

Satellite Imagery Analysis for Urban Planning and Infrastructure Management

Shatadal Samui ¹

¹Student at School of Computer Engineering, KIIT University,
Bhubaneswar, 751024, Odisha, India.

Contributing authors: 22053361@kiit.ac.in;

Abstract

Urban planning, development and infrastructure growth today rely on analysis of high-resolution satellite imagery. Efficient and accurate analysis of such imagery is essential for monitoring land use and infrastructure changes. Manual analysis of such images is very slow, difficult, and prone to human error, which can lead to inaccurate assessments. Existing solutions are cloud-based which require a high-speed internet connection to upload large images and create privacy issues for sensitive data. Therefore, a high-performance offline application is needed which can run locally on standard laptops and desktops without relying on cloud services. This improves data privacy, works in low or no internet areas, and reduces operational costs. We used a YOLO26-small (9.5 million parameters) trained on the xView dataset to detect urban objects such as buildings, roads, and vehicles directly from GeoTiff imagery. The detected objects are then stored in a database. We used Rust, a low-level systems language for high speed processing and Python for model training. To handle large images, we implemented a virtual tiling algorithm that divides large images into overlapping tiles to fit into the model and reduce memory usage. We also used a spatial grid NMS algorithm to remove duplicate detections. This method runs in $O(N)$ time and is much faster than standard NMS. We deployed the model in ONNX (Open Neural Network Exchange) format with FP16 precision, and it runs on CPUs, NPUs and GPUs. In our tests, GPU processing time was approximately 14 times faster than CPU. It accurately analyzes large images for different use cases.

Keywords: Satellite Imagery Analysis, Object Detection (YOLO), Rust, Concurrency, High Performance Computing, GDAL

1 Introduction

Urbanization has been growing very fast over the last decade. To keep up with this pace, the demand for cost-effective and accurate geospatial analysis and monitoring is very important. It has now become an almost fundamental technique for urban planning, development and infrastructure management as these things require continuous monitoring and assessment of land usage, density of buildings and structures, traffic density and roads networks to provide accommodations to the growing population, proper allocation and distribution of resources. High quality satellite images do provide the necessary details for the above mentioned procedures but these images are very large and cannot be analyzed manually. This leads to the need for a fast satellite image analyzer that can scan large images in local devices to find and classify urban objects. There are existing solutions for satellite data analysis but most of them are cloud based and introduces certain problems like data privacy as data needs to be uploaded and it is time consuming as uploading large images needs high-speed internet.

This project basically addresses the above mentioned problems by introducing a high performance offline application using RUST programming language that can be run locally, by using a YOLO26-small model in it that is trained on the xView dataset. It is deployed in ONNX (Open Neural Network Exchange) format, thus it gives a professional grade result without using any specialized server infrastructure equipment or any other cloud services.

The application that we have made, can bridge the gap between raw geospatial data and municipal decision-making. It uses a processing pipeline system with high throughput with a virtual tiling feature and it also uses a $O(N)$ Spatial Grid NMS (Non-maximum suppression) algorithm. This software can efficiently detect and show the urban objects like buildings, cars etc. and map them to their exact global coordinates. The transformation from raw images to structured information is a huge step for urban planners to move from data acquisition to informed infrastructure in a matter of a few seconds. This application is also very versatile as it can run across GPUs, CPUs and NPUs. It can thus prove to be functional across a variety of hardware environments and give optimal performances at the same time.

This document is organized in such a manner that it can provide a detailed overview of the life cycle of the project. Chapter 2 of this document contains some related works and existing literature that deals with satellite image analysis and urban planning or monitoring. Chapter 3 contains the methodology of the project which includes problem statement, techniques used and proposed methods. Chapter 4 includes test results, model performance and some example images. Chapter 5 is the conclusion of this document and it contains summary of the key technical findings made and it also includes some potential future scopes.

2 Literature Review

Satellite image analysis for city planning and development is a rapidly growing area of study. The main objective of such study is to convert raw geospatial data into usable information from which insights can be generated. New emerging technologies like AI and deep learning are replacing traditional GIS based analysis. But no single method is yet perfect. Most of these methods struggle with scalability, speed and practicality. [Table 1] compares related works, showing the dataset used, methodology, key results, and accuracy.

We have made great strides in making the analysis of satellite images more automated. But most researchers focus on the model score and high accuracy. Processing of high resolution GeoTIFF data, hardware requirements, and real-world deployment are concerns which are easily overlooked. So, we need tools which are not only accurate but also fast and scalable and can work efficiently in real world scenarios.

Adam Van Etten *et al*[1] introduced YOLT which is a specialized framework that significantly improves the existing YOLO framework for geospatial use. They implemented tiling techniques, non-max suppression and a fine grid system. The model was able to detect tiny, dense objects such as cars and small structures that are usually lost in standard analysis. This approach allowed them to maintain high speed and accuracy while processing massive satellite datasets.

Adrian Albert *et al*[2] showed that deep learning can identify how land is used in cities in a much cheaper way than traditional manual surveys. They automated the process by training CNNs on satellite images and map data. The work is unique as their model can be used to compare neighbourhoods and find similarities across various cities. The accuracy of the system varied depending on the location, but the solution was powerful and scalable.

Hannes Taubenböck *et al*[3] developed a method for mapping urban areas using high-resolution satellite images. They used segmentation to group pixels into recognizable objects like roads and buildings. Analysing the shape and surroundings of these objects helped them to identify natural features such as water and vegetation. They utilized Istanbul to show that observing how many buildings are in a location helps us realize how a city is growing.

Sana Basheer *et al*[4] tested how different software and satellite sources affect map accuracy. They compared images from different satellite sources and highlighted that SVM classifier performed the best when used with high resolution images. The team was able to map out things like urban growth and deforestation, and came to a conclusion that right tool and image quality is the key to good monitoring.

Rami Al-Ruzouq *et al*[5] created a technique to use satellite images to monitor city growth over a 40 year span. They used a special process to line up old images to see how the city had expanded and verified their findings against Google Earth data. They discovered that the city expanded greatly from 22% all the way to 92%. Their approach was effective since it maintained accuracy even when the old photos were of poor quality.

Qihao Weng *et al*[6] compared two advanced methods, LSMA and Artificial Neural Networks (ANN) to determine which is better at mapping hard urban surfaces like roads and parking lots. They addressed the mixed pixel issue where a single pixel

Table 1 Comparison of related works in satellite imagery analysis

Author	Dataset	Methodology	Key Results	Accuracy
Adam Van Etten et al.[1]	COWC, SpaceNet	YOLT (YOLO), tiling, NMS	Fast small-object detection in large images	F1-Score: 0.6–0.9
Adrian Albert et al.[2]	Urban Atlas	CNN (ResNet, VGG), transfer learning	Urban land-use classification across cities	70–80%
Hannes Taubenböck et al.[3]	IKONOS	OBIA, segmentation, rule-based classification	Urban structure and density extraction	~82%
Sana Basheer et al.[4]	Landsat, Sentinel, Planet	SVM, RF, CART	High-accuracy LULC classification	Up to 96%
Rami Al-Ruzouq et al.[5]	Landsat	Edge detection + change detection	Long-term urban expansion analysis	84–92%
Qihao Weng et al.[6]	ASTER, Landsat	ANN + LSMA	Improved impervious surface mapping	RMSE: 12–18%
Chongyuan Zhang et al.[7]	WorldView, Sentinel	ML + NDVI + regression	Crop monitoring and yield prediction	R^2 up to 0.94
Nhat-Trinh Le et al.[8]	Sentinel-2	CLIP (vision-language)	Zero-shot urban feature detection	~77%
Taskin Kavzoglu et al.[9]	IKONOS	Spectral + NIR-based classification	Road condition classification	–
L. Gobatti et al.[10]	Landsat	NDVI + LST time-series	Urban cooling effect analysis	$\Delta T \sim 3.5^\circ C$
Stefan Voigt et al.[11]	MODIS, SAR	GIS + multisensor fusion	Rapid disaster mapping	–
Ravindra Kumar Verma et al.[12]	Landsat, IRS	GIS + RS integration	Urban planning and monitoring	–
F. Sunar Erbek et al.[13]	Landsat, SPOT	Image fusion + classification	Accurate land-use mapping	Up to 99%
Jatin Babbar et al.[14]	UC Merced, AID	Segmentation + feature-based ML	Overview of classification techniques	–
Naina Said et al.[15]	CrisisLex, Crisis-MMD	CNN + NLP + multi-modal fusion	Disaster detection using multi-source data	–
Tahereh Nasr et al.[16]	Landsat time-series	MLC + CA-Markov	LULC prediction and green infra analysis	94–98%
Our Work	xView	YOLO, virtual tiling, concurrency, spatial grid NMS	Efficient and accurate object detection in large satellite imagery	F1-Score: upto 0.725

contains multiple materials using satellite data. According to their study ANN was more accurate in recognizing patterns than alternative approach. They also added that summer photos are ideal for analysis because vegetation is more visible.

Chongyuan Zhang *et al*[7] used high-resolution satellite imagery for monitoring crops and improving plant breeding. By utilizing the methods proposed by the authors, researchers can track plant health and growth by analysing satellite signals. In this way the plants will not be harmed. They also showed us that if we combine machine learning with drone data, our predictions become more accurate. The authors believe that these strategies are vital for solving global food security, despite encountering hurdles like cloud cover.

Nhat-Trinh Le *et al*[8] introduces us to a new tool called RemoteCLIP. This tool enables computers to understand satellite images and text together. This allows the system to find houses and factories easily without repeated training by human. In their testing the authors found out that this method is better and flexible than older tools. They concluded by saying that because this technology can rapidly monitor development and dangers, it can be an excellent step towards smart cities.

Taskin Kavzoglu *et al*[9] assessed condition of urban streets and identify asphalt damages by investigating satellite images. Through testing the authors found out that certain light ranges like NIR are sensitive to changes in road quality. City planners can use this method to determine road conditions. The authors proved that using satellite images to monitor road condition is cost effective in large scale inspections.

L. Gobatti *et al*[10] analysed decades of satellite data from Zurich and established a direct link between plant growth and reduction of surface heat. Their research confirmed that satellite imagery is a powerful tool for monitoring how these spaces fight the urban heat island effect. Additionally, their research revealed that whereas tree-based systems take a lot longer to become effective, grass-based systems offer quick cooling. These results imply that while creating plans to cool cities, urban planners should take these various timescales into account.

Stefan Voigt *et al*[11] focused on practical workflow which are required for rapid mapping during disasters. They outlined a full-service cycle which starts from taking picture to analysing them for extraction of real time information to emergency responders. The authors combined different types of satellite data with local geographic information. This approach allowed for fast and accurate damage control during crises. They came to a conclusion that while automated technology is helpful, successful disaster responses rely on human coordination and the ability of people to understand and act on the data globally.

Ravindra Kumar Verma *et al*[12] discussed about the crucial role of GIS and remote sensing technologies in handling India's infrastructural problems and fast urbanization. City planners are able to combine physical map data with social and economic information by using satellite imagery. Their study was more focused on scalable and economical digital systems. The old manual techniques were mostly ignored. The authors concluded their work by deducing that these tools are essential for long-term, data-driven urban development, even though there are still some institutional and technological obstacles.

Naina Said *et al*[13] looked for the best ways to track disasters. They combined social media posts and satellite imagery. The authors created a dataset from the combined information and used AI tools to fuse the data together for analysis. This combined information contained metadata like locations and stamps. The main challenge according to the authors is that satellites are not always overhead to provide real time information. They highlighted that using two sources is cleaner and accurate than one source.

Jatin Babbar *et al*[14] looked at the hard work which goes behind analysing satellite images. They found that the main bottlenecks are handling huge datasets and computing power. The authors divided the pictures into manageable chunks to make things easier. Additionally, they highlighted high-quality benchmark datasets to ensure accuracy throughout the process. They also compared various model training techniques, such as supervised and unsupervised learning. This work provides a fundamental framework for the analysis of satellite images.

F. Sunar Erbek *et al*[15] showed us how satellite image maps are a practical and cheap way to plan cities. The authors created a special process in which they mixed different types of satellite data into one digital system. They used this mix in Istanbul to create 3D views and track growth of buildings and roads. In the end they concluded that using high-resolution satellites is much faster and more accurate than the old-fashioned way of making maps.

Tahereh Nasr *et al*[16] looked at how Tehran’s green spaces are changing and what they might look like by 2040. They used a special computer model to track how fast the city is taking over natural land. They found that green areas and farms have actually grown, but this growth is not good because there isn’t enough water to keep them healthy. The study showed that the city’s nature cannot keep up with all the new buildings. Ultimately, the researchers concluded that for the city to stay liveable, future planning must focus on saving water and protecting the environment.

Our research is unique because it is not just another model. It is a high-performance system which is ready for real world deployment. Most current tools use Python, but it is slow and can cause crashes while processing large satellite images. Our system can operate more quickly and manage big files without running out of memory since we designed it in Rust. In order to eliminate duplicate data, we have included features like virtual tiling and a unique Spatial NMS technique. Additionally, we’ve ensured that our system may function flawlessly on any hardware or computer by utilizing ONNX.

3 Methodology

3.1 Problem Statement

In the current scenario of urban development and infrastructure growth, municipal authorities, city planners, engineers, law enforcement, and various government/non-government agencies need to regularly monitor land use, infrastructure growth, traffic and roads. To do this, these organizations rely on GeoTiff satellite imagery with very high resolution, ranging from 0.3m to 0.01m GSD (Ground Sample Distance). However, these images are just raw pixel data and lack structured information, making it

very difficult and time consuming to manually count buildings, identify illegal structures, or track vehicles and encroachment in remote or restricted areas that are difficult to access. Manually scanning these massive images is extremely slow, and human analysts often miss small objects like vehicles or new structures in remote areas. Also it becomes very difficult to accurately count vehicles and building, etc where objects are densely packed and a single images covers several square kilometers and contains thousands of objects.

But there are solutions that exist which are cloud-based Software-as-a-Service (SaaS) architectures. This creates a massive problem for the local organizations because they need high speed internet just to upload their images to analyze which are often very large and also raise data privacy concerns for sensitive data. These standard tools are rarely optimized or made for consumer grade laptops and desktops which can handle large images because they do not have sufficient memory. Attempting to pass large images through standard models which cant run on consumer laptops and desktops can regularly cause system crashes and Out-of-Memory (OOM) failures.

So, the main goal of this project is to provide a fast, completely offline desktop application that can work in these hardware constrained devices. To fix this, we used a highly optimized setup, a lightweight but accurate model called YOLO26-small (which has only 9.5 million parameters) and combined it with a virtual tiling algorithm. This application automatically cuts huge 4K to 8K images into smaller 896x896 pixel patches before sending it to the model without losing any quality. Since this matches the model's native training resolution exactly, the system can detect and map objects accurately without causing system crashes or Out-of-Memory errors.

3.2 Techniques Used

The Satellite Image Analyzer is made with highly efficient algorithms and technologies, that is why we are able analyze large high quality satellite imagery on standard laptops and desktops. By using low-level system optimizations with lightweight deep learning models, the application achieves professional grade results without requiring specialized server infrastructure or cloud based processing. Our system prioritizes memory efficiency and high speed processing to handle massive GeoTIFF images within limited hardware constraints.

3.2.1 Physical Tiling and Labeling

Before training the model, the raw 4K GeoTiff images must be processed into usable image format. For example, size of the image the model going to take and the label file for each image. It is done by using the physical tiling script which cut large images into the desired size (ex:- 896 by 896 pixels) with some overlap so no objects get cut in between. It uses the TurboJPEG library for image conversion from GeoTiff to JPEG or PNG formats. This process creates the structures training dataset, many 896 by 896 pixels images and their TXT label files. This ensures images did not get down sampled and lose any detail, this way the model learns accurately and efficiently.

3.2.2 Virtual Tiling Algorithm

To solve the problem of system crashes and Out-of-Memory (OOM) crashes during deployment, the prototype uses a virtual tiling algorithm which instead of spitting out the small images physically it stores the data in either VRAM of the GPU or the system RAM depending on which is available and inference is done from there. When a massive 4K to 8K image is loaded, the pipeline programmatically defines a grid of overlapping 896 by 896 patches. This saves time and space because it skips the process of saving the small images in the disk and deletes the small tiles as soon as it gets the coordinates for the objects detected. This way we can process multiple and large files on standard consumer end devices.

3.2.3 YOLO26s and ONNX Runtime

We used the YOLO26s model as a base model which has 9.5 million parameters. We chose this model because of its lightweight architecture and other heavy transformer based models cannot be run on hardware constrained devices whereas this is CNN and attention based. The trained PyTorch model is exported in ONNX (Open Neural Network Exchange format). We further converted the ONNX model from Full-Precision (FP32) to Half-Precision (FP16), which significantly increases speed and reduces the VRAM usage on consumer-grade GPUs. This allows the application to run at high speeds across different hardware environments, including CPUs and GPUs, without requiring a heavy Python dependency.

3.2.4 Spatial Grid NMS

To handle the duplicate detections caused by overlapping virtual tiles, we implemented a custom Spatial Grid Non-Maximum Suppression (NMS) algorithm. Unlike standard NMS, this approach runs in $O(N)$ complexity, allowing the system to filter thousands of duplicate detections almost instantly. We did this by mapping the image onto a basic grid. The standard method forces the computer to check every single object against everything else on the map, which takes a massive amount of time ($O(N^2)$ time complexity). We just assign each item to a specific square based on its center point. When removing duplicates, the system only looks at that exact square and the ones that are adjacent to it. Since it no longer compares objects on other parts of the image, it skips thousands of useless calculations. The precision stays the same, but the processing time reduces significantly.

3.2.5 Coordinate Mapping

As pixel coordinates are not useful for mapping the object on the original images, the application converts these into global actual coordinates. We utilized GDAL Affine Transforms to accurately convert the localized pixel coordinates back to the original GeoTiff's coordinates. This converts the $(x_{min}, y_{min}, x_{max}, y_{max})$ bounding boxes into exact Global Geospatial Coordinates (Latitude and Longitude), allowing users to pinpoint the exact location of detected objects on original GeoTiff image and the map.

3.2.6 Concurrency and Memory Safety

To properly process the large images in low compute devices, we used Rust as the primary implementation language. It is used to ensure memory safety and implement parallelism using Rayon. Using parallelism or using multiple cores of a CPU at the same time to process images makes it faster and efficient. The language's strict borrow-checker removes memory leaks and null-pointer de-references during heavy data processing.

3.3 Proposed Methods

The Satellite Imagery Analyzer is composed of 3 different stages: preparing the dataset from the 4K GeoTiff images, training the model, exporting it to a lightweight and deployable format and using it in efficient system so that it works in low compute power devices like standard laptops and desktops totally offline. A combination of high performance system and deep learning methods has been used efficiently in this work. Finally all these modules are tested and validated.

3.3.1 Dataset Preparation

This step starts with creating high quality training and validation datasets which consists of images in JPEG or PNG format and label files in TXT format from large 4K GeoTiff images and GeoJSON files. It is written entirely in Rust to maximize safety and performance, this module relies on the GDAL (Geospatial Data Abstraction Library) to read massive, uncompressed GeoTIFF images and convert them into JPEG files of required size. It utilizes Rayon for parallel data processing across multiple CPU cores, and TurboJPEG for high-speed image compression. [Figure 1] shows the steps of data preprocessing for dataset preparation and further model training.

- **Image Tiler and Label Mapper component:**

Since the raw GeoTiffs are too large for GPUs to handle during training and inference stages we developed a Image Tiler component. This physically crops the large images into small fixed sized tiles or small images with 10 to 20 percent overlap ensuring no objects are cut in between and converts them into JPEG format using the TurboJPEG library which compress the image to be able to fit it in the memory. At the same time a different component Label Mapper uses the *Serde* library to parse GeoJSON files and create the label files for each tile in TXT formats which has the coordinates and class ids for training.

At last the system sets the unique image Ids and uses the *Rand* library to automatically shuffle and split the data into ratio of 9:1 training and validation sets and giving a configured *data.yaml* for training the model.

3.3.2 Model Training and Exporting

The model trainer is written in Python. It uses the PyTorch and the Ultralytics YOLO framework, relying on NVIDIA CUDA library to talk with the GPU's Tensor Cores directly for training the model. This module starts with required checks before training

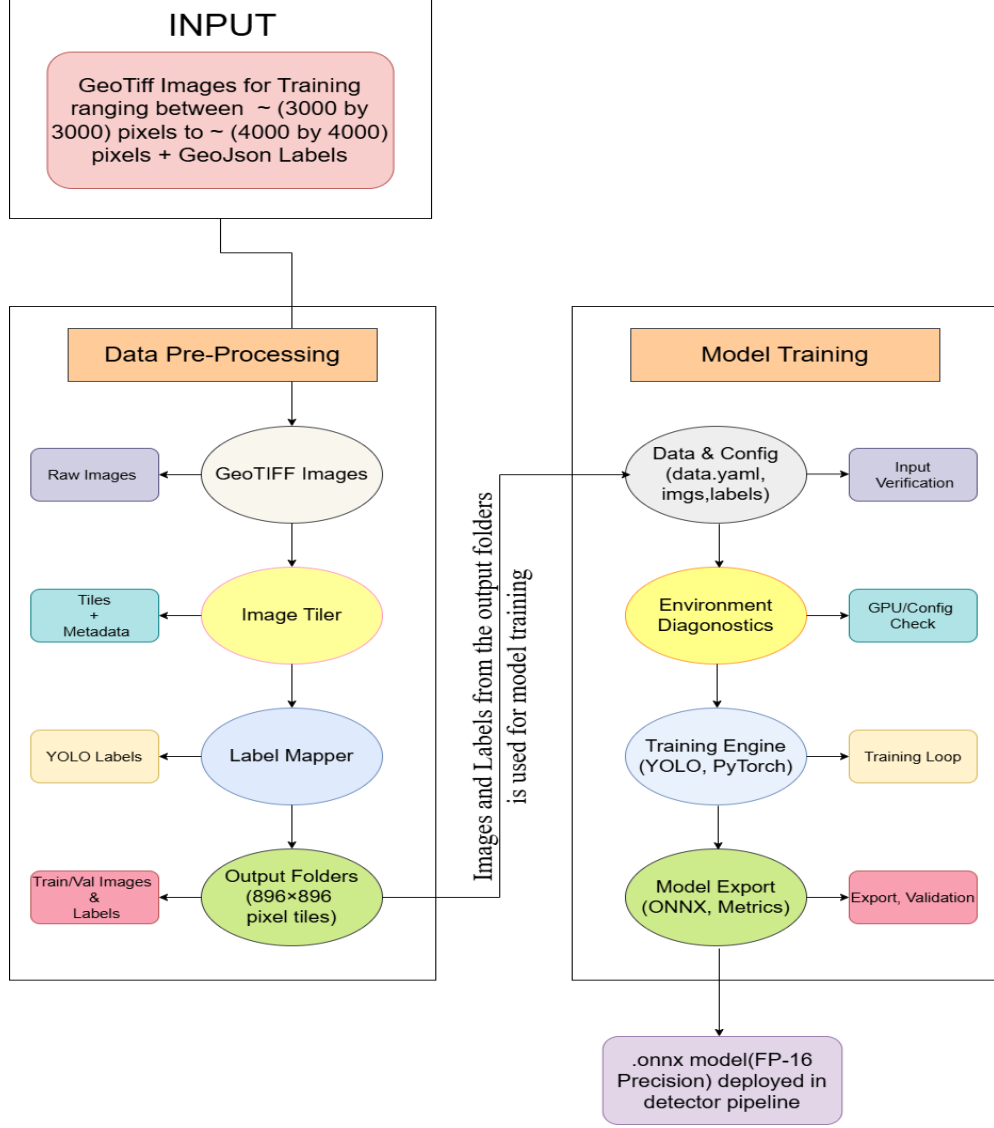


Fig. 1 Data Preprocessing and Model Training

starts using the Torch library verifying there is an Nvidia GPU available for CUDA compute and then if it succeeds the training starts.

- **Training Constraints:**

The trainer pulls the data.yaml configuration file generated before and then the system starts the training using predefined constraints for training such as image size, batch size, learning rate, weight decay, losses, optimizers, epochs, etc optimized for

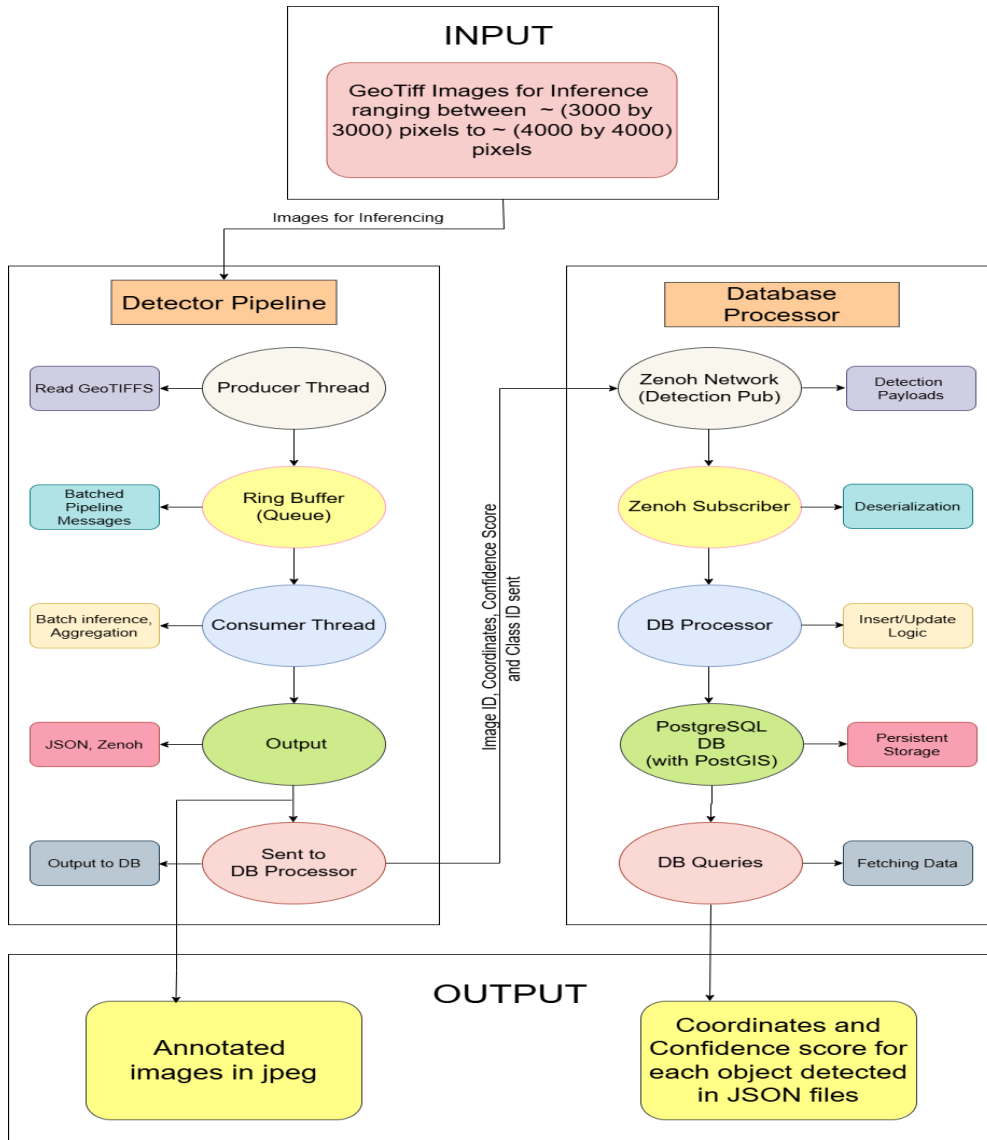


Fig. 2 Inference Pipeline and DB Integration

high-resolution satellite imagery. To monitor performance in real-time, the module uses tqdm for progress tracking and uses NumPy library to extract and analyze validation metrics and spit them out in a csv file. This helps for a precise evaluation of the model's accuracy, specifically focusing on the mAP (Mean Average Precision) for detecting small objects in different areas.

- **Model export to ONNX format:**

After the training is complete the best performing PyTorch check point or *best.pt* is taken and exported into ONNX format and some scripts are run to change the internal weights to FP16 from FP32 which increase the speed and reduces the memory usage by half. Changing to FP16 or half precision decreases accuracy by a very small amount which is negligible.

3.3.3 Inference

This is the main, high-performance stage of the project. [Figure 2] shows the inference pipeline and database integration. It processes large satellite images in GeoTiff format and detects objects. It stores them in a database. Implemented in Rust for maximum speed, this module utilizes the ONNX Runtime (ort) configured with the CUDA Execution Provider to run models directly on the GPU and it automatically shifts back to use CPU when GPU is not available. It uses Tokio Runtime for asynchronous functions and Crossbeam to create fast, memory safe channels.

- **Virtual Tiling Algorithm:**

This algorithm which automatically splits large 4K GeoTIFF images into fixed-sized overlapping tiles according to the model requirement storing them in memory without writing them to disk. These tiles are sent via Crossbeam channels to a consumer thread, which batches them into groups of 4 to 32 and normalizes the pixel data into tensors and correct planar configurations using parallel processing via Rayon. After this it is sent to the model for inference.

- **Inference & Post-Processing:**

The pixel data are put into the ONNX model, using the NVIDIA Tensor Cores for high-speed inference. The raw outputs are then parsed, and an optimized Spatial Grid Non-Maximum Suppression (NMS) algorithm is applied to remove duplicate detections. Finally, the localized pixel bounding boxes are verified, efficiently batched and put in a JSON files for each image and published to the network via Zenoh to the database.

- **Global Coordinate Mapping & Storage:**

After the inference, outputs are pixel-based bounding boxes, the processor applies a GDAL Affine Transform to convert local pixel coordinates into precise global coordinates (latitude/longitude) for the original GeoTiff image. The Database Writer component then batches these converted detections and runs bulk INSERT queries, inserting both pixel-level and geospatial-level database tables. By indexing these geographical data via PostGIS, the system ensures that queries can perform very fast complex spatial queries on the detected objects.

4 Results and Discussion

4.1 Test Results

To ensure that the application was stable enough for the analysis of objects, important system behaviors were checked against in [Table 2], specific constraints and their results are recorded.

Table 2 Testing and Verification

Test ID	Test Case	Test Condition	System Behavior	Expected Result
1	Large Data Ingestion (OOM Prevention)	GeoTIFF exceeds VRAM	Virtual tiling splits image into patches	No Out-Of-Memory crash
2	NMS Duplicate Resolution	Object split across tiles	Spatial Grid NMS applied	Duplicate detections merged
3	Precision Handover Integrity	FP16 used in post-processing	Cast back to FP32	No numerical instability
4	Geospatial DB Integration	Detection sent via Zenoh	Data deserialized + GDAL transform	Stored correctly in PostGIS

4.2 Model Performance

During the final model training run (Run 14), detailed performance metrics were recorded. [Table 3] shows the validation summary for the final model’s performance metrics. The Run 14 results show strong model scores, particularly in core urban infrastructure objects. The model showed excellent detection accuracy for Small Vehicles (mAP50: 0.751) and Buildings (mAP50: 0.686) across over 99,000 combined instances in the validation sets. As expected in complex satellite imagery datasets, rarer infrastructure classes (such as containers or tanks) showed lower confidence.

Table 3 Performance Metrics of Model

Class	Images	Instances	Precision	Recall	mAP50	mAP50-95
All Classes	1906	109732	0.500	0.427	0.425	0.222
Long Truck	555	3633	0.473	0.365	0.386	0.192
Boxy Truck	671	4648	0.458	0.344	0.359	0.197
Small Vehicle	921	43962	0.666	0.796	0.751	0.309
Building	1025	55930	0.595	0.709	0.686	0.370
Container	32	112	0.222	0.214	0.176	0.113
Construction Vehicle	202	717	0.557	0.491	0.507	0.294
Tank	52	145	0.474	0.269	0.272	0.154
Container Lot	85	585	0.558	0.231	0.261	0.149

- **Inference Speed:**

To verify the application’s performance against the offline deployment constraints, the final YOLO26s architecture was tested across different local hardware constraints. [Table 4] details the inference speed required to process a standardized high-resolution satellite dataset. Converting the model weights to the ONNX FP16 format caused massive performance gains when ran with appropriate hardware. The NVIDIA RTX 4060 GPU, utilizing full FP16 hardware acceleration via Tensor Cores, processed the imagery at a rate of 0.31 seconds per square kilometer. In CPU execution, the Intel i9-13900HX CPU relying on software emulation for FP16 math processed the same area at 4.39 seconds per square kilometer. These results confirm that while the application successfully maintains adaptability according to different hardware platforms (allowing it to run on standard CPUs if necessary), utilizing a dedicated GPU increases processing speed by a factor of 14x, fully meeting the requirements for running on consumer end hardware.

Table 4 Inference Speed Comparison (Approximate)

Hardware	Model	Precision	Total Time (s)	Time/km ²	Notes
GPU	YOLO26s	FP16	70	0.31	NVIDIA RTX 4060
CPU	YOLO26s	FP16	999	4.39	Intel i9-13900HX

[Figure 3] shows the F1-score of our model. [Figure 4] shows the mean average precision@50 of our model. [Figure 5] shows the precision, recall, mAP@50 and mAP@50-90. [Figure 6] shows the different loss metrics of our model. [Figure 7] shows the confusion matrix.

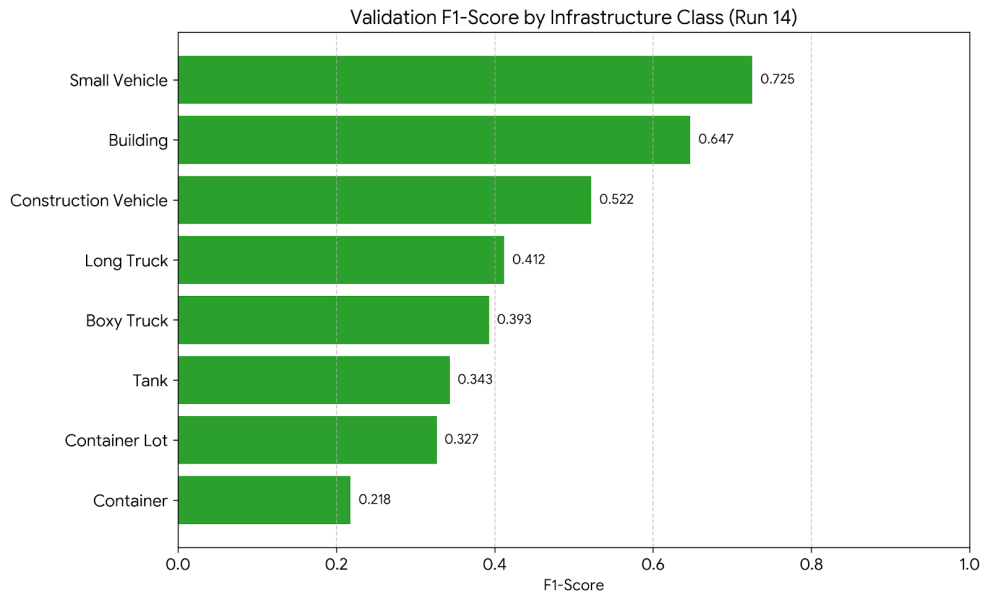


Fig. 3 F1- Score

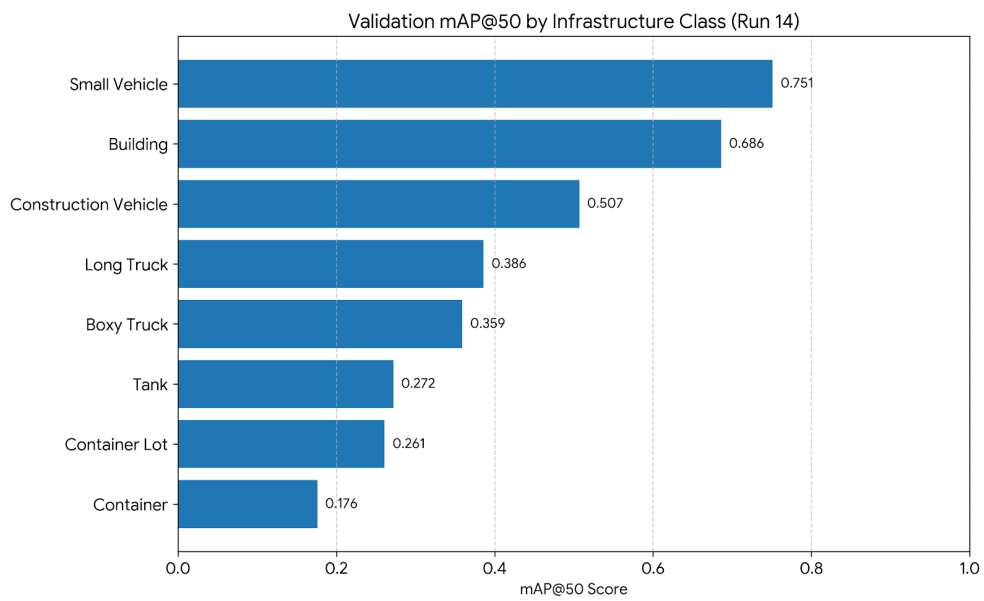


Fig. 4 mAP@50

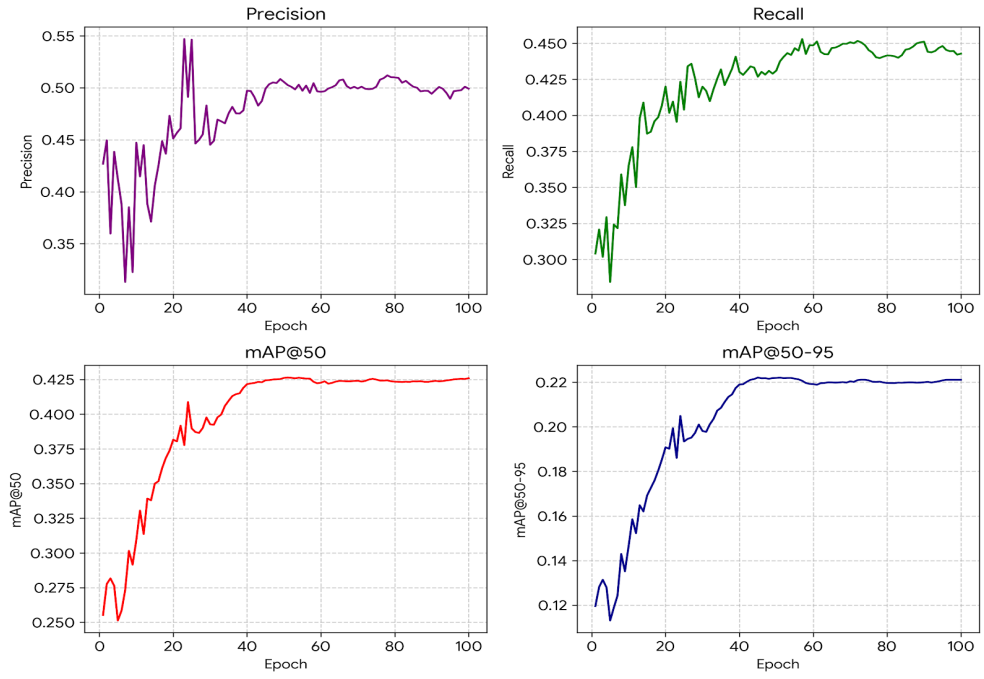


Fig. 5 Performance Metrics

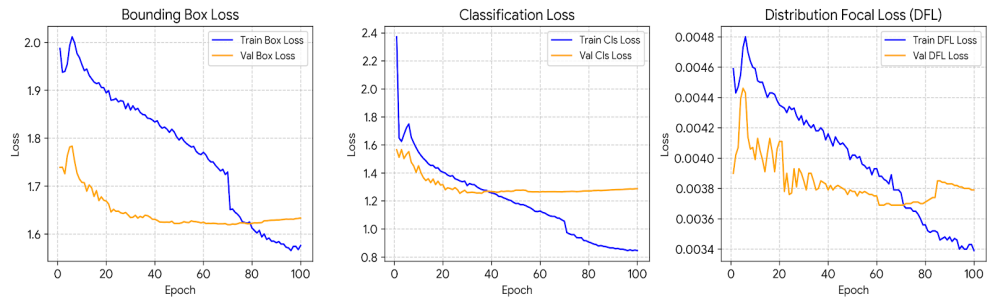


Fig. 6 Loss

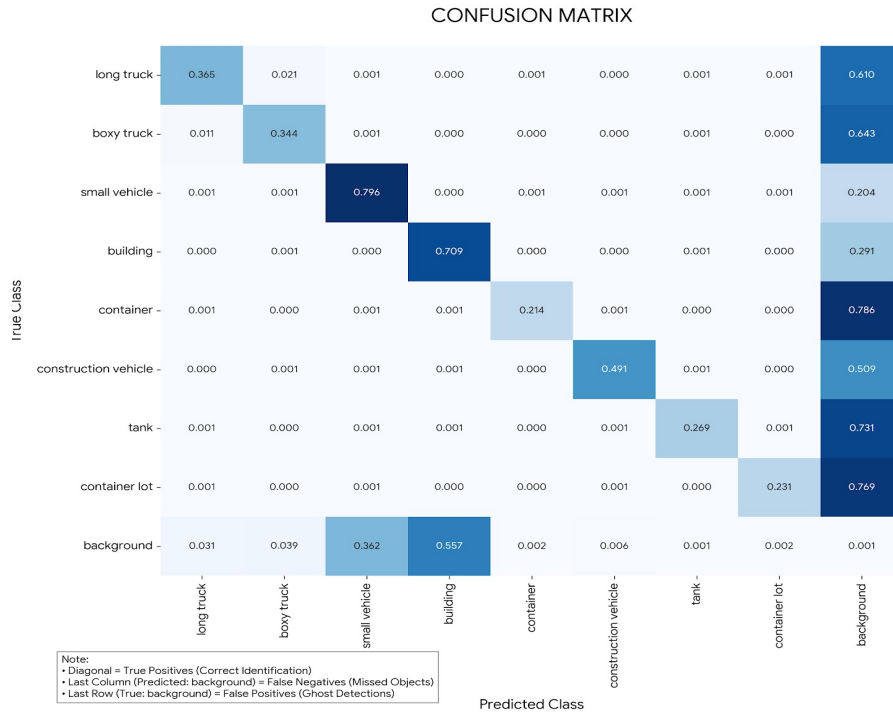


Fig. 7 Confusion Matrix

4.3 Example Images

Below are some examples of images which are analyzed by our application. Both before and after images are added one after another and we can clearly see our application is successfully detecting objects both in dense and sparse areas. It is also detecting very small object like cars which are 12 by 8 pixels in size in 4k images. Also it is successfully classifying between cars, trucks, building, water tanks and other vehicles. It has very few false positives and false negatives. It is working in different terrains such urban areas, forests, barren lands and deserts which proves that it can perform and give excellent results in hardware constrained devices which saves costs.



Fig. 8 EXAMPLE 1 - BEFORE

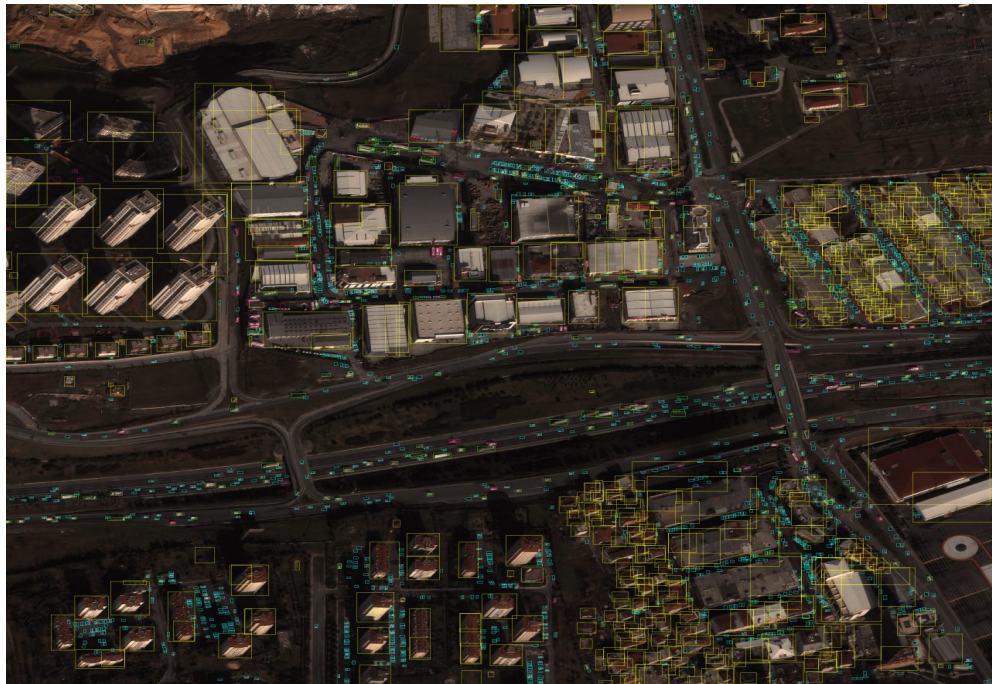


Fig. 9 EXAMPLE 1 - AFTER



Fig. 10 EXAMPLE 2 - BEFORE



Fig. 11 EXAMPLE 2 - AFTER



Fig. 12 EXAMPLE 3 - BEFORE

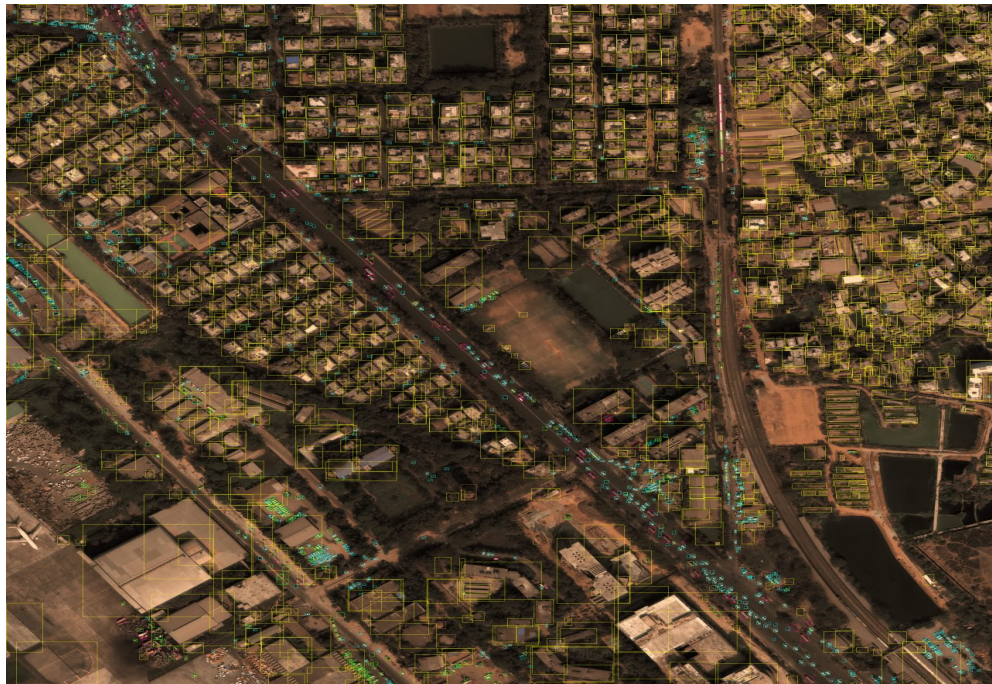


Fig. 13 EXAMPLE 3 - AFTER

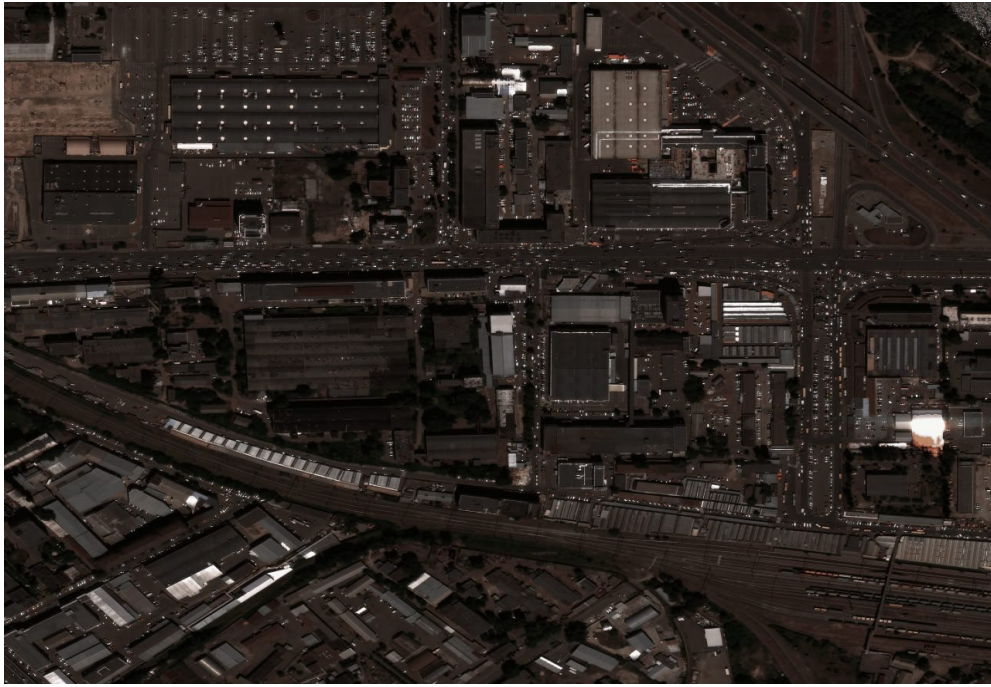


Fig. 14 EXAMPLE 4 - BEFORE

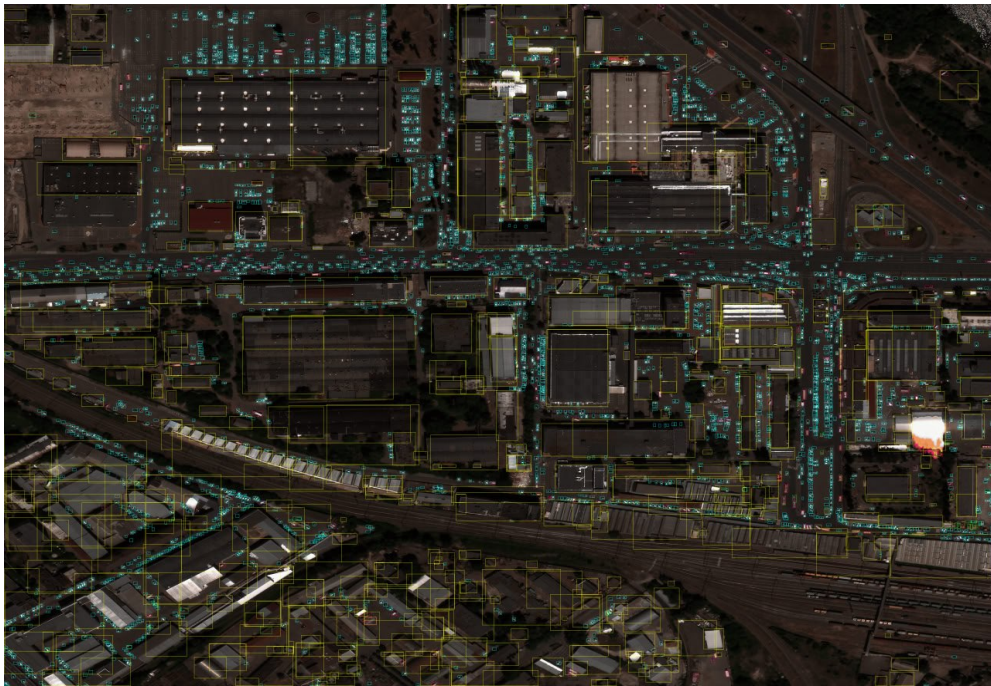


Fig. 15 EXAMPLE 4 - AFTER

5 Conclusion and Future Scope

5.1 Conclusion

The development of the fully offline high quality satellite image analyser solves the problems of the need of high end server infrastructure. It also saves a lot of time as it is automated. This naturally eliminates the issue of any human error. The usage of the state-of-art YOLO26s model, trained on the massive xView dataset that we prepared, has led to the development of a very robust mechanism. It can easily identify a variety of urban assets like buildings, transportation networks, infrastructures, etc. This is all done directly from the high resolution GeoTIFF images.

Our project shows that a dual language architecture can easily avoid any bottleneck situations associated with large scale data pre-processing of high resolution satellite images. The dual language mechanisms used here are Rust and Python. Rust ensures memory safety and simultaneous working. On the other hand Python provides its excellent deep learning capabilities. We have furthermore implemented Virtual Tiling and an optimized custom spatial Grid NMS algorithm. These two processes enable proper processing of the very large high resolution images without overlapping or duplicity and memory failures.

Along with the above mentioned processes we have also used migration to FP16. This improves the performance by a lot as the throughput on local hardware is almost doubled. The raw pixelated data that is detected is converted into real world latitude and longitude coordinates. This bridges the gap between raw data and urban intelligence. The transformation from raw pixelated data to structured metadata is a huge step for urban planners to move from data acquisition to informed infrastructure in a matter of a few seconds. All of this is done while maintaining total data privacy and in a completely offline environment.

5.2 Future Scope

The system that has been developed gives a really strong and reliable foundation for Urban monitoring. Still, there are several other enhancements that can be made in future to extend the capabilities of this system. Few of those has been mentioned below.

- **Temporal Change Detection:**

An algorithm for detection of change can be included. This would be useful to compare satellite images over a period of time. This system would also incorporate the use of automatic detection of new urban assets in a map, like new constructions of buildings or roads. It can also be essential to track illegal land usage. Urban expansion can be tracked as well using this system.

- **Multispectral and SAR Integration:**

The usage of Synthetic Aperture Radar (SAR) and multispectral imagery will allow monitoring which are not visible to normal cameras or objects which are hidden. It is because it would become usable under any kind of weather and also during night time. This would be very helpful in giving reliable data.

- **Edge Deployment and Real-Time Feeds:**

In future various optimizations can be performed on this system to make it deployable on Low-power edge devices. It can also be optimized to be integrated with real-time drone feeds. This would enable rapid situational awareness during a state of emergency or large-scale events.

- **Automated Reporting and Analytics:**

This is a different type of future development where automated reporting tools can be integrated to the already existing system. This would generate structured PDF reports, density heat maps, etc. extremely fast. This would be done directly from the PostGIS database and would thus increase the efficiency of decision-making.

References

- [1] Van Etten, A.: You only look twice: Rapid multi-scale object detection in satellite imagery. arXiv preprint arXiv:1805.09512 (2018)
- [2] Albert, A., Kaur, J., Gonzalez, M.C.: Using convolutional networks and satellite imagery to identify patterns in urban environments at a large scale. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1357–1366 (2017)
- [3] Taubenböck, H., Esch, T., Roth, A.: An urban classification approach based on an object-oriented analysis of high resolution satellite imagery for a spatial structuring within urban areas. Proceedings on CD-ROM (2006)
- [4] Basheer, S., Wang, X., Farooque, A.A., Nawaz, R.A., Liu, K., Adekanmbi, T., Liu, S.: Comparison of land use land cover classifiers using different satellite imagery and machine learning techniques. Remote Sensing **14**(19), 4978 (2022)
- [5] Al-Ruzouq, R., Hamad, K., Shanableh, A., Khalil, M.: Infrastructure growth assessment of urban areas based on multi-temporal satellite images and linear features. Annals of GIS **23**(3), 183–201 (2017)
- [6] Weng, Q., Hu, X.: Medium spatial resolution satellite imagery for estimating and mapping urban impervious surfaces using lsma and ann. IEEE Transactions on Geoscience and Remote Sensing **46**(8), 2397–2406 (2008)
- [7] Zhang, C., Marzougui, A., Sankaran, S.: High-resolution satellite imagery applications in crop phenotyping: An overview. Computers and Electronics in Agriculture **175**, 105584 (2020)
- [8] Le, N.-T., Thai, N.-T., Bui, C.V.: Advancing urban development through vision-language models: applications and challenges of satellite imagery analysis. In: 2024 9th International Conference on Applying New Technology in Green Buildings (ATiGB), pp. 184–188 (2024). IEEE
- [9] Kavzoglu, T., Sen, Y.E., Cetin, M.: Mapping urban road infrastructure using

- remotely sensed images. *International Journal of Remote Sensing* **30**(7), 1759–1769 (2009)
- [10] Gobatti, L., Bach, P.M., Scheidegger, A., Leitão, J.P.: Using satellite imagery to investigate blue-green infrastructure establishment time for urban cooling. *Sustainable Cities and Society* **97**, 104768 (2023)
 - [11] Voigt, S., Kemper, T., Riedlinger, T., Kiefl, R., Scholte, K., Mehl, H.: Satellite image analysis for disaster and crisis-management support. *IEEE transactions on geoscience and remote sensing* **45**(6), 1520–1528 (2007)
 - [12] Verma, R.K., Kumari, S., Tiwary, R.: Application of remote sensing and gis technique for efficient urban planning in india. In: *Geomatrix Conference Proceedings* (2009)
 - [13] Sunar Erbek, F., Ulubay, A., Maktav, D., Yağiz, E.: The use of satellite image maps for urban planning in turkey. *International Journal of Remote Sensing* **26**(4), 775–784 (2005)
 - [14] Babbar, J., Rathee, N.: Satellite image analysis: A review. In: *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, pp. 1–6 (2019). IEEE
 - [15] Said, N., Ahmad, K., Riegler, M., Pogorelov, K., Hassan, L., Ahmad, N., Conci, N.: Natural disasters detection in social media and satellite imagery: a survey. *Multimedia Tools and Applications* **78**(22), 31267–31302 (2019)
 - [16] Nasr, T., Ghalehtemouri, K.J., Khedmatzadeh, A., Mousavi, M.N., Rajabi, A.: Predicting urban green infrastructures of ecosystem services in tehran metropolitan area sprawl with landsat satellite time series data. *Journal of Architectural/planning Research and Studies (JARS)* **22**(1), 268554–268554 (2025)