Satellite Settlement Classification

Vincent Zhang
University of Virginia
School of Data Science
vincent.zhang@virginia.edu

Abstract

We evaluate unsupervised, shallow supervised, and deep learning models for classifying human settlements from true-color satellite imagery, categorizing regions as urban, rural, or uninhabited. Global coverage is considered, excluding areas with minimal human presence. Models compared include k-means clustering, gradient boosting, and convolutional neural networks (CNNs). Ground truth labels are derived from reference raster data. Evaluation is performed under extreme class imbalance, reflecting the predominance of uninhabited regions. While CNNs achieve higher overall accuracy and performance across most metrics, gradient boosting attains superior recall for inhabited areas, making it the most reliable model for detecting settlements in heavily skewed datasets. These findings highlight the trade-offs between model complexity and practical effectiveness in global settlement detection.

1. Introduction

Satellite imagery has become increasingly abundant, with thousands of artificial satellites continuously capturing vast quantities of high-resolution data of the Earth's surface. This volume of imagery contains many spatial and environmental patterns, but manual analysis is very time consuming and inefficient at scale, making machine learning an essential tool for extracting meaningful results. The objective of this research is to evaluate and compare several learning methods in their ability to classify true-color satellite tiles into urban, rural, and uninhabited settlements.

1.1. Data Collection

To ensure that the model is trained on authentic imagery, we restrict our data to true-color satellite products rather than simulated or synthetically generated layers. We obtain our imagery from the Microsoft Planetary Computer [5], which hosts the Harmonized Landsat and Sentinel (HLS) v2.0 collection providing 30-meter multispectral observations with global coverage [2]. This dataset serves as the basis for all experiments, and the following steps describe the acquisition pipeline:

1. For each continent, fetch 100 random queries. Total size estimate $= 6\,\text{continents} \times 100 \times 35\,\text{MB} = 21\,\text{GB}$
2. For each query, download the corresponding TIFs for the RGB color channels.
3. Combine all three channels into a single raster.
4. Save the raster on disk with the file name corresponding to which continent the data was from.
5. Add optional short circuiting to prevent re-downloading when already cached.

To ensure equal coverage of most human settlements in the world, six bounding boxes will be selected, each representing a geographical area of human activity for every inhabited continent. Europe and Asia will be considered separate continents since analysis is done on human and not natural geography. Some additional selection optimizations:

- For North America, much of northern Canada and all of Alaska will be cut off due to low population density.
- For Asia, much of the Siberian wilderness and the Eurasian Steppe will be omitted due to low population density.
- For South America, most of Patagonia will be cut off due to sparse population density.
- For Africa, most imagery will be selected from a bounding box that focuses on sub-Sahara due to similar reasons as above.
- Australia's outback will be kept to see how the model performs.

Edge cases, such as ocean data and Antarctica, may appear sporadically but are not the main focus of the model. Fig. 1 shows the coverage of the world by the training dataset.

The ground truth data for this study are derived from the Global Urban and Rural Settlement (GURS) dataset [4]. This dataset provides comprehensive coverage of urban and rural settlements from 2000 to 2020, offering high-resolution, tile-based mapping of global settlement patterns. Using GURS 2020 ensures that our analyses leverage the most up-to-date and systematically curated settlement data,

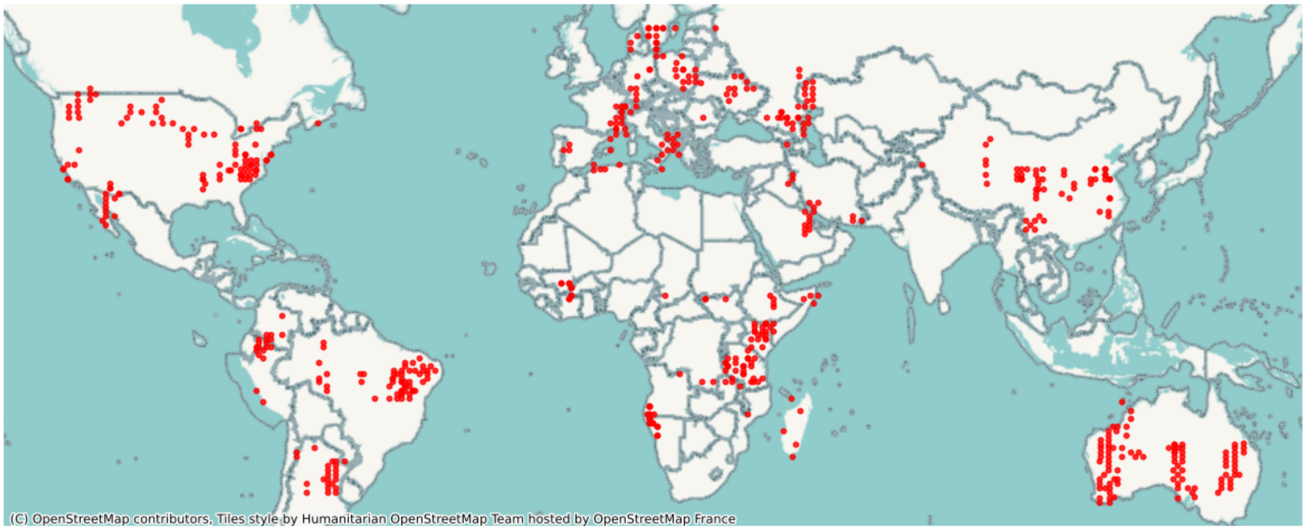
(C) OpenStreetMap contributors, Tiles style by Humanitarian OpenStreetMap Team hosted by OpenStreetMap France

Figure 1. Global map showing the locations of selected satellite tiles, with each red dot indicating a region from which imagery is sampled.

benefiting from the extensive effort undertaken by Liu *et al.* to scan and classify settlements across all tiles worldwide.

2. Data Preprocessing

The raw GeoTIFF imagery from HLS v2.0 comes in various coordinate reference systems, whereas GURS operates in Mollweide [4]. Accordingly, both GURS and all raw imagery are reprojected to EPSG:4326 to ensure spatial consistency. To facilitate efficient processing and avoid loading entire GeoTIFFs into memory, each raster is subdivided into 64 tiles, which enhances surface-level detail for the model. Tiles that contain predominantly empty regions—defined as having more than 20% black pixels—are discarded. All remaining tiles are saved to a dedicated output directory for subsequent use.

2.1. Ground Truth Determination

The ground truths for each tile are extracted from the GURS 2020 raster using two principal steps:

1. Isolate the portion of GURS 2020 corresponding to the geographical extent of each tile.
2. Assign a score to each land-cover label in GURS (urban, rural, uninhabited) reflecting the fractional area of that class within the tile.

2.1.1. Raster Windowing

Let the bounds of a training tile be defined by four latitude-longitude pairs representing the corners of the bounding box, denoted as UL, UR, LL, and LR. The objective is to determine the row and column indices in GURS corresponding to these bounds: $x \in [x_0, x_f]$, $y \in [y_0, y_f]$.

Only two degrees of freedom (UL and LR) are required to define the full bounds. The affine transformation of the raster is used to map the tile coordinates to row and column indices:

$$\mathbf{T} = \begin{bmatrix} 0 & 0 & -180 \\ 0 & -0 & 77.91 \\ 0 & 0 & 1 \end{bmatrix}. \tag{1}$$

Apply the inverse to get row and column bounds:

$$\mathbf{T}^{-1}UL = \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix}, \quad \mathbf{T}^{-1}LR = \begin{bmatrix} x_f \\ y_f \\ 1 \end{bmatrix}, \tag{2}$$

where y represents the rows and x represents the columns. An additional processing step is needed to ensure $x, y \in \mathbb{Z}$:

$$\begin{aligned} x_0 &= \lfloor \min(x_0, x_f) \rfloor, \quad x_f = \lfloor \max(x_0, x_f) \rfloor, \\ y_0 &= \lfloor \min(y_0, y_f) \rfloor, \quad y_f = \lfloor \max(y_0, y_f) \rfloor. \end{aligned} \tag{3}$$

Let $GURS_{wnd} = GURS[y_0 : y_f, x_0 : x_f]$ represent the slice of the GURS raster at the defined bounds. After following these steps above, we get $GURS_{wnd} \in \{1, 2, 127\}^{126 \times 148}$. This is slightly smaller than the training tile's dimensions because of difference in resolution.

2.1.2. Computing Relative Class Frequencies

Define a binary mask for a value n:

$$\mathcal{M}^{(n)}(i,j) = \begin{cases} 1, & GURS_{wnd}[i][j] = n, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

To get the frequency of any value n, define

$$\mathcal{F}(n) = \frac{\sum_{i=y_0}^{y_f} \sum_{j=x_0}^{x_f} \mathcal{M}^{(n)}(i,j)}{126 \times 148 = 18648}. \tag{5}$$

Next, the weights of each inhabited class are set to $100/9$. This is because each GURS pixel is $100 \times 100 = 10000\ \mathrm{m}^2$ whereas the tile pixels are only $30 \times 30 = 900\ \mathrm{m}^2$. The ratio of the weights represents the size differences of the pixel. Uninhabited pixels do not get weighted as the population density is negligible. We use the following vectorized function to ensure normalized weighted frequencies:

$$\vec{F}_w = \frac{\vec{w} \odot \vec{f}}{\vec{1}(\vec{w} \odot \vec{f})}, \tag{6}$$

where $\odot$ is the Hadamard product operator, $\vec{1} \in \mathbb{R}^3$ is the vector of all ones, $\vec{w} = \begin{bmatrix} w_1 & w_2 & w_{127} \end{bmatrix}^\top$, and $\vec{f} = \begin{bmatrix} \mathcal{F}(1) & \mathcal{F}(2) & \mathcal{F}(127) \end{bmatrix}^\top$.

3. Candidate Models

In this section, we outline the training procedures and tuning strategies applied to each of our candidate models. Although all models are trained using soft probability distributions as their target outputs, it is assumed that evaluation proceeds using discrete class assignments. Specifically, a predicted label is obtained by computing argmax over the model's soft probability vector and converting the result into a one-hot encoded representation by setting the selected class entry to 1 and all others to 0.

3.1. K-Means Clustering

We reduce each image tile $\mathbf{I}_i \in \mathbb{R}^{3 \times H \times W}$ to a compact numerical representation. This process should allow k-means to easily cluster the data based on certain visual aspects of the tile. For example, urban tiles are more noisy due to the number of structures. Uninhabited deserts should look smooth and homogeneous. For each band $b \in \{\text{red, green, blue}\}$, define the following statistics over all pixels (x, y) in the band:

$$\mu_{i,b} = \frac{1}{HW} \sum_{y=0}^{H-1} \sum_{x=0}^{W-1} \mathbf{I}_i[b][x][y], \tag{7}$$

$$\sigma_{i,b} = \sqrt{\frac{1}{HW} \sum_{y=0}^{H-1} \sum_{x=0}^{W-1} (\mathbf{I}_i[b][x][y] - \mu_{i,b})^2}, \tag{8}$$

$$P_{i,b}^{(10)} = 10^{\text{th}} \text{ percentile of } \mathbf{I}_i[b], \tag{9}$$

$$P_{i,b}^{(50)} = 50^{\text{th}} \text{ percentile of } \mathbf{I}_i[b], \tag{10}$$

$$P_{i,b}^{(90)} = 90^{\text{th}} \text{ percentile of } \mathbf{I}_i[b], \tag{11}$$

where $0 \leq i \leq 30443$ and H is the height and W is the width in pixels. To capture intra-tile heterogeneity, we convert the tile to grayscale:

$$\begin{aligned} \mathbf{I}_{i,\text{gray}} = 0.2989\, \mathbf{I}_i[\text{red}] + 0.587\, \mathbf{I}_i[\text{green}] \\ + 0.114\, \mathbf{I}_i[\text{blue}]. \end{aligned} \tag{12}$$

These coefficients are derived from the ITU-R BT.601 (Rec. 601) standard [1], which defines how to convert RGB (in the sRGB color space) to luma, i.e., perceived brightness. We then can compute a noise statistic based on the gradient magnitude using the Sobel operator:

$$|\nabla \mathbf{I}_{i,\text{gray}}| = \sqrt{\left(\frac{\partial \mathbf{I}_{i,\text{gray}}}{\partial x}\right)^2 + \left(\frac{\partial \mathbf{I}_{i,\text{gray}}}{\partial y}\right)^2}, \tag{13}$$

$$\mu_{i,\text{noise}} = \mathbb{E}\big[\, |\nabla \mathbf{I}_{i,\text{gray}}| \,\big], \tag{14}$$

$$\sigma_{i,\text{noise}} = \sqrt{\mathbb{E}\Big[\big(|\nabla \mathbf{I}_{i,\text{gray}}| - \mu_{i,\text{noise}}\big)^2\Big]}. \tag{15}$$

Lastly, we can capture local heterogeneity by dividing the grayscale image into $n \times n$ patches $\{P_k\}_{k=1}^{n^2}$, and computing the variance within each patch:

$$\mu_{i,\text{patch}} = \frac{1}{n^2} \sum_{k=1}^{n^2} \mathrm{Var}(P_k), \tag{16}$$

$$\sigma_{i,\text{patch}} = \sqrt{\frac{1}{n^2} \sum_{k=1}^{n^2} (\mathrm{Var}(P_k) - \mu_{i,\text{patch}})^2}. \tag{17}$$

The final feature vector $\vec{x_i} \in \mathbb{R}^{1 \times 19}$ combines all of these statistics:

$$\vec{x_i} = \begin{bmatrix} \mu_{i,\text{red}} & \sigma_{i,\text{red}} & P_{i,\text{red}}^{(10)} & \cdots & \sigma_{i,\text{patch}} \end{bmatrix}. \tag{18}$$

Stacking each $\vec{x_i}$, we get the feature matrix

$$\mathbf{X} = \begin{bmatrix} \vec{x_0} \\ \vec{x_1} \\ \cdots \\ \vec{x}_{30443} \end{bmatrix} \in \mathbb{R}^{30444 \times 19}. \tag{19}$$

We now want to determine the optimal number of clusters. This is done by finding a K such that maximum curvature is obtained on the elbow plot. Let $\mathrm{WCSS}(K) : \mathbb{N} \to \mathbb{R}$ return the within-cluster sum of squares for a particular K. Then, the formal objective is

$$K^* = \arg\max_K \left(\frac{|\mathrm{WCSS}''(K)|}{\sqrt{(1 + \mathrm{WCSS}'(K)^2)^3}} \right). \tag{20}$$

Qualitatively analyzing Fig. 2, we see that at no point does the curvature reach a significant value. We can still try to determine K^* analytically, but it will be not that useful as it will be a very small value ($\mathrm{WCSS}(K^*) \approx 10^{-9}$). In short, a smooth exponential-looking elbow plot tells us the data lacks sharp natural clusters, which is to be expected of fairly homogeneous satellite imagery. Going forwards, let us try again but this time we use the highest silhouette score instead.

From Fig. 3, there is a clear value of $K = 3$ giving us the highest mean silhouette score. This means having three

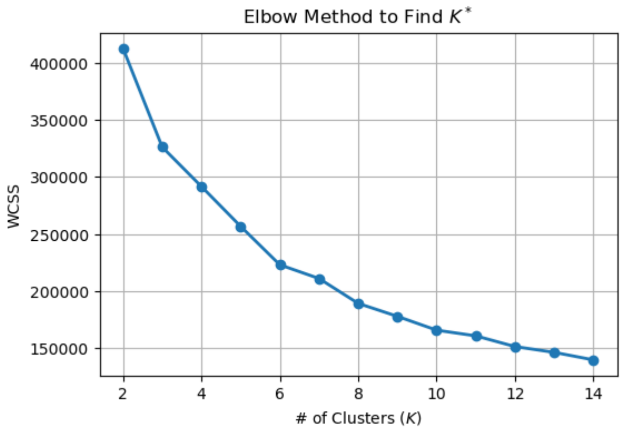

Figure 2. Elbow plot representing WCSS vs. K.

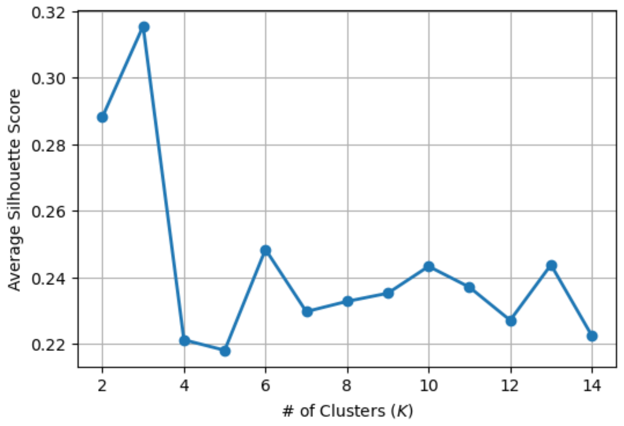

Figure 3. Mean Silhouette Score vs. Number of Clusters.

clusters gives us the best chance. Based on the results, this score tells us that the clusters are reasonably separated, but there is some overlap, which is not excellent, but typical for real-world, noisy, high-dimensional data like satellite tiles. Fig. 4 shows some sample tiles from each cluster to visualize the distinction between each cluster.

3.2. Gradient Boosting

We reduce each image tile $\mathbf{I}_i \in \mathbb{R}^{3\times H\times W}$ to a compact numerical representation. For each band $b \in \{\text{red, green, blue}\}$, define the following statistics over all pixels x, y in b:

$$\mu_{i,b} = \frac{1}{HW}\sum_{y=0}^{H-1}\sum_{x=0}^{W-1}\mathbf{I}_i[b][x][y], \tag{21}$$

$$\sigma_{i,b} = \sqrt{\frac{1}{HW}\sum_{y=0}^{H-1}\sum_{x=0}^{W-1}(\mathbf{I}_i[b][x][y]-\mu_{i,b})^2}, \tag{22}$$

$$\min(\mathbf{I}_i[b]), \quad \max(\mathbf{I}_i[b]), \tag{23}$$

where $0 \le i \le N$ and H is the height, W is the width in pixels, and $N = 30444$ is the cleaned dataset size. The feature vector is

$$\vec{x_i} = \begin{bmatrix}\mu_{i,\text{red}} & \sigma_{i,\text{red}} & \cdots & \mu_{i,\text{green}} & \cdots\end{bmatrix} \in \mathbb{R}^{1\times 12}. \tag{24}$$

Stacking each $\vec{x_i}$, we get the feature matrix

$$\mathbf{X} = \begin{bmatrix}\vec{x_0}\\ \vec{x_1}\\ \cdots\\ \vec{x}_{N-1}\end{bmatrix} \in \mathbb{R}^{N\times 12}. \tag{25}$$

The target matrix $\mathbf{y} \in \mathbb{R}^{N\times 3}$ has each row $\vec{y_i} = \begin{bmatrix}p_{\text{urban}} & p_{\text{rural}} & p_{\text{uninhabited}}\end{bmatrix}$ representing a set of probabilities of each class being the dominant settlement type for each tile. As mentioned earlier, we derive discrete one-hot class labels via $t_i = \arg\max_c \vec{y_i}[c]$, where $t_i \in \{\text{urban, rural, uninhabited}\}$.

To find the optimal hyperparameters, we perform K-fold cross validation wrapped in a grid search. Formally, let $\Theta = \{(\eta, ..., \lambda, \alpha)\}$ denote the hyperparameter space. For each candidate $\theta \in \Theta$, we split the data into K folds and compute the mean validation loss:

$$\bar{\mathcal{L}}(\theta) = \frac{1}{K}\sum_{k=1}^{K}\mathcal{L}_{\text{test}}^{(k)}(\theta), \tag{26}$$

where $\mathcal{L}_{\text{test}}^{(k)}(\theta)$ is the loss of the validation set at the kth fold. Then, the objective is to select

$$\theta^* = \arg\min_{\theta\in\Theta}\bar{\mathcal{L}}(\theta). \tag{27}$$

Once θ^* is found (see Table 1), we fit the model on all training data and evaluate on a hold-out set. The model outputs predicted class probabilities

$$\hat{p_i} = \text{softmax}(F^{(T)}(\vec{x_i})). \tag{28}$$

Due to extreme class imbalance, we also want to oversample minority classes to give more representation and balance out the dataset. After performing 7-fold cross-validation over a grid of 432 candidate models, we evaluate performance using selected metrics, as summarized in Table 3.

3.3. Convolutional Neural Network

We define a mapping $\mathcal{D} : i \mapsto (\mathbf{x}_i, \vec{y_i})$, where $\mathbf{x}_i \in \mathbb{R}^{3\times 256\times 256}$ is a RGB tensor for training. The following steps describe $\mathcal{D}$:

1. We load the raw tile from disk into $\mathbf{X}_i^{\text{raw}} \in \mathbb{R}^{3\times H\times W}$ where $H \times W$ are the original spatial dimensions of the tile.

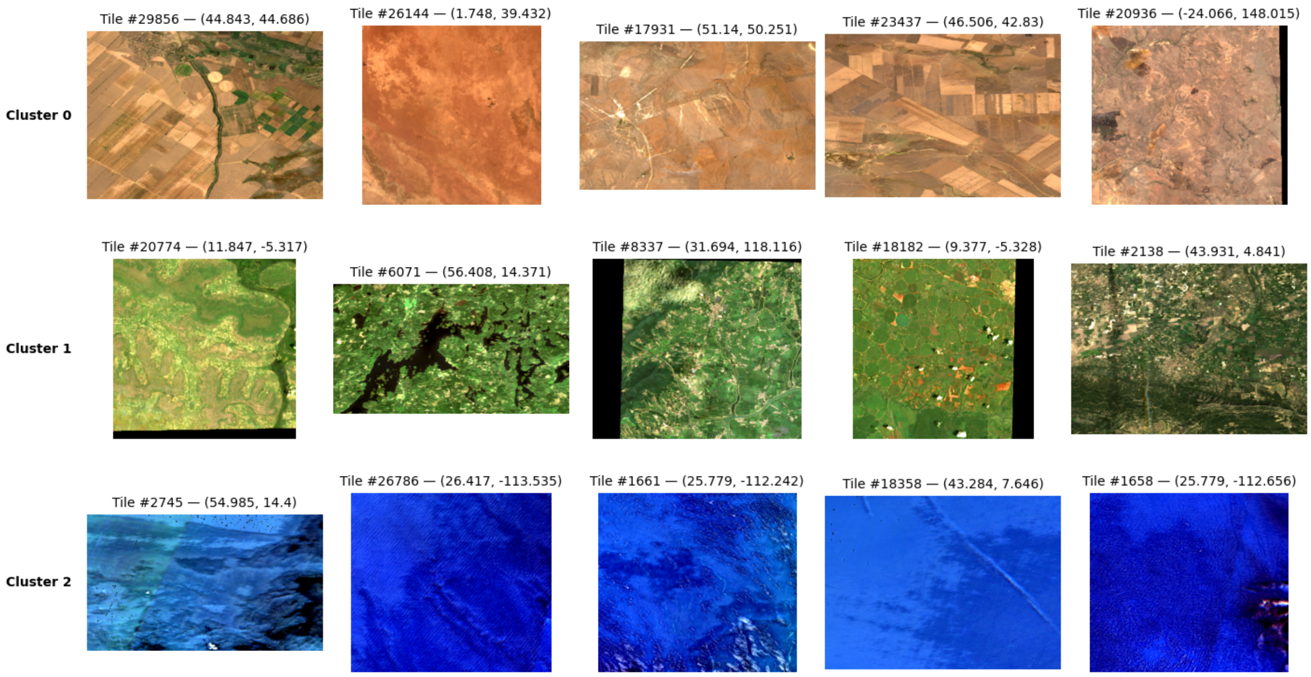
Cluster 0
Tile #29856 — (44.843, 44.686)
Tile #26144 — (1.748, 39.432)
Tile #17931 — (51.14, 50.251)
Tile #23437 — (46.506, 42.83)
Tile #20936 — (-24.066, 148.015)
Cluster 1
Tile #20774 — (11.847, -5.317)
Tile #6071 — (56.408, 14.371)
Tile #8337 — (31.694, 118.116)
Tile #18182 — (9.377, -5.328)
Tile #2138 — (43.931, 4.841)
Cluster 2
Tile #2745 — (54.985, 14.4)
Tile #26786 — (26.417, -113.535)
Tile #1661 — (25.779, -112.242)
Tile #18358 — (43.284, 7.646)
Tile #1658 — (25.779, -112.656)

Figure 4. Sample visualization of the three clusters, shown over representative satellite tiles arranged in a grid. Each row illustrates characteristic imagery associated with one of the identified clusters.

Table 1. Optimal gradient boosting hyperparameters after grid search. Log loss ≈ 0.142.

Hyperparameter	Value
`colsample_bytree`	0.8
`n_estimators`	800
max depth	8
η	0.01
α	0
λ	1
subsample	0.8

2. We normalize to $[0, 1]$:

$$\mathbf{x}_i^{\text{norm}} = \frac{\mathbf{X}_i^{\text{raw}}}{255} \in [0,1]^{3\times H\times W}. \tag{29}$$

3. Finally, we resize to a fixed input shape using bilinear interpolation:

$$\mathbf{x}_i = \text{resize}(\mathbf{x}_i^{\text{norm}}) \in \mathbb{R}^{3\times 256\times 256}. \tag{30}$$

This completes the mapping of $\mathcal{D}$. Let the batch size be B. We select a set of indices

$$\mathcal{I}_{\text{batch}} = \{i_1, i_2, ..., i_B\} \subseteq [0, 30443]. \tag{31}$$

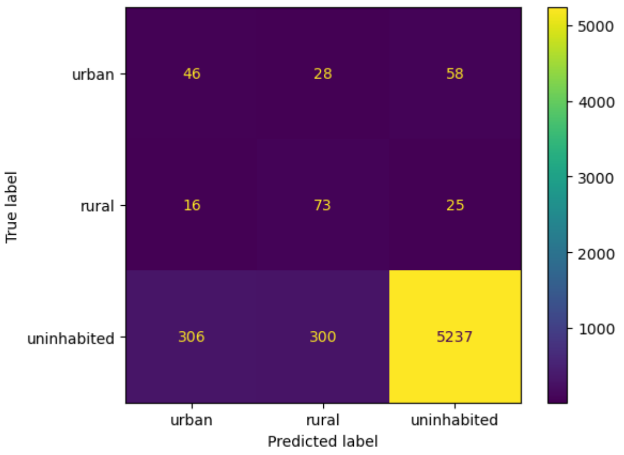

Figure 5. Confusion matrix for gradient boosting predictions on the test set.

Then, the batch tensors are

$$\mathbf{X}_{\text{batch}} = \mathbf{x}[\mathcal{I}_{\text{batch}}] \in \mathbb{R}^{B\times 3\times 256\times 256}, \tag{32}$$

$$\mathbf{Y}_{\text{batch}} = \mathbf{y}[\mathcal{I}_{\text{batch}}] \in \mathbb{R}^{B\times 3}. \tag{33}$$

Over an entire epoch,

$$\mathcal{D} = \bigcup_{t=0}^{\lceil 30444/B\rceil - 1} (\mathbf{X}_{\text{batch}}^{(t)}, \mathbf{Y}_{\text{batch}}^{(t)}). \tag{34}$$

Table 2. Optimal ResNet-34 hyperparameters after grid search. Validation loss ≈ 0.3195.

Hyperparameter	Value
η	0.0001
λ	10^{-5}
p	0.2

We move on to model selection and description. There are many CNNs in PyTorch but we will use ResNet-34 due to hardware constraints of the testbed. He *et al.* [3] report that the 34-layer residual network requires approximately 3.6 billion FLOPs. Given the theoretical peak performance of our GPU (4.4 TFLOPS), this allows for over a thousand forward passes per second. Although actual training will be slower, this margin provides sufficient headroom to avoid significant slowdowns even under realistic workloads.

Similar to gradient boosting, we employ a $K = 5$ fold cross-validation selection scheme. For this model, we select a hyperparameter grid that consists of the learning rate, weight decay, and dropout rate: $\Lambda \in \{\eta, \lambda, p\}$. In this process, we split into K folds: $\bigcup_{k=1}^{K} \mathcal{D}_k$, then train on $\mathcal{D}_k^{\text{train}}$ and validate on $\mathcal{D}_k^{\text{test}}$:

$$\theta^{(k,\Lambda)} = \arg\min_{\theta} \frac{1}{|\mathcal{D}_k^{\text{train}}|} \sum_{i \in \mathcal{D}_k^{\text{train}}} \mathcal{L}_{\text{reg}}^{(i)}(\theta). \tag{35}$$

Compute vectorized validation loss per fold:

$$\mathcal{L}_{\text{test}}^{(k,\Lambda)} = \frac{1}{|\mathcal{D}_k^{\text{test}}|} \sum_{i \in \mathcal{D}_k^{\text{test}}} \mathcal{L}^{(i)}(\theta^{(k,\Lambda)}). \tag{36}$$

Average over folds to select optimal hyperparameters:

$$\bar{\mathcal{L}}_{\text{test}}(\Lambda) = K^{-1} \sum_{k=1}^{K} \mathcal{L}_{\text{test}}^{(k,\Lambda)} \tag{37}$$

$$\Lambda^* = \arg\min_{\Lambda} \bar{\mathcal{L}}_{\text{test}}(\Lambda). \tag{38}$$

We also oversample the minority datasets to account for class imbalance. The best hyperparameters found after a 5-fold cross validation on a grid of 18 candidates are shown in Table 2. As mentioned previously for gradient boosting, selected evaluation metrics for ResNet-34 are summarized in Table 3.

4. Discussion

Evaluating the unsupervised model, we can see that from Fig. 4, despite having statistics that quantify noise, k-means still primarily clustered on color. Cluster 0 mainly has tiles of warmer hues, such as deserts, red soil, and dry vegetation. Cluster 1 is composed mainly of green forestry and grassland tiles, which may indicate a higher presence of human settlements. Cluster 2 corresponds to water tiles, characterized by a distinct and uniform blue texture. Overall, rather than clustering based on human settlements, k-means clustering partitioned the tiles according to biome, which may open opportunities for applications in environmental monitoring, ecological conservation, biodiversity assessment, land-use planning, climate impact studies, and natural resource management.

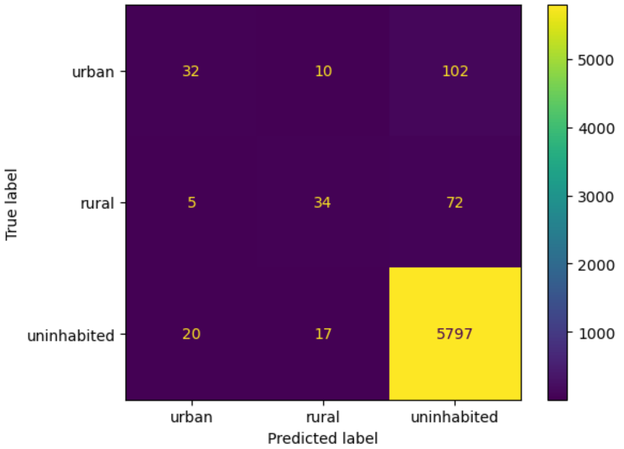

Figure 6. Confusion matrix for ResNet-34 predictions on the test set.

For both supervised learning models, the confusion matrices makes it abundantly clear that uninhabited tiles outweigh urban and rural tiles even minority oversampling is done. From both Fig. 5 and Fig. 6, we can see that the true positives for uninhabited tiles are so numerous that the other predictions are negligible by comparison. Such an imbalance is inevitable and present in virtually all satellite imagery datasets, as the proportion of land covered by human settlements is minuscule compared to uninhabited areas. Importantly, this imbalance is extreme even after sampling and excluding obviously uninhabited regions through bounding box selection.

Another decision for evaluation was reporting macro-averaged recall, precision, and F1 score rather than the standard (micro) metrics. Macro-averaging treats each class equally, regardless of its prevalence, which is particularly important here because uninhabited tiles vastly outnumber inhabited ones as mentioned previously. Overall, ResNet-34 outperformed gradient boosting on all selected metrics except for loss and macro recall. For all metrics shown in Table 3 except loss, higher values indicate better performance. While these metrics could likely be improved with additional training, model tuning, or larger datasets, the inherent characteristics of satellite tiles impose a natural performance barrier that may limit further gains.

Metric	XGBoost	ResNet-34
Loss	0.2932	0.3172
Accuracy	0.8796	0.9629
Macro Precision	0.4305	0.6965
Macro Recall	0.6284	0.5074
Macro F1 Score	0.4686	0.5653

Table 3. Comparison of gradient boosting and ResNet-34 across key evaluation metrics.

4.1. Ethical Considerations

Developing models that identify human settlements carries an inherent risk of misuse, even when the intended purpose is to support beneficial domains such as urban planning, emergency rescue, and disaster response. The same digital system responsible for designing better land-use could also be repurposed for harmful goals, such as unauthorized population tracking, nation-state surveillance, political repression, or selective resource denial. Because the underlying algorithms scale easily once developed, the ethical responsibility lies in recognizing that precise insights from satellite imagery can generate both constructive and destructive actions depending on who controls them.

A second ethical consideration involves responsible and transparent data sourcing, particularly when using large-scale satellite imagery datasets. While these datasets are typically open and designed for research, the obligation remains to ensure that the imagery complies with licensing terms, respects privacy expectations for publicly captured scenes, and does not inadvertently include sensitive or restricted content. Ethical sourcing also requires acknowledging the limitations of satellite imagery, such as resolution constraints, atmospheric artifacts, or temporal gaps, to avoid overstating what the model can reliably infer. Being transparent ensures that the models are explainable, understandable, and interpretable to non-technical stakeholders [6].

The choice of evaluation metrics carries ethical weight as well. In the context of identifying human settlements, recall is often the most important metric we should consider, even though other metrics may have more technically impressive numbers. This is especially the case given our dataset is extremely imbalanced. The primary harm arises from misclassifying a settlement as uninhabited rather than incorrectly flagging uninhabited land. A model with high recall ensures that most true settlements are detected rather than overlooked. In contrast, if we were to focus on overall accuracy or precision, we could fail to detect certain types of settlements, such as small, remote, or marginalized communities. Prioritizing recall therefore acts as a safeguard against creating these blind spots that often mask underrepresented populations.

Finally, it is important to recognize that even as these models become highly performant and precise, the final screening should always be done by a human operator. When dealing with human settlements, a single mistake could impact not only individual lives but entire communities and cultures. Not all human settlements are built equally, and we must be careful in generalizing thousands of unique lifestyles, traditions, cultures, and heritage when applying a machine learning model to predict a class label.

5. Conclusion

In conclusion, based on our experimental results, we recommend *gradient boosting* as the preferred supervised model for this satellite tile dataset. Gradient boosting demonstrates several advantages: it is computationally lighter than deep learning alternatives, making it more feasible for large-scale deployment, and it achieves lower loss and higher recall, which is particularly important given the extreme class imbalance in our data.

Potential improvements for future work include integrating temporal satellite data to capture dynamic changes, incorporating higher-resolution imagery for detecting small or informal settlements, and combining gradient boosting with unsupervised embeddings to better generalize across different geographic regions. Such extensions could expand the model's applicability to scenarios that have not yet been fully explored, including predicting settlement growth, monitoring environmental impacts of urbanization, or supporting policy decisions in undermapped regions.

References

[1] ITU-R Recommendation BT.601: Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios, 2011. Approved 8 March 2011; in force. 3

[2] M. Claverie, J. Ju, J. G. Masek, E. Vermote, J.-C. Roger, S. Skakun, J. Dungan, and C. Justice. The Harmonized Landsat and Sentinel-2 surface reflectance data set. *Remote Sensing of Environment*, 219:145–161, 2018. 1

[3] K. He, X. Zhang, S. Ren, and J Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. IEEE, 2016. 6

[4] Z. Liu, S. Huang, C. Fang, L. Guan, and M. Liu. Global urban and rural settlement dataset from 2000 to 2020. *Scientific Data*, 11:1359, 2024. 1, 2

[5] Microsoft AI for Earth. Planetary Computer. https://planetarycomputer.microsoft.com, 2021. 1

[6] M. Paolanti, S. Tiribelli, B. Giovanola, A. Mancini, E. Frontoni, and R. Pierdicca. Ethical framework to assess and quantify the trustworthiness of artificial intelligence techniques: Application case in remote sensing. *Remote Sensing*, 16(23), 2024. 7