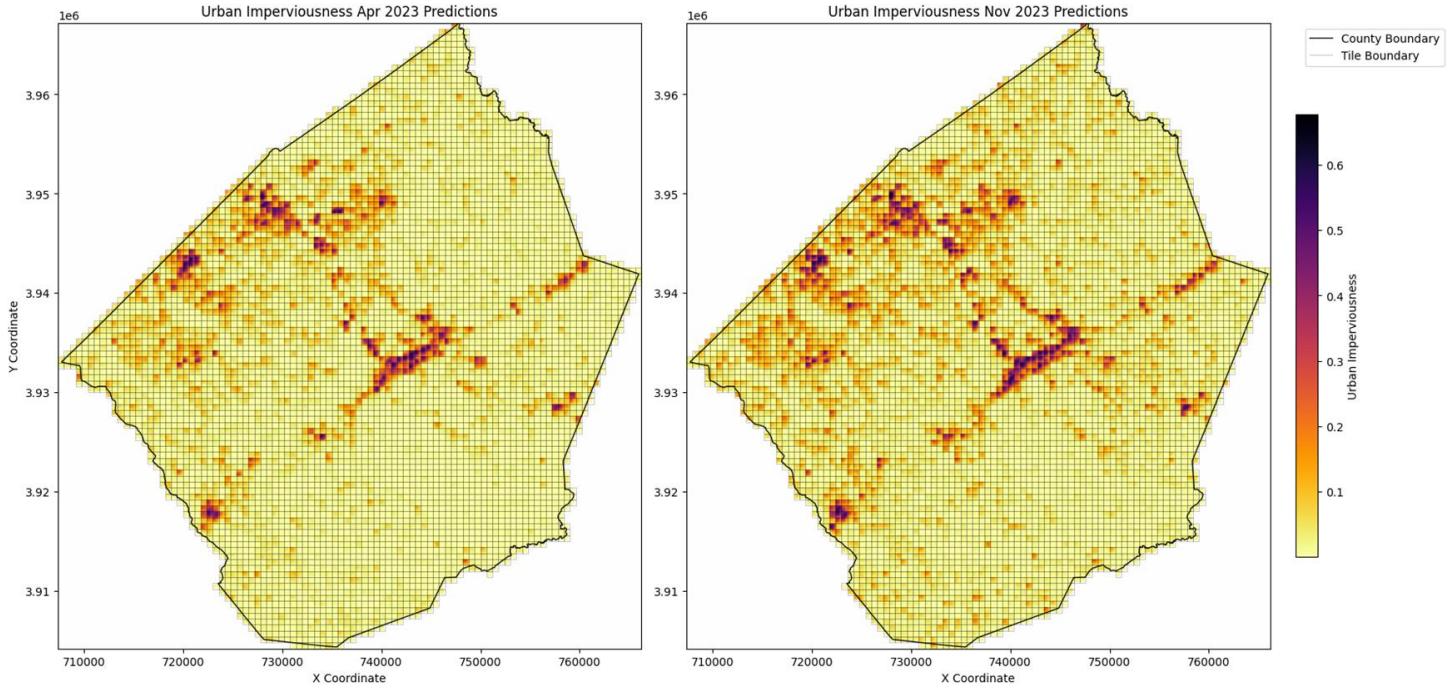

Figure 8 – Model-estimated urban imperviousness for Johnston County. (Left) April 2023 estimates (spring). (Right) November 2023 estimates (winter).

Performance of Neural Network Architectures

The evaluation of neural network architectures demonstrated that deep neural networks (DNNs) consistently outperformed simple neural networks (SNNs) when considering root mean squared error (RMSE) as the primary metric. As shown in Figure 9, DNN models achieved consistently lower RMSE values, indicating superior ability to capture the spatial and spectral patterns necessary for accurate predictions of urban imperviousness. The DNN models benefited from their deeper architecture, which allowed for the extraction of more complex hierarchical features from the Clay model's spatial embeddings. In contrast, SNNs exhibited higher variability in performance, as evidenced by the larger interquartile range in Figure 9.

Hyperparameter optimization revealed that the most effective DNN models shared common characteristics, including moderate dropout rates (0.1–0.2), small weight decay values (e.g., 0.0001), and hidden layer sizes of 64–128 units. These configurations achieved a balance between regularization and model capacity, enabling the DNNs to generalize well without overfitting to the training data. Table 1 highlights the top 10 models ranked by RMSE, where these hyperparameters consistently appeared. The use of learning rates between 0.0005 and 0.001 further supported stable convergence during training.

Interestingly, when mean absolute error (MAE) was used as the evaluation metric, SNN models emerged as competitive or superior in certain cases. Table 2, which ranks models by MAE, shows that

multiple SNN architectures achieved lower MAE values than many DNN models. This behavior can be attributed to the fundamental differences between RMSE and MAE as evaluation metrics. RMSE penalizes larger errors more heavily, favoring models that effectively reduce extreme outliers, whereas MAE treats all errors equally, making it more robust to smaller deviations. Consequently, SNNs' simpler architectures, with smaller hidden layers (e.g., 32 neurons) and higher dropout rates (0.3–0.5), may have been better suited to minimizing consistent, smaller errors in the predictions.

This divergence between metrics demonstrates the importance of aligning model selection with the specific objectives of the analysis. For applications prioritizing the reduction of extreme errors or capturing overall spatial variability, DNNs are the preferred choice. However, for scenarios where consistent accuracy across predictions is more critical, such as monitoring small-scale features like roads, SNNs may offer distinct advantages.

Figure 9 illustrates the distribution of RMSE values across the top 125 models for each architecture type. The DNNs not only achieved lower median RMSE values but also displayed less variability, reflecting their robustness in capturing complex spatial relationships. In contrast, the wider spread of RMSE values for SNNs indicates their sensitivity to hyperparameter configurations and reduced capacity to generalize across diverse spatial patterns.

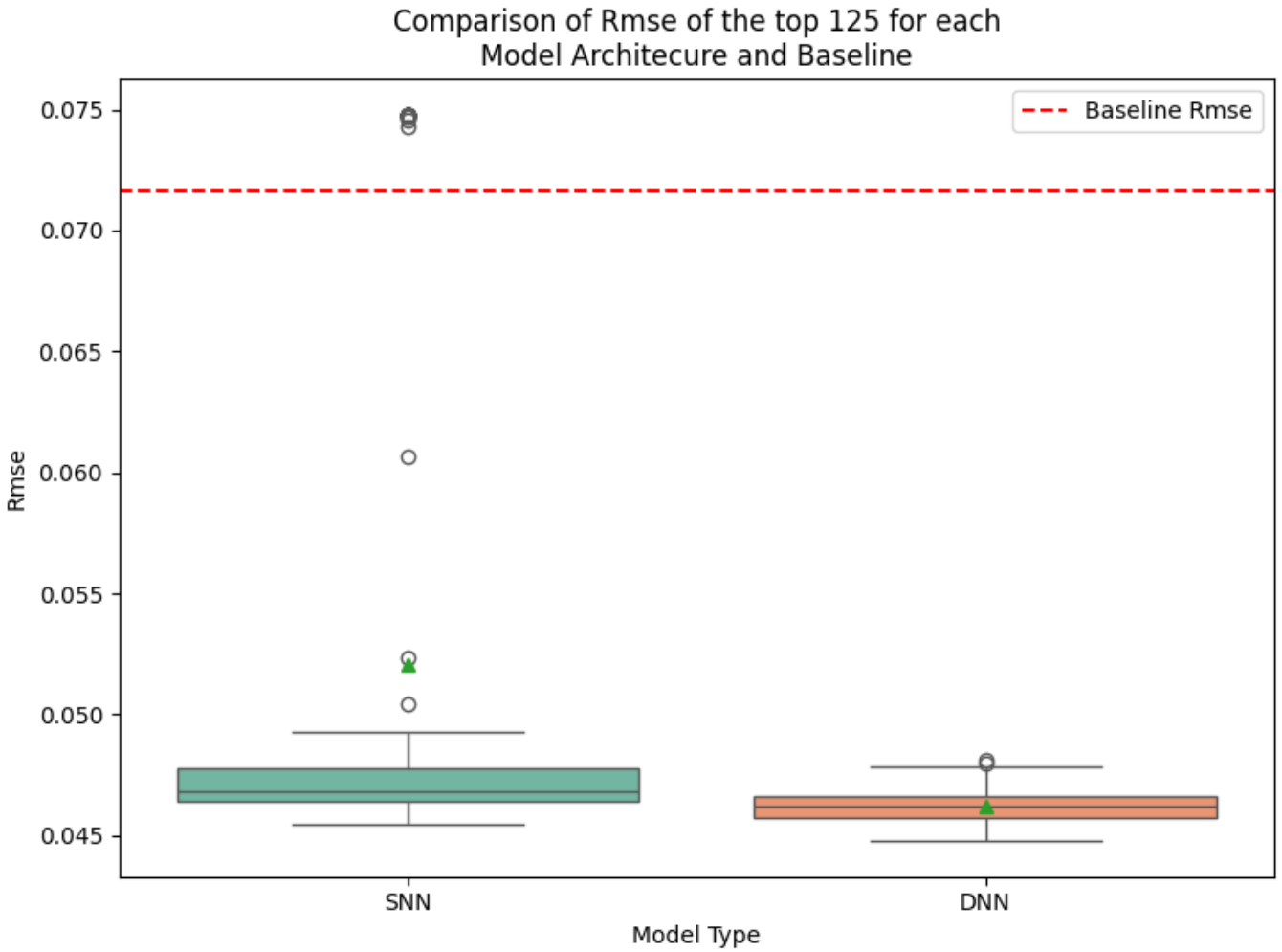

Figure 9 – RMSE comparison of the top 125 simple neural networks (SNNs) and deep neural networks (DNNs), with the baseline RMSE indicated by the red dashed line. Lower RMSE values indicate better performance. The DNNs achieved both lower median RMSE values and reduced variability compared to SNNs.

Table 3: Top 10 models ranked by RMSE, with hyperparameters and performance metrics.

Model Type	RMSE	MSE	MAE	Hidden Units	Dropout Rate	Weight Decay	Learning Rate	Patience	Criterion
Deep NN	0.0448	0.0020	0.0209	128	0.1	0.00010	0.0005	10	MSE Loss
Deep NN	0.0451	0.0020	0.0196	64	0.1	0.00000	0.0005	10	MSE Loss
Deep NN	0.0452	0.0020	0.0192	128	0.2	0.00010	0.0005	20	MSE Loss
Deep NN	0.0453	0.0020	0.0210	64	0.1	0.00010	0.0010	10	Smooth L1 Loss
Deep NN	0.0453	0.0021	0.0186	64	0.1	0.00010	0.0010	10	MSE Loss
Deep NN	0.0453	0.0021	0.0214	128	0.1	0.00010	0.0010	20	Smooth L1 Loss
Deep NN	0.0453	0.0021	0.0191	128	0.1	0.00010	0.0005	20	Smooth L1 Loss
Deep NN	0.0453	0.0021	0.0196	128	0.1	0.00000	0.0005	10	MSE Loss
Deep NN	0.0453	0.0021	0.0203	64	0.1	0.00001	0.0005	10	MSE Loss
Deep NN	0.0454	0.0021	0.0201	128	0.1	0.00010	0.0010	30	MSE Loss

Table 4: Top 10 models ranked by MAE, with hyperparameters and performance metrics.

Model Type	RMSE	MSE	MAE	Hidden Units	Dropout Rate	Weight Decay	Learning Rate	patience	criterion
Deep NN	0.0463	0.0021	0.0182	128	0.1	0.00001	0.0005	10	Smooth L1 Loss
Deep NN	0.0482	0.0023	0.0182	32	0.5	0.00010	0.0010	30	Smooth L1 Loss
Deep NN	0.0482	0.0023	0.0183	64	0.5	0.00010	0.0005	30	MSE Loss
Simple NN	0.0477	0.0023	0.0183	64	0.5	0.00010	0.0010	20	Smooth L1 Loss
Deep NN	0.0485	0.0024	0.0183	32	0.3	0.00010	0.0005	30	MSE Loss
Simple NN	0.0489	0.0024	0.0183	32	0.5	0.00010	0.0010	30	Smooth L1 Loss
Simple NN	0.0493	0.0024	0.0184	32	0.5	0.00010	0.0010	30	MSE Loss
Simple NN	0.0484	0.0023	0.0184	32	0.3	0.00010	0.0010	30	MSE Loss
Simple NN	0.0462	0.0021	0.0185	64	0.3	0.00010	0.0010	20	MSE Loss
Simple NN	0.0469	0.0022	0.0185	32	0.3	0.00001	0.0005	20	MSE Loss

Discussion

The findings of this study highlight the potential utility of integrating foundation models, such as the Clay Foundation Model, with Sentinel-2 imagery for urban growth monitoring, as demonstrated through this proof-of-concept applied to Johnston County, North Carolina. The deep neural networks (DNNs) consistently outperformed simple neural networks (SNNs) in terms of root mean squared error (RMSE), indicating their ability to capture complex spatial and spectral relationships in urban imperviousness data. However, when mean absolute error (MAE) was used as the evaluation metric, several SNNs performed competitively or even better than some DNNs, suggesting that SNNs may be more effective in reducing consistent, small-scale prediction errors. This divergence shows the importance of aligning evaluation metrics with the specific objectives of urban monitoring tasks, such as identifying small impervious features like roads or mitigating the impact of outliers.

One notable challenge observed was the seasonal variability in imperviousness estimates. Winter months consistently yielded higher estimates due to reduced vegetation cover, which minimizes obstruction of urban features in the imagery. In contrast, spring and fall exhibited lower imperviousness estimates as vegetation obscured built surfaces. While this seasonal effect aligns with expectations, future work could

incorporate more robust temporal modeling techniques or vegetation indices to adjust for these dynamics, enhancing the predictions across seasons. Additionally, the difficulty in accurately predicting smaller impervious features such as roads highlights the need for potentially refining data splits or exploring advanced modeling architectures, such as convolutional neural networks (CNNs), to capture spatial dependencies more effectively.

Conclusion

This study developed and evaluated a data-efficient framework for urban imperviousness prediction by combining the Clay Foundation Model with Sentinel-2 imagery. The results highlight the potential of foundation models to address gaps in traditional urban monitoring datasets by providing more frequent and scalable updates. While deep neural networks showed superior performance overall, the competitive results of simple neural networks under certain metrics demonstrate the importance of tailoring modeling approaches to specific objectives. Future research should focus on addressing seasonal variability, refining data representations for small impervious features, and exploring alternative architectures to further improve predictive accuracy and generalizability. This work contributes to sustainable urban planning by offering a robust methodology for monitoring rapid urban growth in resource-constrained environments.

References

1. Ayush, K., Uzkent, B., Meng, C., Kumar, T., Burke, M., Lobell, D., & Ermon, S. (2021). Geography-Aware Self-Supervised Learning. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (pp. 10181–10190). 10.1109/ICCV48922.2021.01002
2. Clay Foundation. (2023). *Clay Foundation Model: An Open Source AI Model for Earth*. Retrieved from https://www.clay.earth
3. Copernicus Sentinel-2 (processed by ESA). (2021). *MSI Level-1C TOA Reflectance Product. Collection 1. European Space Agency*. https://doi.org/10.5270/S2_-742ikth
4. Dionelis, N., Fibaek, C., Camilleri, L., Luyts, A., Bosmans, J., & Le Saux, B. (2024). *Evaluating and Benchmarking Foundation Models for Earth Observation and Geospatial AI*. arXiv preprint, arXiv:2406.18295. https://doi.org/10.48550/arXiv.2406.18295
5. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., & Girshick, R. (2022). "Masked Autoencoders Are Scalable Vision Learners." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16000-16009. doi:10.1109/CVPR52688.2022.01553
6. Jean, N., Wang, S., Samar, A., Azzari, G., Lobell, D., & Ermon, S. (2019). *Tile2Vec: Unsupervised representation learning for spatially distributed data.* Proceedings of the AAAI Conference on Artificial Intelligence, 33(01), 3967–3974. doi:10.1609/aaai.v33i01.33013967
7. Johnston County Department of GIS. (2011). "Johnston County GIS Data." Retrieved from https://www.johnstonnc.com/gis2/content.cfm?PD=data. Metadata last reviewed on 2011-02-01. Contact: Craig Franklin, GIS System Analyst, Johnston County Department of GIS, 212 East Market Street, Smithfield, NC 27577, USA
8. Mai, G., Cundy, C., Choi, K., Hu, Y., Lao, N., & Ermon, S. (2022). *Towards a foundation model for geospatial artificial intelligence*. Proceedings of the 30th International Conference on Advances in Geographic Information Systems (Article No. 106, pp. 1-4). https://doi.org/10.1145/3557915.3561043
9. Microsoft Open Source, McFarland, M., Emanuele, R., Morris, D., & Augspurger, T. (2022, October 28). microsoft/PlanetaryComputer: October 2022 (Version 2022.10.28) [Computer software]. Zenodo. https://doi.org/10.5281/zenodo.7261897

10. R. Goetzke, M. Braun, H. -P. Thamm and G. Menz (2008). *Monitoring and Modeling Urban Land-Use Change with Multitemporal Satellite Data*. IGARSS 2008 - 2008 IEEE International Geoscience and Remote Sensing Symposium, Boston, MA, USA, 2008, pp. IV - 510-IV - 513, doi: 10.1109/IGARSS.2008.4779770
11. STAC Specification. (2023). *SpatioTemporal Asset Catalog (STAC) API: OpenAPI Definition (Version 1.0.0)*. Retrieved from http://stacspec.org
12. U.S. Census Bureau. (2024, November 28). Census.gov | U.S. Census Bureau. https://www.census.gov/
13. U.S. Geological Survey. (2021). "National Land Cover Database (NLCD) 2019 Products." *U.S. Department of the Interior*. Retrieved from https://www.mrlc.gov/data
14. Zhu, X. X., Tuia, D., Mou, L., Xia, G.-S., Zhang, L., Xu, F., & Fraundorfer, F. (2017). "Deep Learning in Remote Sensing: A Comprehensive Review and List of Resources." *IEEE Geoscience and Remote Sensing Magazine*, 5(4), 8-36. doi:10.1109/MGRS.2017.2762307

Appendix

Pseudo-code For Initial Data Processing

1. **Setup Libraries and Directories**
    - Import required libraries for geospatial, raster, and deep learning processing.
    - Define the directory paths to store tiles, boundaries, and processed data.
2. **Load and Preprocess County Boundary Data**
    - Load `county_boundary.shp` as a GeoDataFrame (`GDF`).
    - Reproject to **EPSG:32617 (UTM Zone 17N)** to match Sentinel-2 data CRS.
    - Calculate bounding coordinates of the boundary with `minx`, `miny`, `maxx`, and `maxy` values and align these to a tile size (600m).
3. **Generate Grid Tiles for the County**
    - Create a grid of 600x600-meter tiles that cover the county boundary.
    - Store these as polygons within a new `GDF`.
    - Ensure each tile intersects with the county boundary; keep full tile geometry.
    - Initialize columns in `GDF` to track whether each tile has been processed and its associated data files.
    - Save the tiles as a GeoJSON file.
4. **Create Unified Boundaries for All Tiles**
    - Combine all tiles into a single unified boundary polygon.
    - Save this as a **shapefile** and create a **raster mask** with a **10-meter pixel resolution**.
    - Define a transform for the raster that aligns with the bounds and CRS of the tiles.
5. **Load and Mask Urban Imperviousness Data**
    - Load **NLCD urban imperviousness raster** and clip it to the county's bounding box.
    - Save this clipped data as an intermediate TIFF file.
    - Resample the clipped raster to **10m and 200m** resolutions, reprojecting it to **EPSG:32617** with **bilinear resampling**.

- Save both resampled TIFF files.

6. **Extract Urban Imperviousness Data for Each Tile**
   - For each tile, extract a 200m-resampled urban imperviousness raster corresponding to that tile's bounding box.
   - Save each extracted tile raster as a TIFF, updating the `GDF` with the path to each tile's raster file.
7. **Query Available Sentinel-2 Dates**
   - Set up a query for **Sentinel-2 imagery** within a specified date range (e.g., 2016-01-01 to 2024-08-31).
   - Filter for images with **<1% cloud cover** using Microsoft Planetary Computer STAC API.
   - Save the available dates to a pickle file.
8. **Select Optimal Dates for Data Collection**
   - Group the available dates by year and select a specified number of dates per year (e.g., quarterly).
   - Store the selected dates as the main temporal dataset.
9. **Download and Save Sentinel-2 Image Tiles**
   - For each tile and each selected date, query STAC items for Sentinel-2 imagery in the tile's bounding box.
   - Extract and save each band (Blue, Green, Red, NIR) as TIFF files, ensuring alignment with the tile bounds and CRS.
   - Update the `GDF` with the file paths to each downloaded image file.
10. **Generate Mosaics and RGB Composites**
    - Create RGB mosaics for the tiles using bands B02 (Blue), B03 (Green), and B04 (Red).
    - Save the merged mosaics as TIFF files, creating RGB composites for visualization.
11. **Merge Subdivided Urban Data Tiles into a Single Raster**
    - Collect all 200m-resampled urban tiles and merge them into a single **GTiff** raster.
    - Ensure the final merged raster retains the CRS and spatial alignment of the tiles.
12. **Assign Data Files to Dates for Each Tile**
    - For each tile, associate its downloaded Sentinel-2 data with specific dates.
    - Update the `GDF` to track which dates are associated with available data files, facilitating data retrieval for modeling.
13. **Error Handling and Cleanup**
    - Identify missing files, incomplete downloads, and tiles with multiple files for the same date.
    - Re-query missing data and remove redundant or incomplete files.

Model Embeddings and Train/Test Data Setup Pseudocode

1. Setup Environment

- Import required libraries for geospatial processing, deep learning, data handling, and utilities.

- Set up directories and file paths for data, tiles, and model checkpoints.
- Define parameters such as Sentinel-2 bands, spatial resolution (10m), and coordinate reference system (EPSG:32617).

2. Load Data

- Load the geospatial data files (tiles_with_dates.geojson and county_boundary.shp) as GeoDataFrames.
- Define data directories and mapping for the Sentinel-2 bands to color labels (e.g., B02 -> Blue).
- Load available dates from a pre-saved pickle file to check available imagery for each tile.
- Load the 20 selected dates used by the Sentinel-2 data (~3 per year, 2016-2024).

4. Initialize Clay Model for Embeddings

- Define and load the Clay Foundation Model from the checkpoint file.
- Set the model to evaluation mode and load it onto the available device (GPU or CPU).

5. Retrieve and Stack Sentinel-2 Image Data for Each Tile

- Define a function to retrieve image data (bands) for each tile based on its geometry.
- Stack multispectral image data for each selected date, aligning with the Sentinel-2 bands.
- Normalize data based on band-specific means and standard deviations.

6. Generate Additional Features

- Calculate the temporal embedding for each date using sine and cosine transformations to encode week and hour information.
- Calculate normalized latitude and longitude embeddings based on each tile's centroid.

7. Prepare Data Cubes for Embedding Extraction

- Create a data cube for each tile by combining spectral, temporal, and spatial information.
- Normalize pixel values and generate lat/lon and temporal embeddings as per the model's expected format.

8. Extract Embeddings for Each Tile and Save

- For each tile, run the model to generate spatial embeddings for the date-specific data cube.
- Store embeddings and associated dates within the GeoDataFrame for future retrieval.
- Periodically save the updated GeoDataFrame to a pickle file to avoid data loss during long processing runs.

9. Process Urban Imperviousness Data

- Load each tile's 200m urban imperviousness data as matrices and normalize values (e.g., divide by 100 for percentages).
- Save the urban imperviousness data for each tile as a separate GeoDataFrame.

10. Combine Embeddings and Urban Imperviousness Data

- For each tile, extract embeddings, coordinates, and urban imperviousness matrices.
- Flatten embeddings and urban imperviousness matrices, aligning with the spatial and temporal features.
- Save the final merged GeoDataFrame with all relevant features, embeddings, and labels.

Train/Test Split and Model Setup Pseudocode

1. Load and Preprocess Data

- Load the merged GeoDataFrame containing embeddings and urban imperviousness matrices.
- Filter data by date, designating pre-2017 data for training/testing and post-2017 data for future predictions.

2. Define Train, Validation, and Test Splits

- Extract the centroid coordinates of each tile in the GeoDataFrame and store the `x` and `y` values without scaling.
- Check for an existing saved KMeans model. If it exists, load it; otherwise, apply KMeans clustering with `n_clusters` to group tiles based on their centroid coordinates and save the model for future use.
- Assign the predicted cluster labels to a new cluster column in the GeoDataFrame.
- Shuffle the unique cluster labels randomly to avoid spatial bias, and split them into Train, Validation, and Test sets using predefined ratios (e.g., 70% train, 10% validation, 20% test).
- Create a mapping between the cluster labels and their corresponding set (Train, Validation, or Test).
- Add a set column to the GeoDataFrame, mapping each tile's cluster to its assigned set.
- Validate the splits by checking proportions, geographic distribution, and visualizing the splits to ensure spatial consistency across clusters..

3. Define Feature and Target Columns

- Identify the columns representing model features (e.g., feature_0 to feature_767) and target (urban imperviousness matrix).
- Reshape the urban imperviousness data into 3x3 matrices for each tile to be used as model targets.

4. Prepare Data for Deep Learning Model

- Convert features and targets for each dataset (train, validation, test) into NumPy arrays.
- Reshape urban imperviousness matrices to a flattened 9-element vector per tile for compatibility with model input.

5. Convert Data to PyTorch Tensors

- Convert the feature arrays and target arrays into PyTorch tensors.
- Flatten the 3x3 matrices to 1D arrays of 9 elements to simplify the training process.

6. Save Data for Model Training

- Print dimensions to confirm shapes align with model input requirements.
- Finalize and save the processed data as tensors in PyTorch format, ready for training and evaluation.