GeoAI and Deep Learning
Student Projects
Winter Term 2024/25

Heidelberg University

Project Titles:

- Enhancing Participatory Mapping through AI: Detecting Hand-drawn Markings Using Siamese YOLOv9e
- Benchmarking Deep Learning Models for Road Surface Classification on StreetSurfaceVis
- PV Panel Detection from High-Resolution Aerial Imagery in Heidelberg using Deep Learning
- Building Detection Using Satellite Imagery in South India

Course Lecturer:
Steffen Knoblauch
Steffen.Knoblauch@uni-heidelberg.de
Heidelberg University

Enhancing Participatory Mapping through AI: Detecting Hand-drawn Markings Using Siamese YOLOv9e

Clemens Langer
Celina Thomé

ABSTRACT

Participatory Mapping empowers communities to contribute localized spatial knowledge vital for urban planning, disaster preparedness, and environmental risk assessment. These valuable inputs are often captured in analogue formats—such as Sketch Maps—to bridge the digital gap and include local population. However, these analogue maps pose significant challenges for digital interpretation due to visual variability, scanning artefacts, and complex backgrounds. The Sketch Map Tool (SMT) addresses this through a multi-stage deep learning pipeline that extracts annotations from scanned maps. We enhance the SMT by replacing its object detection module with a Siamese YOLOv9e architecture. Our dual-input approach processes both clean and annotated versions of the same map, using feature-level fusion to isolate user-added content. Trained on a large-scale dataset of synthetic and real-world Sketch Maps, our approach improves recall, precision, and mean average precision. Experiments across OpenStreetMap and satellite imagery basemaps demonstrate improved robustness and generalization. This focused upgrade makes the SMT pipeline more scalable for automated Participatory Mapping, while keeping it easy to understand and practical to use in real-world field settings. This ensures communities can meaningfully contribute to spatial planning through inclusive, data-driven insights.

KEYWORDS

Participatory Mapping, Change Detection, GeoAI, Siamese Neural Network, YOLO, Sketch Maps

1 INTRODUCTION

Participatory Mapping (PM) enables local communities to contribute spatial insights rooted in everyday local knowledge—insights that are often overlooked by top-down data collection methods [17]. In contexts ranging from urban development to climate adaptation and disaster risk reduction, these inputs are invaluable for identifying hazards, infrastructure needs, and social vulnerabilities. A common approach involves community members drawing directly on printed basemaps—creating so-called *Sketch Maps* (SMs), to highlight relevant features. Although rich in qualitative data, annotated maps remain a technical challenge to digitize as geographical data due to visual variability and lack of standardization [3]. Hand annotations vary widely in style, color, and scale. Maps are scanned or photographed under inconsistent conditions. Existing methods for digitization often involve manual tracing, thresholding, or basic image differencing, which are labor-intensive and brittle. As Elwood and Cope emphasize, the shift toward digital PM raises questions of legibility and power—whose knowledge gets formalized and how faithfully it is represented [26].

The Sketch Map Tool (SMT) is an open-source platform that automates the extraction of community-drawn annotations from scanned or photographed maps using a deep learning-based pipeline. This system integrates object detection, segmentation, and color classification to identify and digitize user-added content. However, existing implementations analyze only the annotated image in isolation, limiting their ability to distinguish markings from static map features—particularly in visually complex or low-contrast areas. To address this, we propose a key enhancement: a Siamese YOLOv9e architecture that simultaneously processes both the clean (unmarked) and annotated versions of each map. By performing feature-level comparison across the two inputs, the model can more accurately detect new annotations while suppressing background content. These detections are then refined through instance segmentation and semantic color classification, enabling more precise and reliable digitization.

Figure 1: Community sketch mapping in Colombia for disaster and flood risk assessment during the Red Cross's EVCA project using SMT [8].

Our contributions are threefold: (1) the introduction of a Siamese detection architecture specifically adapted for participatory SMs, (2) a scalable and modular deep learning pipeline suited to diverse basemap and annotation styles, and (3) a synthetic data generation strategy that enables robust training with high variability. Collectively, these contributions aim to enhance detection performance, particularly in complex scenes with irregular or ambiguous annotations, by improving recall, precision, and overall robustness.

By integrating advanced change detection into a fully automated participatory mapping system, this work bridges the gap between analogue, community-driven knowledge, and structured digital geospatial data. The approach enables more inclusive, responsive spatial planning, particularly in humanitarian and low-resource contexts.

2 BACKGROUND AND RELATED WORK

2.1 Participatory Mapping and Applications

Participatory Mapping complements traditional data production by addressing varying spatial coverage and occasional misalignment with local realities in official datasets [17, 28]. Grass roots communities are enabled and empowered to co-create spatial knowledge, enhancing local planning, disaster risk reduction, and environmental management, including their perspectives and reflecting lived experiences which are often not captured in official datasets [5, 15]. In particular in its analogue form, PM empowers communities without technical expertise to highlight their issues, making their realities visible, and thus fostering inclusive, sustainable decision making grounded in local knowledge [14, 17]. Hand-drawn SMs remain one of the most accessible and widely used formats for capturing such knowledge. However, their analogue nature, variability, and informal representation make them difficult to process with standard GIS tools. The transformation of the mapping results into digital, appealing visualizations is necessary to fully realize the potential of PM. Not only does this support advocacy of community's needs among broader audiences and decision-makers, but it is also a powerful way to return the results to the community itself—fostering awareness, dialogue, and empowerment [7]. As Cochrane and Corbett note, PM serves both to visualize the connection between people and place and to support broader social change, while also carrying assumptions and limitations that must be critically evaluated [6].

With the advancement of geospatial artificial intelligence (GeoAI), new opportunities have emerged to automate, scale, and enrich PM practices. GeoAI integrates spatial data with machine learning and computer vision techniques to extract insights from maps, satellite imagery, and other location-based inputs. In participatory contexts, GeoAI methods have been applied to identify patterns in hand-drawn annotations, quantify community-defined hazards, and integrate crowd-sourced spatial narratives into digital platforms. This computational enhancement allows PM to contribute not only localized perspectives but also analytically robust, spatially explicit datasets to inform policy and planning.

Recent practical implementations reflect this trend. The SmartLandMaps project [19] demonstrates the potential of automated sketch interpretation, using computer vision and vectorization to extract parcel boundaries from hand-drawn land tenure maps. Paper2GIS [11] automatically extracts and georeferences hand-drawn annotations from scanned paper maps, converting them into GIS-compatible vector data. Such tools illustrate how GeoAI can operationalize participatory mapping outcomes, supporting spatial analysis and data-driven decision-making.

2.2 Sketch Map Tool and AI pipeline

The SMT is an open-source web application developed by the Heidelberg Institute for Geoinformation Technology (HeiGIT) to support PM in humanitarian, environmental, and civic planning contexts [14]. It bridges analogue practices—such as hand-drawn community maps—with modern geospatial analysis by enabling users to annotate printed base maps derived from OpenStreetMap (OSM) or ESRI World Imagery (EWI) using pens or markers. These maps are then scanned or photographed and uploaded to the platform, which attempts to extract and georeference hand-drawn markings as GIS-compatible vector data (Figure 2).

SMT offers two basemap choices: OSM and EWI. While OSM is widely used, its coverage varies significantly, especially in rural or underserved regions [1, 13]. In such cases, EWI provides a more consistent high-resolution alternative. This flexibility supports a wide range of deployment environments—from post-disaster planning in Timor-Leste to flood risk assessments in Germany [15, 23].

Early versions of SMT relied on image differencing and thresholding to detect user annotations. This approach computed pixel-wise differences between the uploaded and original map images, followed by binarization to isolate changes. While conceptually simple, it was highly sensitive to scanning artifacts, lighting conditions, and small misalignments—leading to frequent false positives and negatives. In response, version 2.0 introduced a multi-stage deep learning pipeline (Figure 2) integrating object detection and segmentation components to enhance robustness, accuracy, and generalizability across diverse mapping contexts. The current pipeline comprises the following components:

(1) **Object Detection (YOLOv8)**: Custom-trained models for OSM and EWI basemaps detect bounding boxes of annotated regions using a 4-channel input (RGB + difference).
(2) **Instance Segmentation (SAM2)**: Detected regions are passed to the Segment Anything Model (SAM2) in zero-shot mode to create precise masks.
(3) **Color Classification (YOLOv8)**: Cropped segments are classified into semantic categories based on marker color using a lightweight CNN.
(4) **Postprocessing and Vectorization**: Morphological operations are applied to clean the masks, which are then exported as GIS-compatible vector formats such as GeoJSON.

The training dataset for SMTv2 includes approximately 500 annotated maps per basemap type. These were printed, drawn upon, scanned, and labeled. SMTv2 uses a 4-channel input (RGB + greyscale pixel-wise difference map) to improve change localization; however, the model still treats this composite as a single image. It does not perform feature-level comparisons between the clean and annotated maps, limiting its ability to differentiate subtle or ambiguous markings, while increasing the number of trainable parameters. This motivates the enhanced Siamese architecture introduced in this paper, which explicitly compares the two images in parallel and learns changes through deep feature-level subtraction.

2.3 Object Detection with Siamese Networks and YOLO

Object detection is the task of identifying and localizing objects within images, typically using bounding boxes and class labels. Traditional two-stage approaches such as Faster R-CNN [25] separate region proposal from classification and offer high accuracy but slower inference. In contrast, single-stage detectors like YOLO (You Only Look Once) [24] predict object locations and classes in a single forward pass, offering a strong trade-off between speed and accuracy. The YOLO family has progressively improved through deeper backbones, cross-stage partial connections, and multi-scale heads, making them suitable for real-time applications.

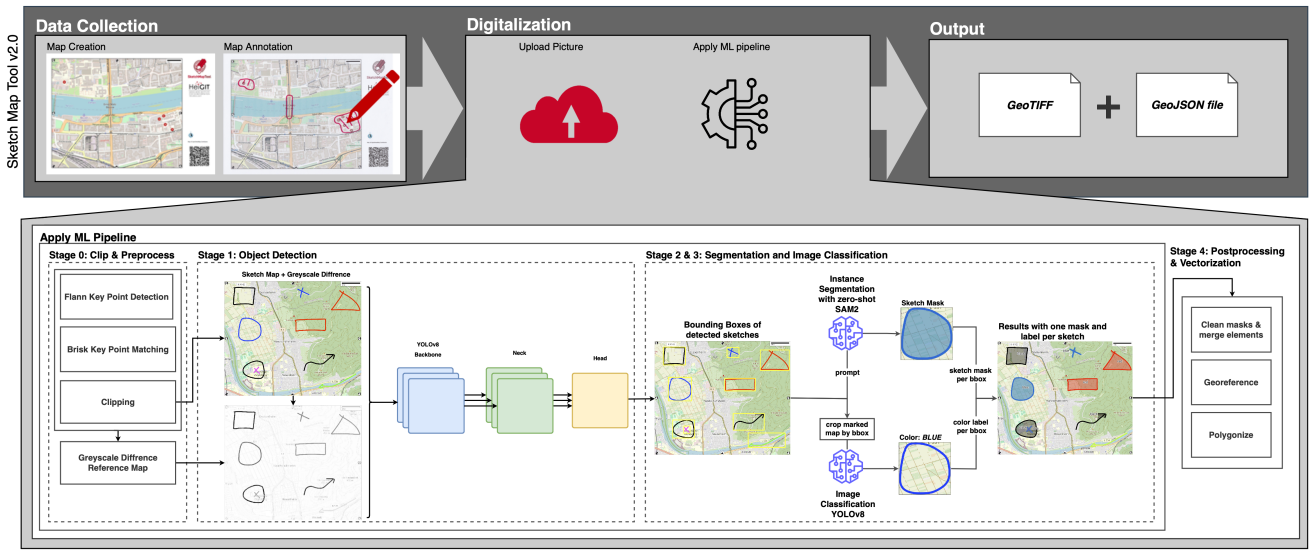


Figure 2: Overview of the AI pipeline in Sketch Map Tool v2 for object detection and segmentation.

In PM, where annotations are often faint and irregular, single-image YOLO detection struggles with subtle visual cues. We address this by shifting to a change detection approach using a Siamese network that compares annotated and unmarked map versions, enhancing sensitivity to participant-drawn changes. A Siamese Network is a neural network architecture consisting of two identical subnetworks whose weights are shared. These subnetworks operate in parallel, each receiving separate input, and their outputs are compared or combined to determine the similarity or difference between inputs [16]. Siamese networks are particularly effective in learning discriminative feature embeddings, making them suitable for tasks involving pairwise comparisons, such as verification, recognition, or change detection. They have found widespread application across various domains, including facial recognition, object tracking, signature verification, and medical imaging analysis [2, 4]. In remote sensing and geospatial analysis, Siamese networks have proven particularly beneficial for change detection due to their ability to directly learn from bi-temporal imagery [10]. Their structure allows them to effectively capture subtle differences while minimizing interference from irrelevant variations such as lighting, viewpoint, and seasonal changes.

While Siamese structures have been extensively studied, their integration with YOLO is relatively novel. Recent research has explored Siamese YOLO or Siamese-attention integrated YOLO architectures for tracking and detecting changes in scenes or specific objects efficiently in real-time [30, 31]. These models use shared YOLO backbones to extract features from two input images, improving change detection while keeping parameter counts similar to non-Siamese models. Recent work has enhanced feature fusion through a Structure Coefficient Block and attention mechanisms [30, 31]. Building on these advances, we adapt a Siamese YOLOv9e model for participatory mapping, using bi-temporal inputs to detect annotations with greater sensitivity and robustness across diverse basemap styles.

3 METHODOLODY AND DATASET

3.1 Proposed Siamese Detection Pipeline

Sketch Map Tool 2.1 introduces a novel architectural approach by reframing the task of SM interpretation as a bi-temporal change detection problem. Rather than analyzing annotated maps in isolation, the model concurrently processes both the original (clean) and the annotated versions of a map to isolate user-added content. To achieve this, we develop a custom Siamese YOLOv9e pipeline [18] that performs parallel feature extraction on both images (Figure 3). This Siamese configuration enables the network to reason over differences between the two images. Intermediate feature maps are fused using element-wise subtraction or alternative attention mechanisms to emphasize the annotations while suppressing static background content. This enhances the model's ability to localize complex, irregular, or faint markings while maintaining precision on small, well-defined features. Our Siamese model extends the Ultralytics YOLOv9 implementation by introducing two identical backbones with shared weights. To stabilize training, one branch is temporarily *frozen* to encode the background (clean map), while the second remains fully trainable. Synchronization is maintained by updating the frozen branch every 10 batches with the current weights from the trainable branch. To integrate the clean and annotated streams, we compute element-wise differences between their backbone feature maps at three FPN levels—P3, P4, and P5, corresponding to strides of 8, 16, and 32. These difference-based feature maps effectively suppress static basemap content and emphasize user-drawn markings. Finally, the fused output is fed into the neck's and supply the small, medium, and large detection heads with change-focused representations. While we experimented with

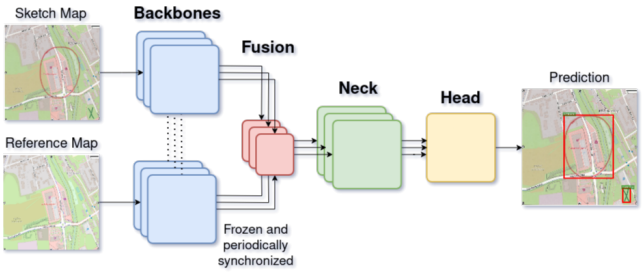


Figure 3: Simplified structure of the proposed Siamese YOLOv9e Model. Highlighting the Dual Backbone structure in comparison to the single backbone in figure 2.

attention-based alternatives such as CBAM and cross-attention, we prioritized the use of simple difference operations due to their lower computational cost and reliable performance.

To ensure consistency between the two input streams during training, we apply synchronized augmentations using the Albumentations library. These include geometric transformations (e.g., flips, scaling, perspective changes) and photometric adjustments (e.g., HSV shifts, brightness/contrast scaling, additive noise), preserving alignment while enhancing visual diversity.

By integrating a bi-temporal design into the YOLOv9e architecture, SMT version 2.1 achieves robust detection of diverse and subtle annotations. This improves generalizability across different basemaps, drawing styles, and real-world mapping scenarios.

3.2 Data

3.2.1 Synthetic Dataset. To pre-train on hand-drawn data and enable controlled experimentation, we generated a synthetic dataset of 18,400 SMs, equally distributed between EWI satellite and OSM basemaps. Sampling followed a country-based train-test split using sampeling around Natural Earth settlement data [21], focusing on locations near human settlements. This geographic split—stratified by country into 80% train, 10% validation, and 5% test—aims to evaluate generalization to unseen regions. OSM's uniform render style likely induces less variability than the more heterogeneous satellite imagery from EWI. Minor deviations from the intended proportions resulted from processing errors and geographic imbalances.

We prioritized settlement-based sampling due to expected application in inhabited areas, but ensured a wide sampling radius to include surrounding natural zones.

Each synthetic SM featured simulated annotations from two base types: manually drawn geometric shapes and algorithmically generated blobs. Masks were created using the `hand-drawn-shapes` dataset [22] and the `blob-masks` Python package. Morphological operations, gradient variations, and noise filters were applied to mimic the appearance of marker strokes. Masks were then composited and blended into the map backgrounds, with bounding boxes stored for training. To simulate photographic distortions found in real SMs, we applied Albumentations-based augmentations including lens distortion, perspective warping, motion blur, additive noise, brightness/color shifts, and shadow overlays. This augmentation strategy helped reduce the domain gap between synthetic and real-world inputs, improving model robustness.

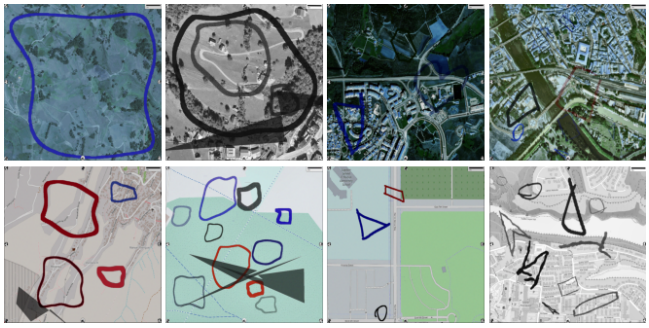


Figure 4: Synthetic dataset samples: EWI (top) and OSM-based (bottom) showing blob-type (left two columns) and altered hand-drawn markings (right two columns).

3.2.2 Hand-drawn Dataset. To create a diverse training dataset, 1,100 randomly generated SMs per background were printed (ISO A4, color and B/W) and distributed to eight coworkers and student assistants for annotation. A variety of markers with differing colors and thicknesses were provided to simulate the heterogeneity of real-world user annotations. The SMs were sampled in a similar manner as the synthetically generated maps around the settlements, but without applying the geographic criteria. Based on user requirements and feedback, e.g. that very large or unusually shaped polygons are often not detected, we deliberately incorporated these challenging edge cases into our training set by giving instructions to the annotators to include exactly these. By assessing and addressing the specific scenarios users found problematic, we tried to ensure that our model generalizes robustly to complex, real-world sketch-map annotations.

The completed SMs were subsequently digitized by scanning or photographing them. This deliberate diversity in participants and drawing materials was essential to ensure that the dataset captured a broad range of annotation styles and visual characteristics, thereby improving the generalizability and robustness of the trained model.

All digitized images were manually labeled using the Computer Vision Annotation Tool (CVAT) [9], following a consistent set of annotation guidelines across the annotators to ensure labeling uniformity and produce high-quality ground truth data for training and evaluation. In a total of 3 annotators (the two authors and one student assistant) labeled the images in CVAT. After that a fourth person reviewed each set of labels before approving it.

From the previous version, we had additional training data that we merged into the training data we already have. This leads to our datasets having the following counts for each split (Table 1).

3.3 Training Setup and Evaluation

In a first step we trained a model on the synthetic dataset. The resulting model we have used as an initial checkpoint to train two separate models, one for EWI sattelite imagery, one for OSM imagery SMs. As we expect major domain shifts for the real world data between EWI and OSM, we have decided to separatly train two individual models, for each background type. To benchmark improvements, we compare the Siamese model against a standard YOLOv9e baseline trained only on annotated images, in a similar manner as

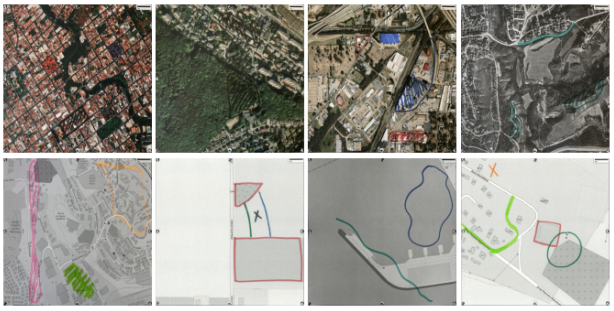


Figure 5: Realworld dataset samples: EWI satellite imagery (top) and OSM imagery (bottom).

Table 1: Counts of unique Sketch Maps and instances by dataset and split.

Dataset	Split	Sketch Maps	Instances
Synthetic	Train	15,795	104,548
	Val	1,666	10,869
	Test	932	6,564
OpenStreetMap	Train	1,135	6,138
	Val	132	773
	Test	138	668
ESRI World Imagery	Train	1,201	6,431
	Val	210	1,139
	Test	140	662

described above. This single-stream setup allows us to isolate the impact of bi-temporal reasoning on detection performance. For the baseline model, we additionally decided to train two models, which have not been pretrained on synthetic data prior, rather relying on the default checkpoint provided by Ultralytics. For the Siamese YOLOv9e this was no option, as no such checkpoint is available. During the training we continuously evaluated each epoch utilizing our validation dataset to prevent overfitting. The final evaluation is based on the test split of each dataset. We evaluated performance using four standard metrics: Precision (TP/(TP + FP)) measures the fraction of predicted positives that are correct; Recall (TP/(TP+FN)) measures the fraction of actual positives detected; mAP_{50} is the mean average precision at an IoU threshold of 0.5; andmAP_{95-50} is the mean average precision averaged over IoU thresholds from 0.5 to 0.95 in 0.05 increments. We adopted a near-default Ultralytics YOLOv9 training configuration with only minimal modifications. We increased the input resolution from the default 640 px to 1024 × 1024 px to preserve detail in fine annotations and selected a batch size of seven, fully leveraging available VRAM for balanced gradient stability and speed. Training closely followed YOLOv9 defaults: 200 epochs total, initial learning rate (lr0) of 0.01 decreasing to a final factor (lrf) of 0.01, momentum at 0.937, weight decay at 0.0005, and a 3-epoch warm-up. Early stopping was adjusted to trigger after 20 epochs without improvement, ensuring robust and stable convergence across synthetic and real-world datasets. Data augmentation utilized the Albumentations library, applying synchronized geometric (random resized cropping, flips, 90° rotations, perspective shifts) and photometric transformations (HSV adjustments, brightness/contrast scaling, CLAHE) to both timestamps. Environmental augmentations (lens distortion, rain, sun flare, shadows, grayscale, Gaussian noise) were applied exclusively to the marked timestamp to simulate realistic variability. Each model underwent a two-phase training: first 100 epochs with augmented data, followed by 100 epochs on original, non-augmented images.

4 RESULTS

In the synthetic test split, both the baseline YOLOv9e (pre-trained on synthetic data) and the Siamese variant deliver comparable detection quality. The Siamese design yields consistent, modest gains in recall and localization metrics, indicating that dual-stream comparison refines marking detection under controlled annotation variability.

Figure 6 shows the training history on the synthetic dataset. Loss components decrease smoothly on a log scale, the learning rate schedule follows the expected warm-up and decay, and validation metrics steadily increase without overfitting.

On OSM backgrounds, synthetic pretraining produces a clear lift over the direct-training baseline. The Siamese YOLOv9e further enhances boundary tightness and reduces missed annotations, demonstrating that even against relatively uniform backgrounds, dual-stream feature fusion can sharpen detection performance.

For satellite scenes, the baseline trained directly on EWI outperforms its synthetic-pretrained counterpart, suggesting that synthetic-to-satellite transfer is imperfect. Incorporating the Siamese architecture recovers this gap and exceeds both baselines, highlighting that explicit change focusing against the unmarked basemap improves robustness to textured, variable backgrounds.

In comparison of the training progress of the models trained in our real world dataset in Figure 6 to the development of the synthetic data, we can observe that it takes more epochs to converge. In addition,, we can see that the metricscoverer spectrum in comparison, with a trend to lower mAP_{95-50}.

To visualize how the Siamese fusion block emphasizes markings, we extracted class activation maps (CAMs) from layers 30 and 60 (P3) and from the fused output at layer 61, and plotted these for sample images in Figure 7. The fused CAM clearly exhibits a wider dynamic range—incorporating negative activations (shown in blue) that suppress static background features—while positive peaks correspond precisely to hand-drawn markings. This demonstrates that the fusion block effectively isolates user-added annotations from the underlying map content.

Figure 8 illustrates our most common real-world failure modes. Both networks rarely confuse prominent map features with hand-drawn markings, for example, on OSM tiles they detect a stadium footprint (b) and map icons (a) as annotations, and on EWI satellite imagery they even pick up a larger coastal area in a very prominent blue (e). Interior polygons within layered markings are occasionally missed or markings (d), touching each other can be merged on OSM backgrounds (c, g). We also observe that the model sometimes omits clearly marked sketches (h).

Table 2: Precision (p), Recall (r), mAP_{50} and mAP_{95-50} for models trained and evaluated on synthetic, EWI, and OSM datasets.

Model	Synthetic				ESRI World Imagery				OpenStreetMap			
	p	r	mAP_{50}	mAP_{95-50}	p	r	mAP_{50}	mAP_{95-50}	p	r	mAP_{50}	mAP_{95-50}
baseline YOLOv9e without training on sythnetical data	-	-	-	-	94.6	87.8	90.6	73.0	95.5	95.5	93.2	80.4
baseline YOLOv9e	99.1	95.7	97.2	95.7	91.2	86.5	91.5	76.2	98.0	96.2	97.3	81.7
Siamese YOLOv9e	99.1	96.8	98.6	96.7	96.9	89.6	97.4	77.2	98.9	97.0	98.1	84.5

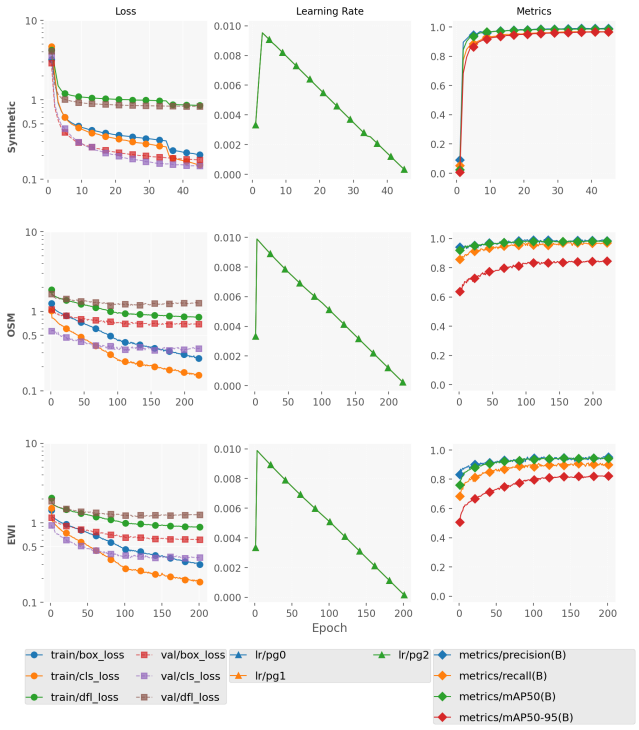


Figure 6: Training history of the Siamese YOLOv9e model on synthetic, OSM, and EWI datasets. From left to right, the plots show the evolution of loss functions, learning rate schedule, and validation metrics on the hand-drawn validation split.

5 DISCUSSION

5.1 Model performance and Limitations

Across all test splits — synthetic, EWI, and OSM — the Siamese YOLOv9e model consistently outperformed the single-stream baseline, yielding modest but reliable increases in precison, recall and localisation (see Table-2, without adding any trainable parameters or altering backbone capacity, as was the case for the SMT v2.0. These gains stem solely from our arithmetic-difference fusion blocks at P3–P5, which highlight user-added annotations by subtracting feature maps from the clean and marked inputs. This comparative focus sharpens change detection on both small and large markings, as well as irregular shapes, while preserving high precision on finer details. We deliberately included challenging edge cases—branched, highly irregular and oversized shapes —to stress-test robustness. Although detection quality drops when contours overlap extensively, the Siamese model still improves performance on clear, high-contrast markings. Informal user feedback, further revealed that even sticker based annotations, which the model never saw during training, are being detected.

The strong performance of the model on a geographically separated synthetic test set indicates an effective generalization across unseen regions. Hand-drawn markers exhibit largely invariant visual patterns—e.g., a red polygon remains red regardless of background—while the Siamese architecture explicitly suppresses static map content. Moreover, OSM's uniform rendering further simplifies detection compared to the textured variability of satellite imagery. Nonetheless, these results are only indicative: future work must employ continent- or land-cover-based splits to rigorously validate cross-domain robustness, and real-world annotations should be spatially stratified to uncover any hidden biases.

Despite these successes, our pipeline has limitations. The training data derived from a small group of annotators using a limited set of markers and scanning devices, which may not capture the full diversity of user-generated sketches worldwide. Consequently, performance on truly novel inputs might diverge from our reported metrics. Additionally, detection performance on EWI lags behind OSM by several percentage points: complex textures, lighting variations, and vegetation in high-resolution satellite scenes introduce background noise that even humans struggle to distinguish from markings. Pre-training the baseline YOLOv9e on our synthetic EWI dataset actually degraded performance versus the Ultralytics default checkpoint, highlighting a domain shift. Addressing this will require task specific synthetic datasets for each basemap type or leveraging related remote-sensing change, detection corpora to better align synthetic and real imagery.

Our qualitative analysis of false positives and false negatives shows that the model still struggles with complex, layered annotations. To mitigate this, we developed a set of annotation best practices covering marker materials (e.g., thick, high-visibility markers), clear polygon delineation, and appropriate basemap selection (e.g., considering texture and colour contrast for EWI). These guidelines are taught to practitioners during training to improve annotation consistency and ensure reliable model performance.

5.2 Potential Improvements and Future Works

We implemented consent-based data collection on the web platform to aggregate a broader variety of user sketches. Annotating and integrating these community-generated maps will inject new marker colours, pen types, chosen print formats, and scanning conditions, reducing dataset bias and enhancing generalization. More importantly, it will provide the SMT with the markings the users actually want to use, further tailoring it to our very specific user requirements. Future iterations of our model will include those real world use case. We also plan to include larger print formats in training, previously excluded for cost reasons. To handle associated

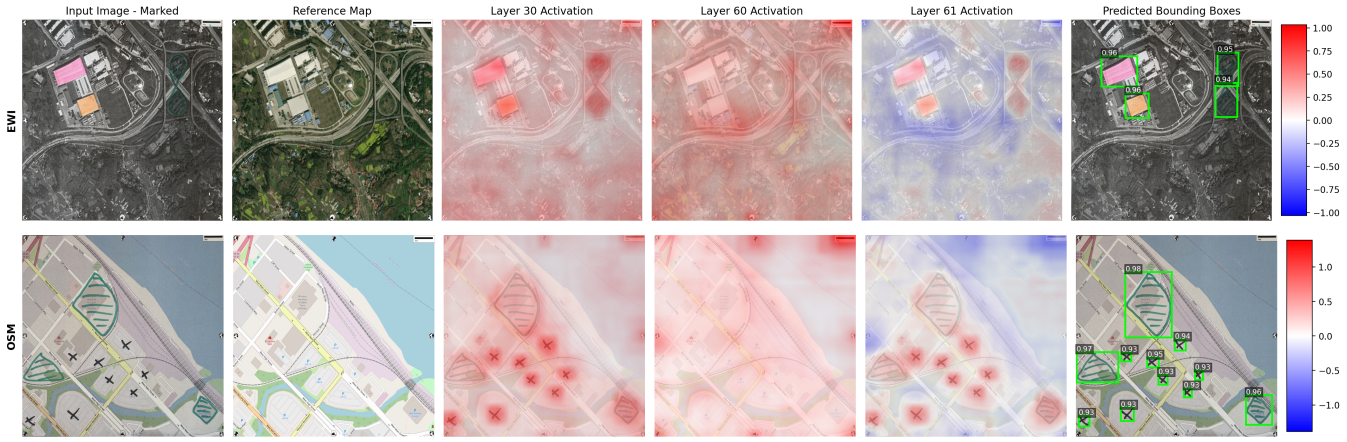


Figure 7: Class Activations Maps for the fusion of the P5-level features in layer 30 (marked), 60 (clean) and 61 (Fused).

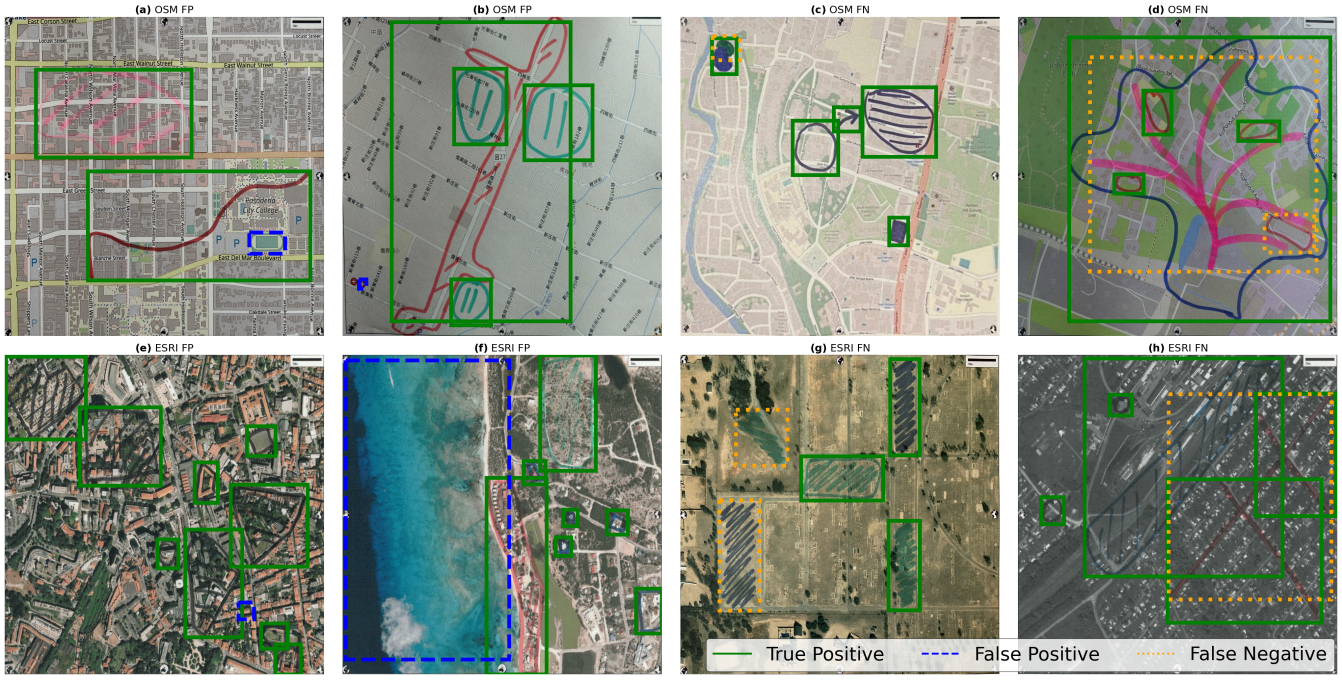


Figure 8: Explanatory common Errors by the proposed object detection for OSM (a-d) and EWI (e-h). Showing examples for False Positive (FP) and False Negative (FN) predictions.

distortions, we are testing sliced inference [29], which is currently showing promising improvements in recall and precision.

Our current arithmetic-difference fusion at three feature scales works well in practice, but findings of previously published work on Siamese YOLO suggests, that the models could benefit from a more complex feature fusion block, leveraging Attention mechanisms [31]. The current model setup already supports a variety of these, but for sake of simplicity and avoiding unnecessary energy consumption, we have focused for now on the arithmetic difference.

By freezing one of the backbones and synchronizing it periodically, we keep its feature output for the clean map fixed for a period. The trainable branch then only needs to learn what differs in the hand-annotated map. This "teacher–student" setup stabilizes training and focuses the model on real changes, similar to the Mean Teacher approach [27]. This configuration limits gradient computation to a backward pass through the student backbone, avoiding dual back propagation through a cloned backbone.

Incase of unfreeze both backbones, the network could potentially learn background and change features together more easily, but

training may become noisier or less stable and we lose the clear reference provided by a fixed teacher. A compromise could be to update the teacher backbone with an exponential moving average of the student's weights, blending stability with adaptability as in the original Mean Teacher approach [27].

Systematic hyperparameter searches, potentially via genetic algorithms implemented in the ultralytics package, can fine-tune learning rates, anchor sizes, and augmentation policies. So far we have mostly relied on tested default parameters provided by the Ultralytics packages. However, now that the architecture is implemented, finetuning the hyperparameters could yield marginal performance gains. Additionally, in a next step we also want to evaluate the base model selection and whether smaller YOLO models could provide similar performance with less computational requirements, as you could treat the selection of model depth as a hyperparameter itself. This would result in a more sustainable solution, consuming less energies and computational resources to maintain and run the model. Evaluating other modern detectors (e.g. DETR, Faster R-CNN) within a Siamese change-detection framework could identify complementary strengths. Although YOLOv9e excels in real-time settings, non-real-time applications may benefit from different architectures offering higher accuracy at the cost of speed. User feedback indicates that our model currently struggles to generate well-delineated masks for larger or less prominent markings—especially when a marking's shape yields a low intersection-over-union (IoU) with its bounding box, as is common for non-horizontal/-vertical linear features or complex shapes. Although our two-level pipeline (YOLO + SAM2) benefits from requiring only object-detection annotations rather than full instance-segmentation labels, zero-shot SAM2 exhibits clear limitations on these more intricate markings. Future research could involve the integration of a single-stage instance segmentation backend, such as Mask R-CNN, with the aim of enhancing the localization of overlapping or nested annotations. Alternatively, research could explore the application of a Siamese segmentation network, which combines our change-focused design with per-pixel delineation, with the objective of achieving improved recall and precision [12, 20]. Furthermore, the creation of a dedicated segmentation benchmark, as opposed to reliance on object detection metrics alone, would facilitate a more comprehensive evaluation of end-to-end performance. To reduce manual labeling effort, we plan to develop a human-in-the-loop pipeline that combines SAM2-based zero-shot mask proposals with selective manual correction. By iteratively refining pseudo-labels and retraining lightweight models, we aim to bootstrap a high-quality instance segmentation dataset with minimal resources.

6 CONCLUSION

In conclusion, the development of the Siamese YOLOv9e for multi-temporal SM annotation change detection advances efforts in bridging human-centric mapping practices with automated analysis. The approach proved that it can improve detection accuracy without added complexity, generalize to different use cases, and even handle inputs it was not specifically trained on. While challenges remain—such as expanding data breadth and addressing cross-platform map inconsistencies—the progress made here lays a solid foundation for future improvements. This work has broader relevance beyond just our test cases: it exemplifies how participatory mapping can be augmented by intelligent systems to achieve faster and more reliable updates. By empowering users with an AI tool that respects the simplicity of their sketch annotations and still delivers precise results, we move towards more responsive and inclusive mapping efforts. Ultimately, we believe that this research contributes to the toolbox for participatory mapping and change detection tasks, helping communities and organizations to leverage the possibilities of participatory mapping. However, potential distortions introduced during automatic digitization, especially considering a potential bias toward our own training data, underscore the need for validation on data from the mapping community itself.

Looking ahead, there are several promising directions to extend this work. A top priority is to collect and incorporate more diverse data from real mapping activities. This could involve collaborating with participatory mapping communities or incorporating the uploaded Sketch Maps into our Training loop. Such diversity will improve the model's robustness and make it truly deployment-ready for global applications, tailored to the needs of our users. Beyond these technical improvements, future iterations could also move past purely color-based annotations—for example, by recognizing handwritten text to enrich Sketch Maps with additional meaning and context.

ACKNOWLEDGMENTS

We gratefully acknowledge the support of the Klaus Tschira Stiftung (KTS) and the German Foreign Office, whose funding made this research possible. We also thank the German Red Cross (GRC) for their collaboration and practical insights throughout the project. Special thanks go to Dr. Carolin Klonner for guiding first prototypes and inspiring this work. We thank our colleagues and student assistants at HeiGIT for their crucial contributions to dataset creation, annotation, and technical implementation, which were instrumental to the success of this work. We thank Heidelberg University's Computing Centre for Accesses to the SDS@hd hot-data storage. Support by the state of Baden-Württemberg through bwHPC (Helix Cluster) and the German Research Foundation (DFG) through grant INST 35/1597-1 FUGG is greatfully acknowledged.

REFERENCES

[1] Christopher Barron, Pascal Neis, and Alexander Zipf. 2014. A Comprehensive Framework for Intrinsic OpenStreetMap Quality Analysis. *Transactions in GIS* 18 (12 2014). https://doi.org/10.1111/tgis.12073

[2] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. 2016. Fully-Convolutional Siamese Networks for Object Tracking. In *European Conference on Computer Vision*. Springer, 850–865. https://arxiv.org/abs/1606.09549

[3] E Eric Boschmann and Emily Cubbon. 2014. Sketch maps and qualitative GIS: Using cartographies of individual spatial narratives in geographic research. *The Professional Geographer* 66, 2 (2014), 236–248.

[4] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a "siamese" time delay neural network. *Advances in Neural Information Processing Systems* 6 (1993), 737–744. https://proceedings.neurips.cc/paper_files/paper/1993/file/288cc0ff022877bd3df94bc9360b9c5d-Paper.pdf

[5] James Bullen and Andrew Miles. 2024. Exploring local perspectives on flood risk: A participatory GIS approach for bridging the gap between modelled and perceived flood risk zones. *Applied Geography* 163 (Feb. 2024), 103176. https://doi.org/10.1016/j.apgeog.2023.103176

[6] Logan Cochrane and Jon Corbett. 2020. Participatory mapping. *Handbook of communication for development and social change* (2020), 705–713.

[7] Logan Cochrane, Jon Corbett, and Peter Keller. [n. d.]. *Impact of Community-based and Participatory Mapping*. Technical Report.

[8] Colombian Red Cross, German Red Cross, and HeiGIT gGmbH. 2023. Digitalizing Paper-Based Community Mapping in the EVCA. DRR in Action Case Study. Available via International Federation of Red Cross and Red Crescent Societies.

[9] CVAT.ai Corporation. 2022. *Computer Vision Annotation Tool (CVAT)*. https://doi.org/10.5281/zenodo.4009388 https://github.com/opencv/cvat.

[10] Rodrigo Caye Daudt, Bertr Le Saux, and Alexandre Boulch. 2018. Fully convolutional siamese networks for change detection. In *2018 IEEE International Conference on Image Processing (ICIP)*. IEEE, 4063–4067. https://ieeexplore.ieee.org/document/8451652

[11] Timna Denwood, Jonathan J Huck, and Sarah Lindley. 2023. Paper2GIS: improving accessibility without limiting analytical potential in Participatory Mapping. *Journal of Geographical Systems* 25, 1 (2023), 37–57.

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask R-CNN. In *ICCV*.

[13] Benjamin Herfort, Sven Lautenbach, João Porto de Albuquerque, Jennings Anderson, and Alexander Zipf. 2023. A spatio-temporal analysis investigating completeness and inequalities of global urban building data in OpenStreetMap. *Nature Communications* 14, 1 (July 2023), 3985. https://doi.org/10.1038/s41467-023-39698-6

[14] Carolin Klonner, Maximilian Hartmann, Rebecca Dischl, Lily Djami, Liana Anderson, Martin Raifer, Fernanda Lima-Silva, Lívia Castro Degrossi, Alexander Zipf, and Joao Porto de Albuquerque. 2021. The sketch map tool facilitates the assessment of OpenStreetMap data for participatory mapping. *ISPRS International Journal of Geo-Information* 10, 3 (2021), 130.

[15] Carolin Klonner, Tomás J. Usón, Nicole Aeschbach, and Bernhard Höfle. 2021. Participatory Mapping and Visualization of Local Knowledge: An Example from Eberbach, Germany. *International Journal of Disaster Risk Science* 12 (2021), 56–71. https://doi.org/10.1007/s13753-020-00312-8

[16] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese Neural Networks for One-shot Image Recognition. In *ICML Deep Learning Workshop*. https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf

[17] Melinda Laituri, Matthew W Luizza, Jamie D Hoover, and Arren Mendezona Allegretti. 2023. Questioning the practice of participation: Critical reflections on participatory mapping as a research tool. *Applied Geography* 152 (2023), 102900.

[18] Clemens Langer, Celina Thomé, Glenn Jocher, Jiayu Qiu, and Ayush Chaurasia. 2025. Ultralytics YOLO (siamese). https://doi.org/10.5281/zenodo.15496547

[19] Claudia Lindner, Auriol Degbelo, Gergely Vassányi, Kaspar Kundert, and Angela Schwering. 2023. The SmartLandMaps Approach for Participatory Land Rights Mapping. *Land* 12, 11 (2023), 2043.

[20] Claudio Michaelis, Ivan Ustyuzhaninov, Matthias Bethge, and Alexander S. Ecker. 2018. One-Shot Instance Segmentation. *CoRR* abs/1811.11507 (2018). arXiv:1811.11507 http://arxiv.org/abs/1811.11507

[21] Natural Earth. [n. d.]. Made with Natural Earth. Free vector and raster map data. https://www.naturalearthdata.com. Accessed: 2025-05-21.

[22] Frobert Pixto. 2024. hand-drawn-shapes-dataset. https://github.com/frobertpixto/hand-drawn-shapes-dataset/tree/main. Accessed: 2025-05-12.

[23] Red Cross Red Crescent Climate Centre Team for Anticipatory Action. 2023. Participatory Mapping in Timor Leste. Project Report. Available via Red Cross Red Crescent Climate Centre.

[24] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You Only Look Once: Unified, Real-Time Object Detection. *CVPR* (2016).

[25] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *NIPS*.

[26] Daniel Sui, Sarah Elwood, and Michael Goodchild. 2012. *Crowdsourcing geographic knowledge: volunteered geographic information (VGI) in theory and practice*. Springer Science & Business Media.

[27] Antti Tarvainen and Harri Valpola. 2017. Mean Teachers are Better Role Models: Weight-Averaged Consistency Targets Improve Semi-Supervised Deep Learning Results. In *Advances in Neural Information Processing Systems*, Vol. 30. 1195–1204. https://arxiv.org/abs/1703.01780

[28] Linnet Taylor and Dennis Broeders. 2015. In the name of Development: Power, profit and the datafication of the global South. *Geoforum* 64 (Aug. 2015), 229–237. https://doi.org/10.1016/j.geoforum.2015.07.002

[29] Ultralytics. 2025. *SAHI Tiled Inference Guide*. Ultralytics. https://docs.ultralytics.com/guides/sahi-tiled-inference/ Accessed: 25 April 2025.

[30] Feifan Yi, Haigang Zhang, Jinfeng Yang, Liming He, Ahmad Sufril Azlan Mohamed, and Shan Gao. 2024. YOLOv7-SiamFF: Industrial defect detection algorithm based on improved YOLOv7. *Computers and Electrical Engineering* 114 (2024), 109090. https://doi.org/10.1016/j.compeleceng.2024.109090

[31] Yi Zhang, Jie Pang, Baicheng Li, and Jianfeng Luo. 2023. Siamese YOLO V5 with Structure coefficient for object-level change detection. https://doi.org/10.21203/rs.3.rs-3806822/v1

Benchmarking Deep Learning Models for Road Surface Classification on StreetSurfaceVis

Runan Duan
Daniel Abanto

ABSTRACT

Accurate road surface classification is essential for enhancing travel time estimation in routing engines and supporting autonomous navigation systems. Recent advances in deep learning have enabled significant progress in visual surface recognition, yet comparative assessments of state-of-the-art architectures on standardized datasets remain limited. In this study, we conduct a systematic evaluation of three high-performing convolutional neural network architectures - ResNet-50, ConvNeXt-Small, and EfficientNet-B4 - on the StreetSurfaceVis benchmark dataset, which comprises diverse road surface types under varying lighting and environmental conditions. Despite overall high classification accuracy across models, performance disparities are observed across surface categories. ConvNeXt-Small achieves the highest class-wise performance, with an F1-score of 0.97 for paving stones, whereas concrete surfaces are consistently misclassified, yielding a maximum F1-score of 0.87. All models exhibit substantial confusion between asphalt and concrete, indicating limitations in discriminating visually similar textures using RGB data alone. These findings suggest that fine-grained material classification may benefit from model architectures incorporating attention mechanisms, texture-aware encoding, or multi-modal input such as multi-spectral images. While architectural differences have minimal impact on average performance, our results emphasize the importance of addressing class-level ambiguity through targeted model design and data-driven strategies. This study provides a rigorous baseline for future research in road surface understanding and contributes to the development of more robust vision-based infrastructure analysis systems.

KEYWORDS

Computer vision, Road surface classification, street-view imagery, ResNet-50, ConvNeXt-Small, EfficientNet-B4, StreetSurfaceVis

1 INTRODUCTION

Accurate classification of road surface types is becoming increasingly important in applications ranging from intelligent transportation systems to autonomous vehicles and open-source routing platforms [2]. The condition and material of road surfaces directly affect travel time estimation, vehicle dynamics, and route planning efficiency. Recent advancements in computer vision and deep learning have enabled automated visual recognition of surface types from images, offering scalable solutions for infrastructure monitoring and mobility optimization [15].

With the growing availability of annotated road surface classification datasets such as Road Traversing Knowledge [16] and StreetSurfaceVis [10], and the rapid development of novel convolutional neural network (CNN) architectures, the question arises: which models are best suited for robust road surface classification? Despite the technological progress, there remains a lack of comprehensive, up-to-date comparison studies that evaluate modern deep learning models on standardized benchmarks in this specific domain.

In this study, we present a systematic evaluation of three well-established CNN architectures - ResNet-50, ConvNeXt-Small, and EfficientNet-B4 - using the StreetSurfaceVis dataset [10], which contains a diverse set of road surfaces captured under real-world conditions. We analyze model performance both in terms of overall accuracy and class-specific F1-scores, with particular attention to the frequent confusion between visually similar classes such as asphalt and concrete. We further discuss model limitations and suggest potential improvements, including the use of attention mechanisms, texture-aware modules, and multi-modal inputs. Our findings provide a rigorous performance baseline and highlight key challenges that must be addressed to advance road surface understanding in real-world applications.

2 MATERIALS AND METHODS

Our experimental pipeline begins with the retrieval of the StreetSurfaceVis dataset, followed by the development of a custom data loader to facilitate standardized data splitting and preprocessing. The dataset is partitioned into training, validation, and test sets, and all images undergo consistent transformations, including resizing and normalization, to ensure compatibility across model inputs. We evaluate three state-of-the-art convolutional neural network architectures—ConvNeXt-Small, ResNet-50, and EfficientNet-B4—using pre-trained weights from the `torchvision` library[5]. Each model is fine-tuned by replacing the final classification layer to match the number of road surface classes in the dataset. All other network weights remain initialized from ImageNet-pretraining to leverage learned visual representations while minimizing training time and overfitting. Model performance is tracked across training epochs using standard metrics, including training, validation, and test accuracy. To assess class-specific performance, we compute F1-scores, generate confusion matrices, and analyze class-wise error rates. These evaluations allow us to identify persistent ambiguities, particularly among visually similar surface types. To further interpret model behavior, we apply Gradient-weighted Class Activation Mapping (Grad-CAM), enabling the visualization of salient regions contributing to class decisions. Misclassified samples are qualitatively examined to understand model limitations and inform potential improvements, such as enhanced feature extraction or data augmentation strategies. All code necessary to reproduce this study is available in the supplementary GitHub repository [4].

2.1 Data Retrieval

2.1.1 StreetSurfaceVis Dataset. StreetSurfaceVis dataset [10], published in 2024, is a publicly available resource that can enable fine-grained classification of road surface types based on street-level

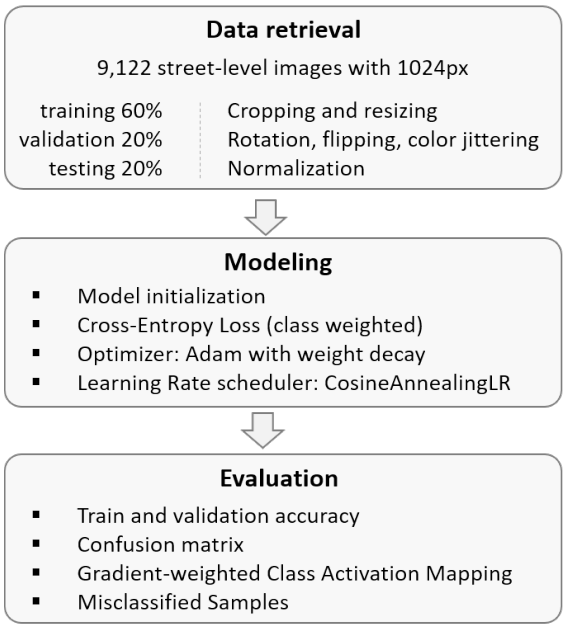


Figure 1: Overview of the experimental workflow. The workflow consists of data acquisition and preprocessing, model adaptation using three pre-trained CNN architectures (ConvNeXt-Small, ResNet-50, and EfficientNet-B4), training and validation on the StreetSurfaceVis dataset, and performance evaluation using class-wise metrics, confusion matrices, and Grad-CAM visualizations. Misclassified samples are analyzed to inform future model improvement strategies.

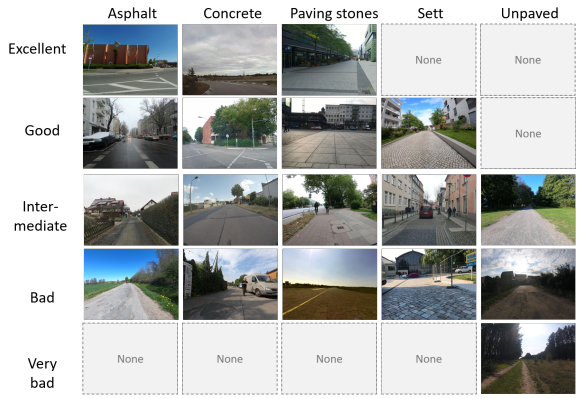


Figure 2: StreetSurfaceVis includes annotations for five road surface types (asphalt, concrete, paving stones, sett, and unpaved) and five levels of road quality (excellent, good, intermediate, bad, and very bad). 'None' indicates missing categories from the original dataset, which are excluded from training, validation and testing.

imagery. The dataset consists of 9,122 georeferenced images sourced from the Mapillary platform [12], with each image uniquely identified by a Mapillary image ID, user ID, and associated geographic coordinates. StreetSurfaceVis captures a broad spectrum of real-world scenarios, including roadways, cycleways, and pedestrian pathways, under varying environmental conditions such as different lighting, weather, and visibility levels. The dataset exhibits a high degree of visual heterogeneity, including variations in brightness, sharpness, and motion-induced blur, making it particularly suitable for developing and evaluating models that must generalize across diverse input conditions. Each image is annotated with surface type labels—asphalt, concrete, paving stones, sett, and unpaved—and surface quality labels—excellent, good, intermediate, bad, and very bad. These annotations are primarily derived from OpenStreetMap (OSM) metadata, specifically the surface and smoothness tags, and correspond to the road segment located at the bottom center of each image. Images are provided at four resolution levels: 256 px, 1024 px, 2048 px, and full original resolution, allowing for resolution-dependent performance analyses. In addition to the annotated image corpus, the dataset includes a comprehensive labeling guideline and documentation, and is freely accessible via Zenodo [9, 10].

For this study, we utilize StreetSurfaceVis images at a resolution of 1024 px to strike a balance between preserving fine-grained surface texture details and maintaining computational efficiency during model training. Customized datasets are constructed by extracting image identifiers and corresponding surface type annotations from the provided metadata, enabling accurate label-image mapping via unique image IDs. The dataset is stratified into training (60%), validation (20%), and test (20%) subsets. Following the dataset's labeling protocol, each image is cropped to its bottom-center region, which corresponds to the focal road surface annotated in the metadata. This region is most representative of the labeled surface class and minimizes the influence of irrelevant background content. To augment the training set and improve generalization, we apply a series of transformations: images are resized to 384 px, followed by random horizontal flipping (probability $p = 0.5$), slight color jittering (brightness = 0.2, contrast = 0.2), and pixel value normalization. Validation and test images undergo only resizing and normalization to ensure consistent evaluation without introducing artificial variation. Class distribution statistics are summarized in Table 1, which shows a dominance of asphalt samples and a relative scarcity of concrete and unpaved surfaces. To mitigate this class imbalance, we apply inverse-frequency weighting within the loss function (`CrossEntropyLoss`), thereby assigning higher penalties to errors on underrepresented classes.

2.2 Modeling

To provide a balanced evaluation across architectures, we select three CNNs that reflect a range of design paradigms: a widely adopted baseline, a modern architecture with transformer-inspired design, and an efficient model optimized for parameter and computational efficiency.

- **ResNet-50** serves as the baseline due to its proven robustness in image classification and feature extraction across vision tasks [6].

Table 1: Frequency of road surface types across training, validation, and test sets. ICW = Inverse Class Weight, computed as the inverse frequency of each class normalized to asphalt.

Surface Type	Train	Validation	Test	Total	ICW
Asphalt	2390	597	747	3734	1.00
Concrete	622	156	194	972	3.84
Paving Stones	1303	326	408	2037	1.83
Sett	872	218	273	1363	2.74
Unpaved	650	163	203	1016	3.68

- **ConvNeXt-Small** integrates convolutional efficiency with architectural innovations inspired by vision transformers, achieving strong performance with relatively low complexity [11].
- **EfficientNet-B4** employs compound scaling of depth, width, and resolution to achieve state-of-the-art accuracy with fewer parameters [18].

All models are initialized with ImageNet pre-trained weights from the `torchvision` library[5]. Only the final classification layer is replaced to match the five surface type classes; all other parameters remain unchanged at initialization. A major challenge is the class imbalance present in the dataset, particularly the underrepresentation of *concrete* and *unpaved* surfaces (see Table 1). To address this, we employ an *inverse frequency-based weighting* scheme in combination with the standard cross-entropy loss, a method shown to be effective for classification and detection tasks under class imbalance [3, 14, 20]. Given a dataset with C classes and class frequencies n_i, the raw weight for each class is computed as:

$$w_i^{(\text{raw})} = \frac{1}{n_i}, \quad i = 1, 2, \ldots, C \tag{1}$$

To ensure numerical stability and interpretability, these weights are normalized by dividing by the smallest weight:

$$w_i = \frac{w_i^{(\text{raw})}}{\min(w_1^{(\text{raw})}, \ldots, w_C^{(\text{raw})})} = \frac{\max(\mathbf{n})}{n_i} \tag{2}$$

As a result, the most frequent class *asphalt* is assigned a weight of 1.00, while the rare classes *concrete* and *unpaved* receive weights of 3.84 and 3.68, respectively. These normalized weights are applied during training to penalize misclassifications of underrepresented classes without altering the dataset distribution.

All models are trained under consistent experimental settings to enable a fair comparison of architectural performance. We employ the Adam optimizer with a weight decay of 0.05 to promote generalization. The initial learning rate is uniformly set to 0.0001 across all models and scheduled using a cosine annealing strategy (`CosineAnnealingLR`) to gradually reduce the learning rate over time and facilitate convergence. Due to hardware constraints and varying model sizes, batch sizes are adjusted per architecture: 32 for ResNet-50, 16 for EfficientNet-B4, and 8 for ConvNeXt-Small. To ensure model convergency, both ResNet-50 and EfficientNet-B4 are trained for 150 epochs while ConvNeXt-Small is trained for 200 epochs on the same training/validation split using the class-weighted cross-entropy loss function described in Equation 1.

3 RESULTS

Figure 3 illustrates the training accuracy progression for all three models, showing convergence around 30 epochs, suggesting that the models have sufficiently learned the road surface patterns. However, the validation accuracy demonstrates distinct behaviors across the models:

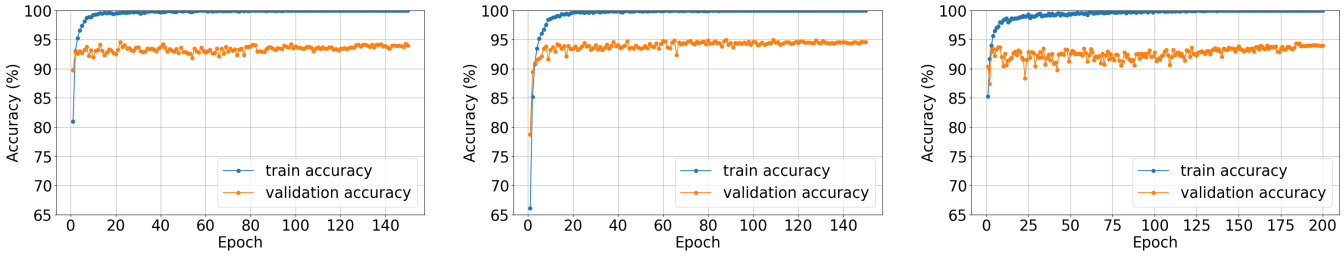
- **ResNet-50** (Figure 3a) reaches a plateau at approximately 93% validation accuracy after 30 epochs, indicating stable performance.
- **EfficientNet-B4** (Figure 3b) exhibits a rapid initial improvement, stabilizing around 94% during the final epochs.
- **ConvNeXt-Small** (Figure 3c) shows significant fluctuations in the early epochs, but the model steadily improves steadily after 125 epochs.

Figure 4 presents the validation accuracy for each class across the applied models. All models struggle with the classification of the *concrete* class, which likely results from the inherent variations in color, texture, and surface patterns (ResNet-50: 81.96%, EfficientNet-B4: 82.47%, ConvNeXt-Small: 81.44%). In contrast, the models perform exceptionally well on the *sett* class, achieving high validation accuracies (ResNet-50: 93.41%, EfficientNet-B4: 96.34%, ConvNeXt-Small: 95.97%).

Table 2 present the precision, recall, F1-score, and Figure 5 confusion matrices for the three models evaluated on the test set. All models exhibit strong overall performance, with test accuracies ranging from 0.94 to 0.95. However, notable differences emerge in the models' ability to handle specific classes. All architectures excel at classifying *Paving Stones* (F1-scores ≥ 0.96) and *Sett* (F1-scores ≥ 0.95), while performance on *Concrete* classification is relatively weaker, particularly for EfficientNet-B4 (F1-scores = 0.84).

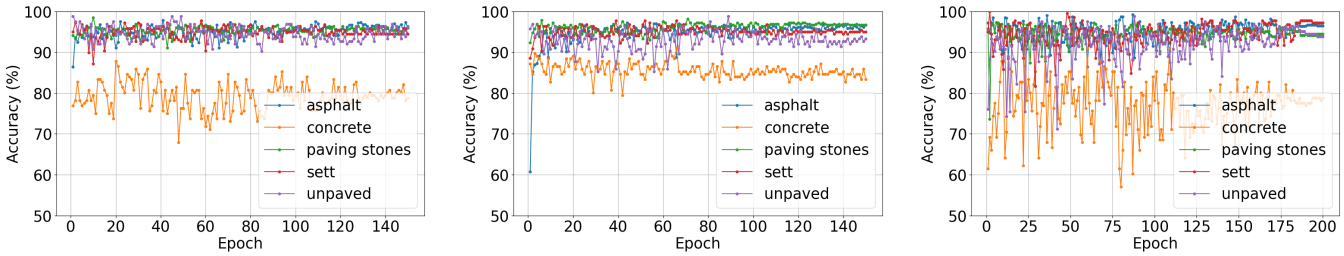
- **ConvNeXt-Small** achieves the highest accuracy (95.18%), with very few misclassifications across all classes. It excels at distinguishing *Asphalt* and *Paving Stones*, with minimal confusion. However, there is slight misclassification between *Asphalt* and *Concrete*, as well as a few *Asphalt* samples misclassified as *Unpaved*.
- **EfficientNet-B4** achieves 94.58% accuracy. It demonstrates strong performance but experiences slightly more confusion compared to ConvNeXt Small, particularly between *Asphalt* and *Concrete*. Notably, 38 samples of *Asphalt* are misclassified as *Concrete*, and minor confusion is observed between *Unpaved* and other classes.
- **ResNet-50** achieves 94.19% accuracy. Although its overall performance is competitive, there is moderate confusion between *Asphalt* and *Concrete*, along with some misclassifications between *Paving Stones* and *Sett*. Misclassifications are more evenly distributed across the classes compared to the other models.

The minimal discrepancy between macro-averaged (0.92–0.95) and weighted averages (0.94–0.95) indicates a balanced class distribution in the test set. Across all models, *Concrete* proves to be the most challenging class to classify correctly, as evidenced by the relatively higher misclassification rates. In contrast, *Paving*



(a) ResNet-50 (b) EfficientNet-B4 (c) ConvNeXt Small

Figure 3: Training and validation accuracy for different models.



(a) ResNet-50 (b) EfficientNet-B4 (c) ConvNeXt Small

Figure 4: Validation accuracy per class.

Stones and *Sett* are consistently recognized with high precision and recall. These results suggest that while all models perform well, ConvNeXt-Small's consistent outperformance in precision, recall, and F1-score positions it as the optimal choice among the evaluated architectures, assuming computational resources are available for its deployment.

4 DISCUSSION

This study highlights key challenges in road surface classification, particularly in distinguishing visually similar surfaces such as asphalt and concrete. Despite the application of advanced deep learning architectures (ResNet-50, ConvNeXt-Small, and EfficientNet-B4), these models exhibit persistent confusion between these classes, indicating that RGB images alone may not capture sufficient discriminative features for fine-grained material classification. To address this, future models should integrate advanced architectural designs, particularly those leveraging multi-modal inputs.

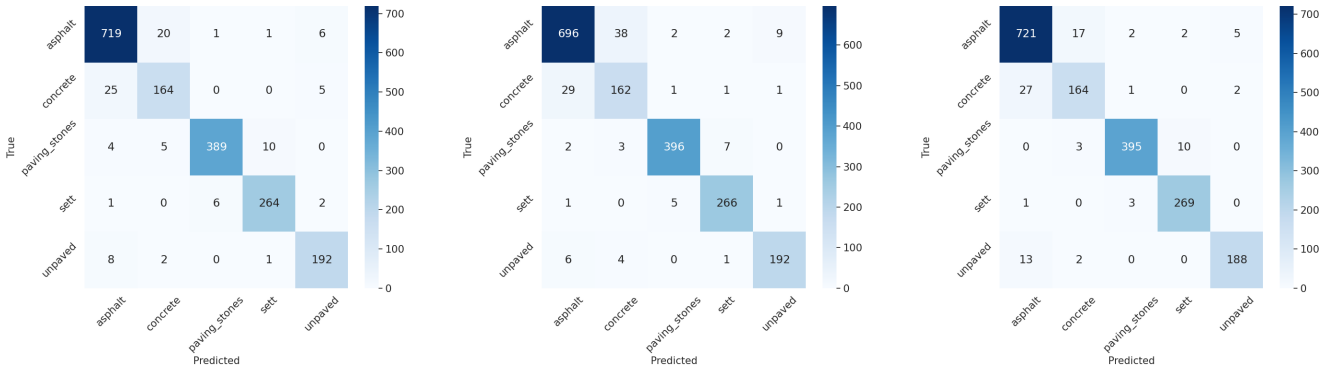
4.1 Dataset

Dataset Size. The size of the dataset is a critical factor in domain transferability, especially for deep learning models pretrained on large-scale datasets like ImageNet. Models such as ConvNeXt, EfficientNet, and ResNet generally benefit from fine-tuning on extensive datasets to achieve robust adaptation. However, our road surface dataset contains fewer than 1,000 samples, which may limit optimal generalization. To address this, we utilize compact model variants like ConvNeXt Small[11], which are less susceptible to overfitting on limited data. For tasks constrained by small datasets, approaches such as few-shot learning[13] and knowledge distillation[21] can also be beneficial. Future research could further explore model performance by benchmarking on additional datasets, such as the one published by [22] published in 2022, which exceeds the size of StreetSurfaceVis.

Input Modalities. Incorporating multi-spectral or hyperspectral imaging presents a promising solution, as these modalities capture surface properties invisible to RGB imaging, such as thermal signatures and material-specific reflectivity. This could improve performance, particularly for distinguishing between surface types with similar visual characteristics. Furthermore, incorporating attention mechanisms, such as self-attention or transformers, could help models focus on key image regions, facilitating the detection of subtle textures, cracks, or wear patterns that differentiate surface types. Additionally, multi-scale feature extraction and texture-aware encoding may enhance texture discrimination, addressing challenges in classifying surfaces with fine-grained textural differences.

Table 2: Comparison of model performance on test dataset

Class	ResNet-50			EfficientNet-B4			ConvNeXt-Small		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
Asphalt	0.94	0.97	0.95	0.95	0.96	0.95	0.95	0.97	0.96
Concrete	0.88	0.82	0.85	0.86	0.82	0.84	0.92	0.81	0.87
Paving Stones	0.97	0.95	0.96	0.97	0.97	0.97	0.97	0.97	0.97
Sett	0.97	0.93	0.95	0.96	0.96	0.96	0.96	0.97	0.97
Unpaved	0.92	0.95	0.94	0.95	0.94	0.94	0.95	0.96	0.95
Accuracy	-	0.94	-	-	0.95	-	-	0.95	-
Macro Avg	0.94	0.92	0.93	0.94	0.93	0.93	0.95	0.93	0.94
Weighted Avg	0.94	0.94	0.94	0.95	0.95	0.95	0.95	0.95	0.95



(a) ResNet-50 **(b) EfficientNet-B4** **(c) ConvNeXt-Small**

Figure 5: Confusion matrix on test dataset for different models.

Category of Classes. Expanding the current five-class dataset is another critical consideration. To improve the model's ability to distinguish subtle differences, fine-grained subclasses, such as differentiating between wet and dry asphalt, should be introduced. Moreover, testing the models across diverse geographical distributions—beyond the StreetSurfaceVis dataset, which primarily represents road surfaces from Germany—would assess model robustness and performance in varying regional contexts [7]. The reliance of current models on color cues contributes to confusion between asphalt and concrete; multispectral data integration could address this issue by emphasizing textural features over color.

4.2 Model Selection

Pre-trained Model on Domain-specific Datasets. Leveraging pre-trained models is another avenue for improvement. Pre-training on domain-specific datasets, such as Mapillary Vistas, could enable models to learn latent road topology features, enhancing classification performance. Additionally, semi-supervised learning techniques applied to large-scale, unlabeled road imagery could help the models generalize across diverse surface types. Integrating attention mechanisms from Vision Transformer (ViT)-based road segmentation models could further refine the model's focus on relevant surface features, enhancing its discriminative power.

Computation Costs. Training costs and parameters for inference should be carefully considered. ConvNeXt-Small, which achieved the highest accuracy (95.18%), required more epochs to converge compared to ResNet-50. While it offers superior accuracy, its longer training time and large parameter count (49,458,533) may limit its suitability for real-time systems with resource constraints, especially in the absence of model compression techniques. On the other hand, EfficientNet-B4, with a smaller parameter count (17,557,581) and faster inference times, achieved slightly lower accuracy (94.58%)

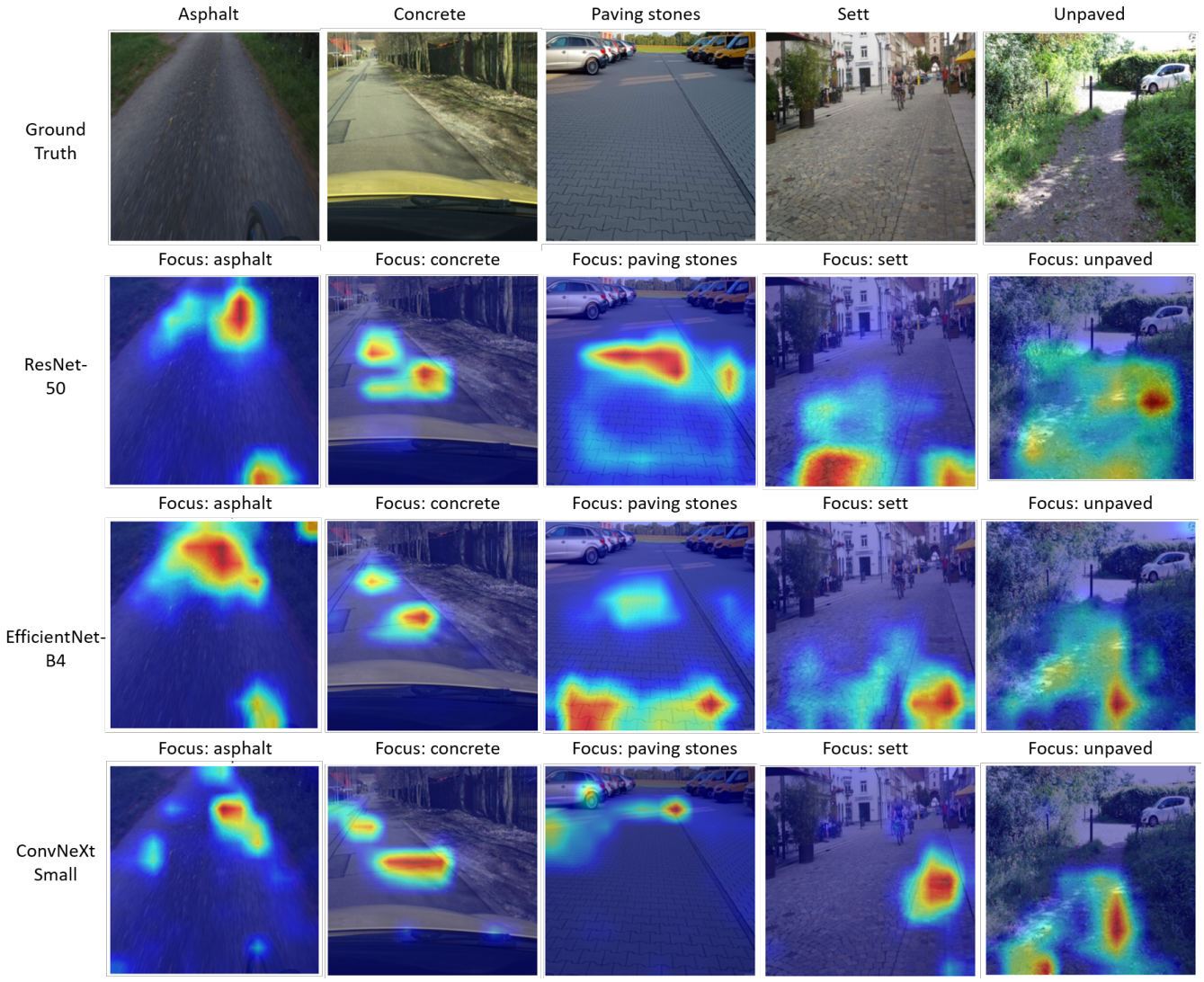


Figure 6: Gradient-weighted class activation mapping on asphalt, concrete, paving stones, sett and unpaved.

but may be more appropriate for resource-constrained environments. Task-specific model selection, balancing accuracy and efficiency, is thus crucial for practical deployment.

4.3 Training Strategies

Hyper-parameter optimization. Hyperparameter optimization plays a critical role in model performance. We adopted a uniform initial learning rate (0.0001) and cosine annealing scheduling for consistency across experiments, this approach may not be optimal for all models. A systematic exploration such as random search[1] could identify better training configurations.

Data augmentation. Data augmentation strategies also offer significant potential to improve model robustness [17]. The analysis of misclassified samples reveals that introducing variability in road surface appearance through techniques like color jittering [19] (to simulate lighting variations) and synthetic defect generation [8] (e.g., potholes, cracks) could help models generalize better to real-world conditions. Texture-preserving augmentations, which modify surface appearance without distorting underlying structural features, may particularly benefit the classification of challenging categories like concrete.

In summary, while deep learning models show considerable promise for road surface classification, addressing the challenges identified here will be essential for improving performance. Future advancements should focus on multi-modal data integration, refining architectural features with attention and texture-aware mechanisms, and expanding the dataset to include more diverse and fine-grained surface categories. These improvements, along with leveraging pre-trained models, will increase model robustness and generalization, making road surface classification systems more applicable to a wide range of real-world scenarios.

CONCLUSION

This study provides a comprehensive benchmarking of state-of-the-art deep learning models for road surface classification using the StreetSurfaceVis dataset. While all evaluated models achieved high overall accuracy, class-specific challenges—particularly the confusion between asphalt and concrete—highlight limitations of RGB-based classification for visually similar materials. Our findings demonstrate that improving road surface classification requires more than architectural advances alone; it also calls for richer input modalities, refined feature extraction strategies, and robust augmentation techniques. Future research should prioritize multi-modal imaging, attention-based architectures, and broader geographic and categorical coverage to enhance generalization and practical applicability. The presented results establish a strong performance baseline and offer key insights to guide the development of next-generation vision-based road infrastructure analysis tools.

ACKNOWLEDGMENTS

We thank Heidelberg University's Computing Centre for Accesses to the SDS@hd hot-data storage. Support by the state of Baden-Württemberg through bwHPC (Helix Cluster) and the German Research Foundation (DFG) through grant INST 35/1597-1 FUGG is greatfully acknowledged.

REFERENCES

[1] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* 13, null (Feb. 2012), 281–305.

[2] Adrian Paul Botezatu, Adrian Burlacu, and Ciprian Orhei. 2024. A Review of Deep Learning Advancements in Road Analysis for Autonomous Driving. *Applied Sciences (Switzerland)* 14, 11 (2024). https://doi.org/10.3390/app14114705

[3] W. Dang, Z. Yang, W. Dong, X. Li, and G. Shi. 2024. Inverse Weight-Balancing for Deep Long-Tailed Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 11713–11721. https://doi.org/10.1609/aaai.v38i10.29055

[4] Abanto D. Zipf. A. Knoblauch S Duan, R. 2025. GitHub repository for this manuscript called Benchmarking Deep Learning Models for Road Surface Classification on StreetSurfaceVis. https://doi.org/10.5281/zenodo.15497076

[5] Daniel Falbel. 2025. torchvision: Models, Datasets and Transformations for Images. https://github.com/mlverse/torchvision R package version 0.6.0.9000.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. arXiv:1502.01852 [cs.CV] https://arxiv.org/abs/1502.01852

[7] Yuanyuan Hu, Ning Chen, Yue Hou, Xingshi Lin, Baohong Jing, and Pengfei Liu. 2025. Lightweight deep learning for real-time road distress detection on mobile devices. *Nature Communications* 16, 1 (may 2025), 4212. https://doi.org/10.1038/s41467-025-59516-5

[8] I A Kanaeva and Ju A Ivanova. 2021. Road pavement crack detection using deep learning with synthetic data. *IOP Conference Series: Materials Science and Engineering* 1019, 1 (jan 2021), 012036. https://doi.org/10.1088/1757-899X/1019/1/012036

[9] Alexandra Kapp, Edith Hoffmann, Esther Weigmann, and Helena Mihaljević. 2024. StreetSurfaceVis: a dataset of crowdsourced street-level imagery annotated by road surface type and quality. https://zenodo.org/records/11449977. https://doi.org/10.5281/zenodo.11449977

[10] Alexandra Kapp, Edith Hoffmann, Esther Weigmann, and Helena Mihaljević. 2025. StreetSurfaceVis: a dataset of crowdsourced street-level imagery annotated by road surface type and quality. *Scientific Data* 12, 1 (Jan. 2025). https://doi.org/10.1038/s41597-024-04295-9

[11] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. 2022. A ConvNet for the 2020s. arXiv:2201.03545 [cs.CV] https://arxiv.org/abs/2201.03545

[12] Mapillary. [n. d.]. Mapillary. https://www.mapillary.com/.

[13] Archit Parnami and Minwoo Lee. 2022. Learning from Few Examples: A Summary of Approaches to Few-Shot Learning. arXiv:2203.04291 [cs.LG] https://arxiv.org/abs/2203.04291

[14] Trong Huy Phan and Kazuma Yamamoto. 2020. Resolving Class Imbalance in Object Detection with Weighted Cross Entropy Losses. arXiv:2006.01413 [cs.CV] https://arxiv.org/abs/2006.01413

[15] Sukanya Randhawa, Eren Aygün, Guntaj Randhawa, Benjamin Herfort, Sven Lautenbach, and Alexander Zipf. 2025. Paved or unpaved? A deep learning derived road surface global dataset from mapillary street-view imagery. *ISPRS Journal of Photogrammetry and Remote Sensing* 223 (2025), 362–374. https://doi.org/10.1016/j.isprsjprs.2025.02.020

[16] Thiago Rateke, Karla Aparecida Justen, and Aldo von Wangenheim. 2019. Road Surface Classification with Images Captured From Low-cost Cameras – Road Traversing Knowledge (RTK) Dataset. *Revista de Informática Teórica e Aplicada (RITA)* (2019). https://doi.org/10.22456/2175-2745.91522

[17] Alberto Signoroni, Mattia Savardi, Annalisa Baronio, and Sergio Benini. 2019. Deep Learning Meets Hyperspectral Image Analysis: A Multidisciplinary Review. *Journal of Imaging* 5, 5 (may 2019), 52. https://doi.org/10.3390/jimaging5050052

[18] Mingxing Tan and Quoc V. Le. 2020. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. arXiv:1905.11946 [cs.LG] https://arxiv.org/abs/1905.11946

[19] Luke Taylor and Geoff Nitschke. 2018. Improving Deep Learning with Generic Data Augmentation. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 1542–1547. https://doi.org/10.1109/SSCI.2018.8628742

[20] Junjiao Tian, Niluthpol Mithun, Zach Seymour, Han-Pang Chiu, and Zsolt Kira. 2022. Striking the Right Balance: Recall Loss for Semantic Segmentation. arXiv:2106.14917 [cs.CV] https://arxiv.org/abs/2106.14917

[21] Wenxuan Yang, Qingqu Wei, Chenxi Ma, Weimin Tan, and Bo Yan. 2025. Scaling Laws for Data-Efficient Visual Transfer Learning. arXiv:2504.13219 [cs.LG] https://arxiv.org/abs/2504.13219

[22] Tong Zhao and Yintao Wei. 2022. A road surface image dataset with detailed annotations for driving assistance applications. *Data in Brief* 43 (2022), 108483. https://doi.org/10.1016/j.dib.2022.108483

PV Panel Detection from High-Resolution Aerial Imagery in Heidelberg using Deep Learning

Maren Strydhorst
Maximiliane Kitzinger

Figure 1: Roboflow Annotations of Heidelberg Orthophotos.

Abstract

The rapid expansion of photovoltaic (PV) systems plays a crucial role in the transition to renewable energy. However, many PV installations remain unregistered, making it difficult for stakeholders to accurately forecast solar energy generation. This study presents a methodology for the automated detection of PV installations using high-resolution orthophotos and deep learning. The research focuses on Heidelberg, Germany, leveraging publicly available aerial imagery (DOP20) to train a YOLOv11-based model. A dataset of manually annotated PV installations was created using Roboflow. The dataset was split into training (80%), validation (10%), and testing (10%) sets, and the model was trained for 50 epochs to prevent overfitting. Evaluation metrics, including precision, recall, and F1 score, indicate a promising detection accuracy of 74%, though challenges such as false positives, false negatives, and dataset limitations persist. The study highlights the importance of diverse training data and proposes improvements for generalizability across different geographic regions. The findings demonstrate the potential of AI-driven remote sensing for urban energy planning and monitoring PV adoption at scale. For more information see the corresponding GitHub repository.

Keywords

Aerial Imagery, Photovoltaic, Object Detection, Deep Learning, YOLOv11

1 Introduction

In the contemporary era, characterised by climate change and the necessity for sustainable energy generation, solar energy has become increasingly important across residential, commercial, and industrial sectors [13]. The demand for up-to-date, accurate, and large-scale mapping of installed PV systems for stakeholders within the energy sector, including market operators and network operators is increasing [4]. These stakeholders need to forecast the generation of rooftop solar PV energy at the level of entire regions [4]. The data is not only essential for economic forecasting [3] but also for energy policy planning, grid management and maintaining [1], and monitoring progress towards climate goals. Nevertheless, a significant number of photovoltaic systems have not been accurately registered, and central records are not up to date [4]. It has been demonstrated that conventional data collection techniques, such as surveys and utility interconnection filings, are constrained by limitations pertaining to completeness and spatial resolution [9].

The application of aerial photography and satellite imagery facilitates analysis of both natural and constructed environments [2]. Consequently, PV systems that are visible from above can be assessed using remote sensing data and machine learning algorithms [2, 4, 9]. In circumstances where existing data is unavailable, the application of aerial imagery in conjunction with automated detection algorithms has been demonstrated to enhance the efficiency and accuracy of the data collection process [5].

In the present study, the focus is on the development of a methodology for the automatic detection of photovoltaic systems using

orthophotos. The methodology involves the training of a model based on the 80/10/10 principle, with subsequent validation using a separate dataset. The primary research question guiding this study is to determine how accurate deep learning models can detect PV systems in high-resolution aerial imagery.

2 Materials and Methods

This chapter provides a detailed overview of the entire workflow, covering all key steps involved in the study. It begins with a description of the the study area and dataset, outlining the sources, characteristics, and relevance of the data used. Subsequently, a step-by-step explanation of the workflow is presented, including data preprocessing, annotation, model training, and evaluation method. This structured approach ensures transparency and reproducibility of the methodology.

2.1 Study Area

Heidelberg is located in southern Germany and has a population of 163.000 inhabitants [10]. As a part of the federal state of Baden-Württemberg, Heidelberg is obligated to comply with the state-mandated policy of integrating photovoltaic technology into newly constructed buildings and conducting comprehensive roof renovations [6]. Heidelberg has been implementing simplified approval processes, which have contributed to an accelerated expansion in recent years. The city has also been promoting the expansion of solar installations as part of its transformation to renewable energies [6]. The city itself has been installing 63 solar power systems not only on new buildings, but also on all municipal buildings where the load-bearing capacity and state of maintenance allow [6]. As of August 2024, the PV systems that had been installed were capable of meeting the electricity needs of 18,000 households [6].

2.2 Data

The dataset utilized in this study was obtained from the Open GeoData Portal of the State Office for Geoinformation and Land Development (LGL) of Baden-Württemberg. Specifically, we accessed Digital Orthophotos (DOP20), which are high-resolution aerial images that have been georeferenced and orthorectified to eliminate distortions. These images provide a spatial resolution of 20cm per pixel, meaning that each pixel represents a real-world area of 20cm × 20cm [11].

The DOP20 imagery is derived from annual aerial surveys conducted by the LGL, covering approximately half of the state of Baden-Württemberg each year. The photographs are captured using digital aerial cameras, and the resulting orthophotos are available in both natural color (RGB) and infrared-enhanced (RGBI) formats. The data is referenced in the ETRS89/UTM coordinate system and is delivered as distortion-free, true-to-scale raster images. Notably, the TrueDOP format ensures that the images are free from perspective distortions or occluded areas [11].

For this study, the DOP20 dataset was downloaded in TIFF format, with each image tile covering an area of 2km × 2km. The dataset is updated annually to ensure temporal accuracy. The following eight RGB image tiles were selected: 475-5474, 477-5474, 475-5472, 477-5472, 475-5470, 477-5470, 475-5468, 477-5468.

These orthophotos provide a valuable high-resolution representation of Heidelberg, covering most of the city's urban neighborhoods. The study area extends from Handschuhsheim in the north to Kirchheim and Emmertsgrund in the south (Fig. 2). This area was selected due to its high building density and the presence of numerous rooftops with potential photovoltaic installations.

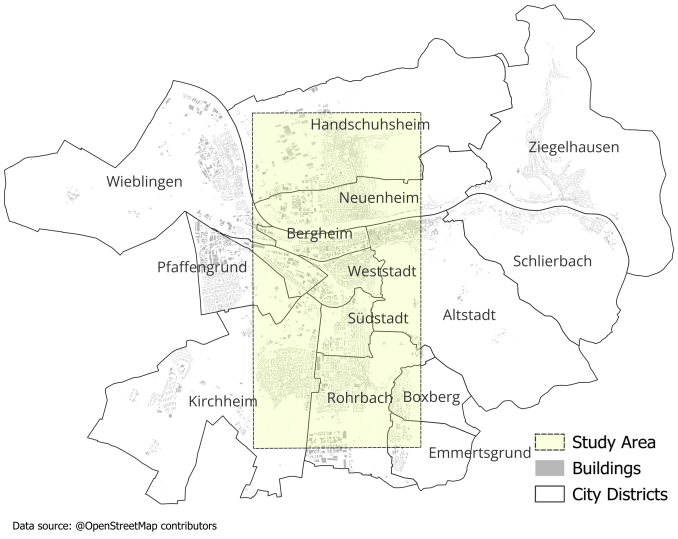


Figure 2: Study Area in Heidelberg. Black lines indicate district boundaries, grey areas represent buildings, and the yellow area marks the selected region of interest for model training and evaluation.

2.3 Methods

The overall workflow of the project is summarized in Figure 3, illustrating the steps from data collection to model evaluation. The process begins with the download and preprocessing of the data. Relevant datasets are acquired, and the orthophotos are converted into suitable formats for annotation. Following the preprocessing, the photovoltaic installations within the orthophotos are manually annotated using Roboflow. These annotations serve as labeled data crucial for training the YOLOv11 model. Once the dataset has been annotated, it is divided into training, validation and testing sets. The model is then trained using the training data, while the validation set is employed to monitor the model's performance. Finally, the model is tested using the separate test dataset to evaluate its performance.

First, the dataset was downloaded as described in the previous section. The raw data consisted of individual image tiles. In the preprocessing stage, these tiles were merged into a single large image using QGIS. The merged image was then retiled into smaller tiles, each sized 640×640 pixels. This process resulted in a folder containing 2,016 .tif files.

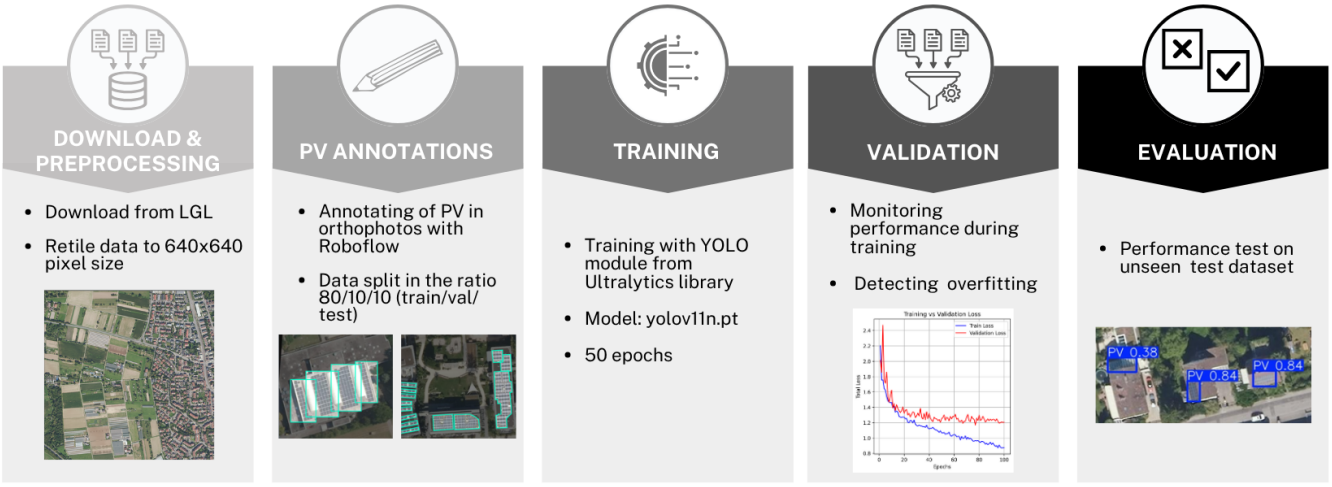


Figure 3: Overview of the proposed workflow for automated PV detection.

Next, an R script was written and run to convert the .tif files into .jpg format, as Roboflow does not accept .tif files. These converted images were then uploaded to the Roboflow platform for annotation.

Roboflow was selected for this study because of its user-friendly interface and collaborative features, which allow multiple users to work on the project simultaneously [7]. For the annotation process, all photovoltaic (PV) installations were manually labeled as polygons with the class 'PV'. Images that did not contain any PV installations or were partially cut off during the tiling process were labeled as 'N_PV' and later removed from the dataset.

After the annotation was completed, 585 images containing 2.558 annotated PV installations remained. To ensure that the images were the correct size for training, they were resized to 640×640 pixels, based on Ultralytics' requirements, using a stretch method [12]. Subsequently, the dataset was randomly divided into three categories: 470 images for training (80%), 57 images for validation (10%), and 58 images for testing (10%). Finally, the dataset was downloaded to a local computer for further use.

For training and evaluation of the model, Python was utilized in combination with the YOLO module from the Ultralytics library. The nano model (yolov11n.pt) was downloaded from the Github repository [12] and used for the training process. Ultralytics YOLO11 is a widely recognized, high-performance implementation of the YOLO (You Only Look Once) object detection algorithm, known for its speed and accuracy [12]. It was chosen due to its ease of use and availability of various model sizes, ranging from nano to extra large. The nano variant, specifically, demonstrates notable advancements in both speed and efficiency when compared to earlier versions [8]. For training, the nano model was selected due to hardware limitations - specifically, the lack of a dedicated graphics card. The training was performed using the 'train' method provided by the YOLO module, over a total of 100 epochs. Since the training results (Fig. 4) show an increasing gap between training and validation loss, indicating that overfitting is starting to occur [14], the training process was modified to run for only 50 epochs.

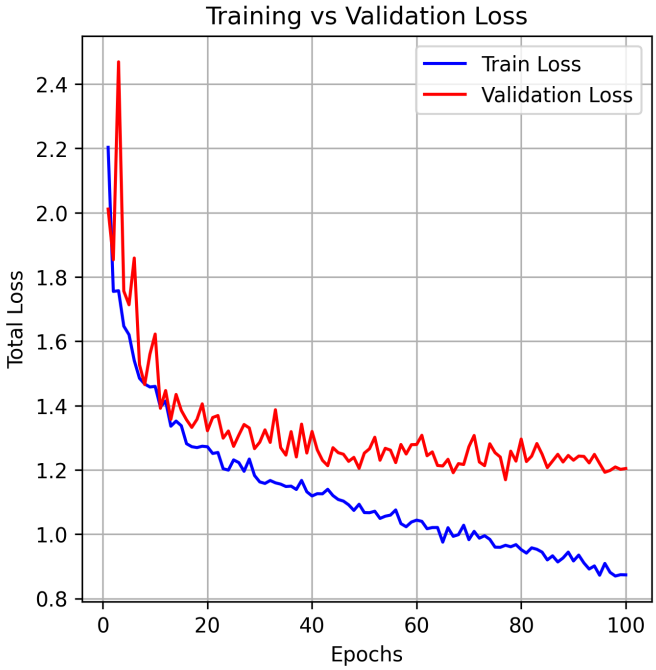


Figure 4: Training and validation loss curves of the YOLOv11 model over 100 epochs. The plot illustrates the model's learning behavior, with overfitting becoming apparent after approximately 50 epochs, as indicated by the increasing gap between training and validation loss.

Subsequently, the model was evaluated using the test dataset and the 'val' method provided by the YOLO module, which allowed

for performance assessment based on unseen data. The results of this evaluation, including metrics such as precision and recall, are discussed in detail in the Results chapter.

3 Results

After training, the model's performance on the test dataset was evaluated, and the findings were visualized through statistical charts.

One of the primary tools used to assess the model's performance is the confusion matrix, which provides a summary of the classification results (Fig. 5). In this case, the confusion matrix displayed two classes: PV and background. The matrix showed 226 true positive (PV correctly identified as PV) predictions, 183 false positive (background misclassified as PV), and 80 false negative (PV misclassified as background). 74% of the true positive PV were correctly recognized, while 26% were falsely identified as background.

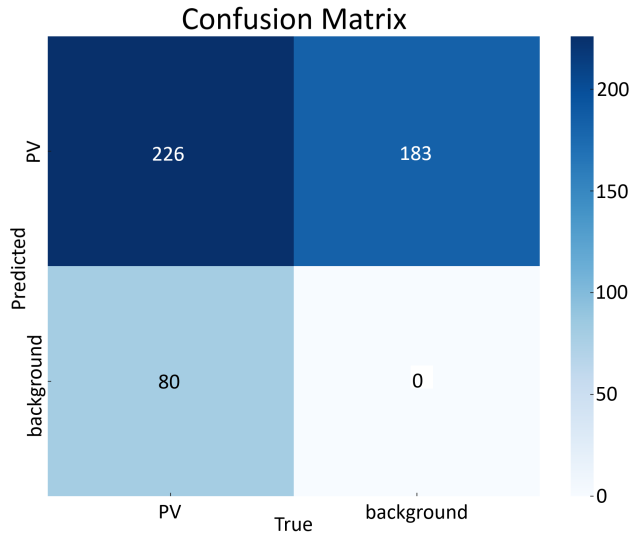


Figure 5: Confusion matrix showing true positives (226), false positives (183), false negatives (80), and true negatives. Darker blue indicates higher counts. The visualization reflects the model's ability to distinguish between PV and background.

Figure 6 presents the Precision-Confidence Curve, which illustrates the relationship between precision and confidence levels. Precision is defined as the proportion of true positives out of the sum of true positives and false positives [14]. The curve begins at a confidence of 0 and a precision of 0.1, then increases sharply, reaching a precision of 0.5 at a confidence level of 0.2. It continues to rise gradually. From a confidence threshold of 0.95 onwards, the precision reaches 1.

Next, the Recall-Confidence curve was calculated and is presented in Figure 7. Recall represents the proportion of actual positive instances that were correctly identified by the model [14]. The curve starts at a confidence level of 0.0 with a recall of 0.9 and steadily decreases as the confidence level increases, reaching a value of 0.0 at a confidence level just above 0.95.

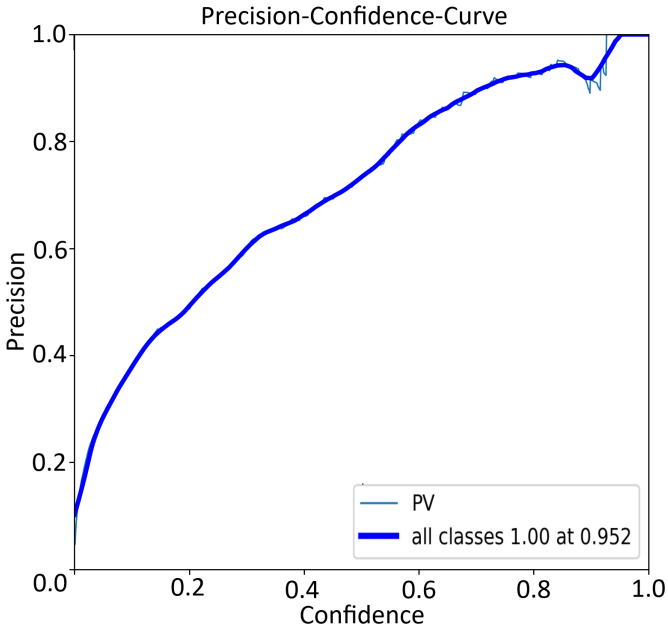


Figure 6: Precision–Confidence curve for the YOLOv11 model. Both the light and bold blue lines represent the PV class, with the bold curve showing a smoothed version of the precision trend across confidence thresholds.

The Precision-Recall curve, shown in Figure 8, provides a graphical representation of the trade-off between precision and recall. The curve starts at a precision level of 1. As the recall increases, precision decreases in a zigzag pattern. For all classes, the model achieved a mean Average Precision (mAP) of 0.679 at a threshold of 0.5. At a recall of approximately 0.9, precision drops to 0.05, and the curve continues downward in a straight line until precision reaches 0 and recall reaches 1.

Finally, the F1 score was calculated as the mean of precision and recall (Fig. 9). The curve starts at a confidence value of 0 with an F1 score of 0.1. As the confidence increases, the F1 score rises steadily, peaking at 0.65 at a confidence level of 0.33. After this point, the F1 score starts dropping, ultimately reaching 0 at a confidence level above 0.95.

The model's performance in identifying PV installations from high resolution aerial imagery is shown in figure 10. The image demonstrates how the trained model detects PV systems with the corresponding confidence level. The model's ability to recognize and classify these PV sites is evident in the results, as it successfully identifies the majority of the PV installations within the given satellite imagery.

4 Discussion

The results of this study demonstrate that deep learning-based object detection, specifically using the YOLOv11 model, is an effective method for identifying photovoltaic (PV) installations from high-resolution aerial imagery. The model successfully identified

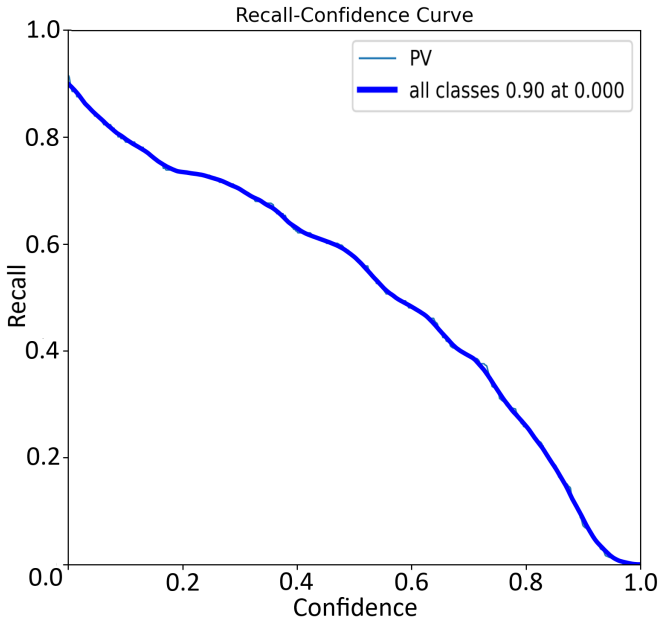


Figure 7: Recall–Confidence curve for the YOLOv11 model. Both the light and bold blue lines represent the PV class, with the bold curve showing a smoothed version of the recall trend across confidence thresholds.

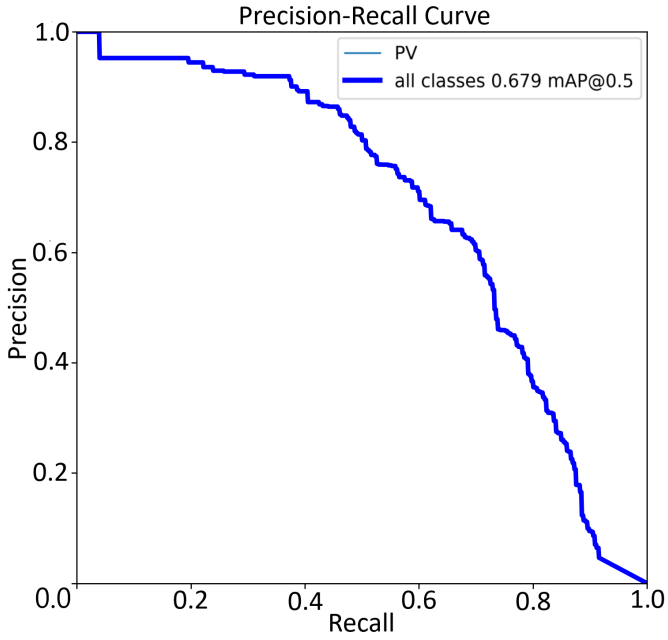


Figure 8: Precision–Confidence curve for the YOLOv11 model. The curve shows precision at varying confidence thresholds for the PV class, with the bold line representing a smoothed version of the trend.

a majority of PV installations in the test dataset, yet certain limitations remain, particularly regarding False Positives and False Negatives.

One key challenge is the relatively small dataset size, which may have limited the model's ability to generalize effectively. Additionally, potential mislabeling in the training data could have affected performance. Since annotations were manually created without cross-referencing with other datasets, their accuracy depends on human perception. Differentiating PV panels from visually similar structures, such as winter gardens, skylights or solar thermal systems, is inherently challenging and may have led to inaccuracies in the labeled data. These difficulties are also reported in the literature, where studies highlight the challenge of distinguishing between PV installations and other structures [4, 5, 14]. Due to the visual similarities and the limitations of aerial imagery, solar thermal systems were classified in the same category as PV installations in this study, as they could not be reliably distinguished by a human labeler. Addressing these issues by expanding the dataset, improving annotation quality, and integrating additional verification methods could significantly enhance the models performance.

A key limitation of this study is that the training data consisted exclusively of images captured in good lighting conditions and favorable weather. As a result, the model's performance in adverse conditions such as fog, snow, or low-light environments remains uncertain. The presence of snow or fog could obscure PV installations, leading to decreased detection accuracy. Similarly, nighttime imagery would likely pose a challenge, as the model was not trained to recognize PV panels in the absence of visible light. Future research should incorporate a more diverse dataset, including images taken under various environmental conditions, to improve the model's robustness.

Another important limitation is the applicability of the trained model to other geographic regions. Since the dataset used for training consisted exclusively of aerial imagery from Heidelberg, it is unclear how well the model would perform in different locations, particularly in regions with significantly different architectural styles and urban layouts. The model may struggle to detect PV installations in areas where rooftops and building materials differ substantially from those in Heidelberg. To improve generalizability, future studies should incorporate training data from diverse geographic regions.

The findings of this study have practical implications for urban planning and renewable energy policy. Automated detection of PV installations can support municipal authorities in monitoring solar adoption rates and assessing compliance with local regulations. In Heidelberg, where PV expansion is a key component of the city's sustainability strategy [6], this method can aid in tracking progress and optimizing resource allocation.

5 Conclusion

Overall, this study demonstrates the feasibility of using deep learning for automated PV panel detection in aerial imagery. While the YOLOv11 model has proven to be a strong starting point, it also highlights areas for improvement. The methodology developed here provides a solid foundation for further advancements in this

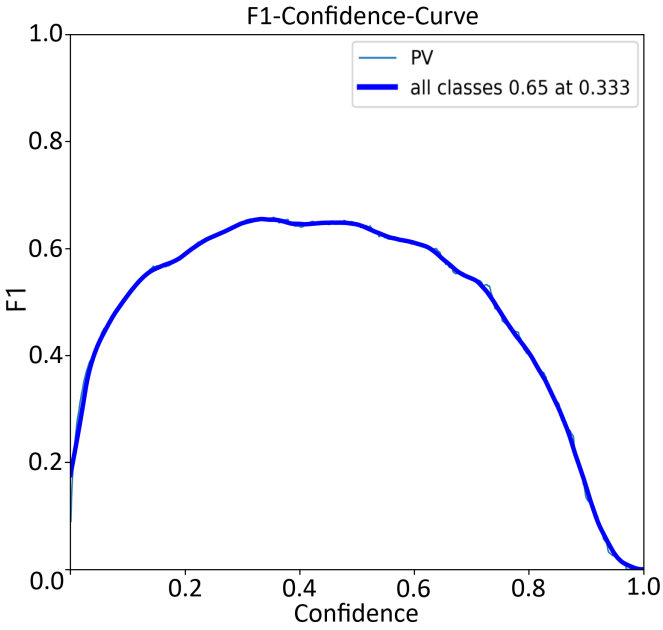


Figure 9: F1–Confidence curve for the YOLOv11 model. The curve shows the F1-Score at different confidence thresholds for the PV class, with the bold line representing a smoothed version of the trend.

field. Enhancing dataset size and quality, optimizing the model, and refining annotation techniques will be key to improving accuracy and applicability. By building on these foundations, more efficient and scalable solutions can be developed, contributing to the growing role of deep learning and remote sensing in the global transition to renewable energy.

References

[1] Montaser Abdelsattar, Ahmed AbdelMoety, and Ahmed Emad-Eldeen. 2025. Applying Image Processing and Computer Vision for Damage Detection in Photovoltaic Panels. *Mansoura Engineering Journal* 50, 2 (2025), 2.

[2] Kyle Bradbury, Raghav Saboo, Timothy L Johnson, Jordan M Malof, Arjun Devarajan, Wuming Zhang, Leslie M Collins, and Richard G Newell. 2016. Distributed solar photovoltaic array location and extent dataset for remote sensing object identification. *Scientific data* 3, 1 (2016), 1–9.

[3] Utpal Kumar Das, Kok Soon Tey, Mehdi Seyedmahmoudian, Saad Mekhilef, Moh Yamani Idna Idris, Willem Van Deventer, Bend Horan, and Alex Stojcevski. 2018. Forecasting of photovoltaic power generation and model optimization: A review. *Renewable and Sustainable Energy Reviews* 81 (2018), 912–928. doi:10.1016/j.rser.2017.08.017

[4] Julian de Hoog, Stefan Maetschke, Peter Ilfrich, and Ramachandra Rao Kolluri. 2020. Using satellite and aerial imagery for identification of solar PV: State of the art and research opportunities. In *Proceedings of the Eleventh ACM International Conference on Future Energy Systems*. 308–313.

[5] Fabio Giussani, Eric Wilczynski, Claudio Zandonella Callegher, Giovanni Dalle Nogare, Cristian Pozza, Antonio Novelli, and Simon Pezzutto. 2024. Use of Machine Learning Techniques on Aerial Imagery for the Extraction of Photovoltaic Data within the Urban Morphology. *Sustainability* 16, 5 (2024), 2020.

[6] Stadt Heidelberg. 2024. Photovoltaik-Ausbau in Heidelberg. https://www.heidelberg.de/7071_33301_34293_34309_5542057_5540639.html Zugriff am 10. Februar 2025.

[7] Roboflow Inc. 2025. Roboflow. https://roboflow.com/ Accessed: Feb 02, 2025.

[8] Rahima Khanam and Muhammad Hussain. 2024. YOLOv11: An Overview of the Key Architectural Enhancements. arXiv:2410.17725 [cs.CV] https://arxiv.org/abs/2410.17725

[9] Jordan M. Malof, Kyle Bradbury, Leslie M. Collins, and Richard G. Newell. 2016. Automatic detection of solar photovoltaic arrays in high resolution aerial imagery. *Applied Energy* 183 (2016), 229–240. doi:10.1016/j.apenergy.2016.08.191

[10] Stadt Heidelberg. 2025. Heidelberg in Zahlen. https://www.heidelberg.de/HD/Rathaus/Heidelberg+in+Zahlen.html. Zugriff am 13. April 2025.

[11] State Office for Geoinformation and Land Development (LGL). 2024. Open GeoData Portal. https://opengeodata.lgl-bw.de/ Accessed: Nov 15, 2024.

[12] Ultralytics. 2025. Ultralytics YOLO11. https://github.com/ultralytics/ultralytics/blob/main/README.md Accessed: Feb 02, 2025.

[13] Dhanasingh Sivalinga Vijayan, Eugeniusz Koda, Arvindan Sivasuriyan, Jan Winkler, Parthiban Devarajan, Ramamoorthy Sanjay Kumar, Aleksandra Jakimiuk, Piotr Osinski, Anna Podlasek, and Magdalena Daria Vaverková. 2023. Advancements in solar panel technology in civil engineering for revolutionizing renewable energy solutions—a review. *Energies* 16, 18 (2023), 6579.

[14] Matthias Zech and Joseph Ranalli. 2020. Predicting PV Areas on Aerial Images with Deep Learning. In *47th IEEE Photovoltaic Specialists Conference, PVSC 2020*. https://elib.dlr.de/148270/

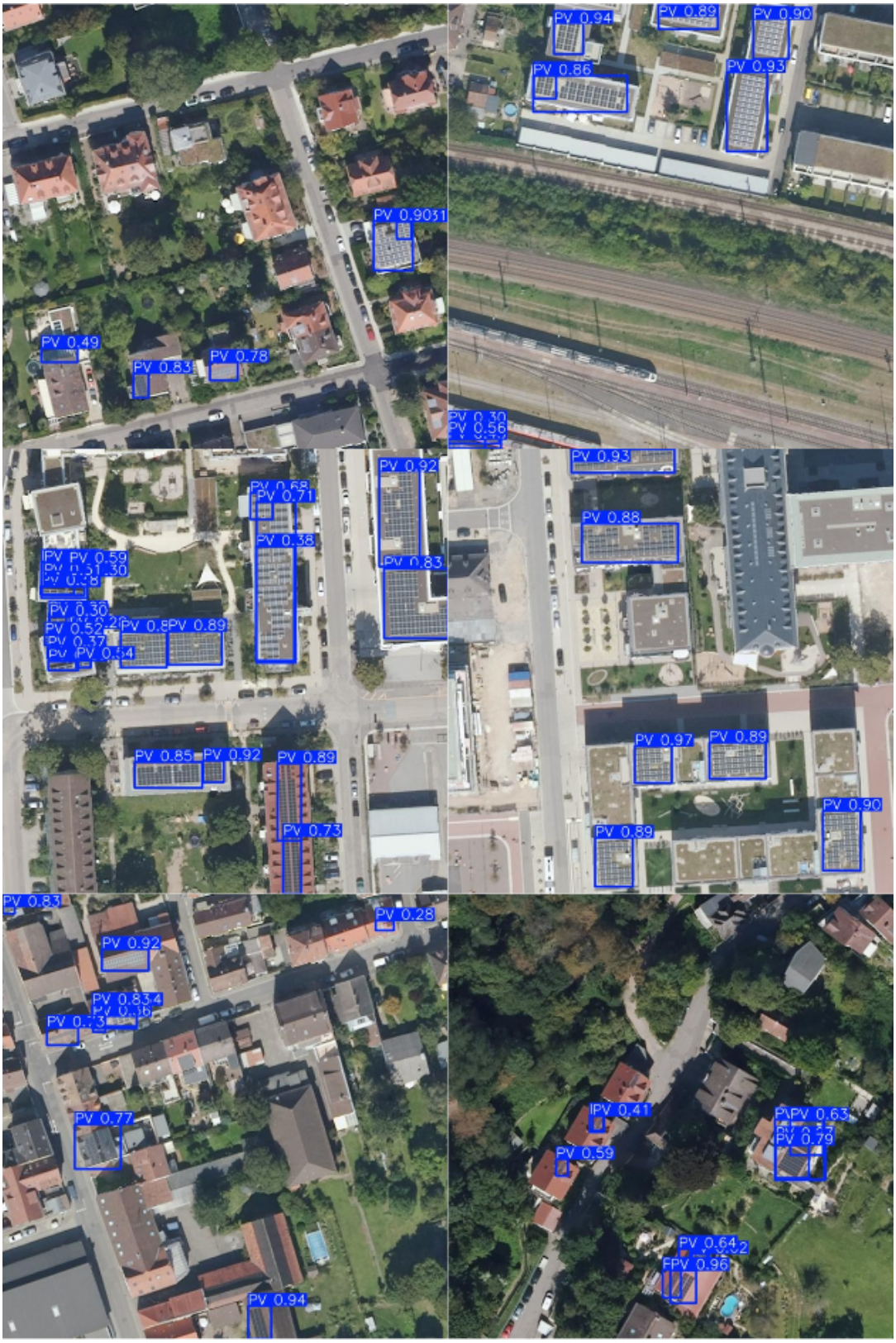


Figure 10: Example results from test run. Data source: LGL, www.lgl-bw.de. Coordinates (from top to bottom), left column: 49°25'05.9"N 8°40'44.3"E; 49°23'22.4"N 8°40'58.2"E; 49°22'44.6"N 8°39'55.6"E; right column: 49°24'32.8"N 8°39'21.9"E; 49°23'14.2"N 8°40'58.2"E; 49°22'44.9"N 8°42'01.6"E.

Building Detection Using Satellite Imagery in South India

Luis Zwießele
Wiktoria Polotzek
Mohamed Raaidh

Abstract

New ways to map and measure human settlements through building detection address a wide range of questions related to urbanization and sustainability. Especially in developing countries where rural settlements still thrive and manual record keeping poses a challenge. We use open and freely available Sentinel-2 satellite imagery on part of South India to test the effectiveness of deep learning to detect buildings and settlements. In this paper, we tested with both manually labeled models and an external label model with different parameters. Our results indicate that the manually labeled model performs better compared to the model that uses external labels. However, manual labeling is time and resource constrained, which limits the area of coverage.

Keywords

Building Detection, Satellite, Manual Labeling, Automated Labeling, Object Detection, YOLOv8, South India, Tamil Nadu

1 Introduction

Building detection through remote sensing has many practical applications in geography, economics, and social sciences. These include mapping land use, managing environmental resources, building damage related to natural disasters, and urban planning [4]. The advancement in satellite remote sensing technology has revolutionized the approaches to monitoring the Earth's Surface and the information we can gather. In this paper, we use satellite imagery and Open Building data labels to test the effectiveness of deep learning to detect buildings in Southern India.

While an increasing number of people are moving into cities, in developing countries such as India, some communities still live in remote places. There are many reasons why these small communities have survived despite rapid urbanization. However, many remote communities are becoming smaller compared to before. As such, it is important to monitor their size and the services available to these communities. Knowing where people live becomes increasingly important when we consider natural disasters [6] [8] and health-related emergencies.

Building detection from remote sensing data has a number of unique challenges, mainly related to the availability of high-quality data, especially for developing countries. Since 2014, remote sensing research has shifted towards deep learning to address these challenges. Now, deep learning is used in a variety of applications such as object detection, change detection, image fusion, and image classification [2].

Traditionally, building detection tasks uses high resolution satellite images [1]. Using low resolution requires applying different augmentation techniques to improve the accuracy. For instance, Sirko and coauthors [7] use 50cm high-resolution imagery as a teacher model and train a student model of 10m low-resolution Sentinel-2 images to reconstruct the labels from the corresponding places. They find the student model retains most of the accuracy of the high-resolution teacher model. Sentinel-2 images based building segmentation has a 79.0 percent mIoU, compared to 85.5 percent mIoU from the high-resolution model. A similar approach was used by Zhang et al. 2021[10].

Additionally, Corbane et al. (2020) [2] presents an overview of using Convolutional Neural Networks (CNNs) for global settlement detection with Sentinel-2 imagery. The study details their methodology for generating a global built-up area map, represented as a probability grid. One of the primary challenges in working with a global dataset is the substantial volume of training data needed to optimize network parameters. To address this, the authors leverage multiple datasets, including the Global Human Settlement Layer, the European Settlement Map, Facebook's high-resolution settlement data, and Microsoft's building footprints. The paper provides an overview of all of the footprint data available in different regions and limitations, which was helpful in extending our model.

2 Materials and Methods[1]

2.1 Study Area

India has a vast territory with a wide range of urban and remote settlements. In this study, we selected a region in South India that includes large cities like Chennai, a few medium-sized cities and many remote villages in between. To cover this diversity, we selected two regions that cover vast parts of Tamil Nadu: The first area covers a bounding box from 79 lon to 89.35 lon and 12 lat to 13.35 lat and includes Chennai and the surrounding area. The second one includes Salem and Tiruchirappalli and stretches from 78 lon to 79.35 lon and 10.65 lat to 12.0 lat. In our first model, we only used the first area since we had to label them manually, but for the second model, we cover both regions. Together they cover about 44,080.86 square kilometers.

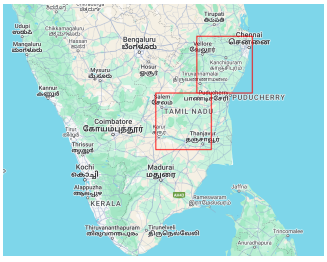


Figure 1: Satellite imagery of the study area: Polygon 1: 79.0, 12.0, 80.35, 13.35, Polygon 2: 78.0, 10.65, 79.35, 12.0. Image © 2025 Google Earth Engine

[1]Scripts and code available at: https://github.com/WikiPol/GeoAI_and_DL_Seminar_Uni_HD_4/tree/main

2.2 Materials

2.2.1 Satellite Imagery. We used Sentinel-2 data from the Copernicus mission downloaded through Google Earth Engine for this project. The Sentinel-2 mission consists of two satellites, Sentinel-2A and Sentinel-2B, with a 180° phase difference, each of which has a total of 13 bands, with a resolution of 10m, 20m, or 60m. A single satellite can complete the revisit within 10 days, so it has a wide range of applications in many fields such as land observation and change detection.

For our project, we use the RGB band (B4, B3, B2) with a resolution of 10m as shown in Figure 2. In order to download them in bulk and in the specified region, we used a Python script. In this Python script, we set a region and downloaded the TIF data within. Because YOLOv8 needs images with a maximum size of 640x640, we cut the region into tiles of this format. The download was executed using Google Earth Engine. This approach allowed us to download the images autonomously and seamlessly upload them to our Google Drive, where we could continue working with them. Figure 2 shows examples of the 10m resolution images we used for this project.

We chose Google Colab to process the data further. We split the TIF data into even smaller tiles (around 256x256) because, firstly, the 640x640 tiles we downloaded were not 640x640 when we transformed them to jpg. Secondly, because we had more labels, the images would have had way too many labels on each image, and with the smaller size, it was feasible.

After splitting, we converted the TIF data to jpg. In our first model, we did the labeling by hand to see if this would give good results as well, even though it is difficult to see separate buildings on the satellite data.

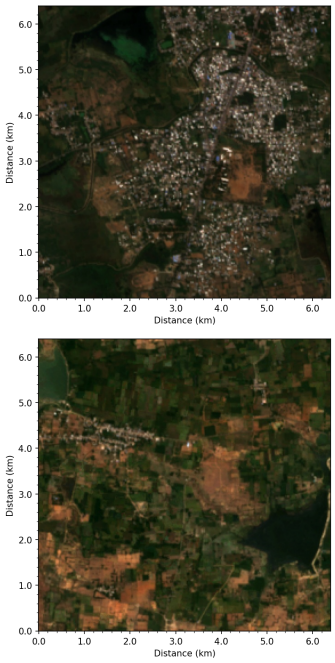


Figure 2: Examples of Sentinel-2 imagery of the study area. Image © 2025 Google Earth Engine

2.2.2 Mask Data. We used Google Open Data Collab Notebook to download the Open Data building footprints for the two previously specified areas. We used a Google Colab of our own to automate the labeling for our images. We uploaded the Google Open Data and referenced it to our own TIF data to get the labels for each of our tiles. We created our own labelmap since we only need one class. The labels are detailed to the point that it even hurts the training of the model, which we could not fix, but more about that is discussed in the designated chapter.

Given that our automatically labeled dataset consists of approximately 10,000 images, we employed a script to upload the images via the Roboflow API to efficiently manage the large volume of data. We selected Roboflow for its comprehensive analytics capabilities, which provided valuable insights into our dataset. Furthermore, Roboflow enables the creation of custom datasets with a range of preprocessing and augmentation options. For our first training attempt on the dataset, we resized all images to 256×256 pixels. For the second training attempt, we resized the images to 512×512 pixels. To preserve image quality despite the rescaling, we downloaded higher resolution input data in the form of 1024×1024 TIFF files. We additionally implemented a script to condense the labels by merging small bounding boxes with their immediate neighbors when they were close together. This adjustment aimed to improve detection performance, as YOLO struggles to detect small objects, especially when image resolution is limited. As part of our preprocessing pipeline, we applied brightness augmentation to enhance the model's fine-tuning process to both attempts.

2.3 Methods

After the labeling process, both models are trained in the same way using the same 70 percent for training, 20 percent for validation and 10 percent for testing split. Given the computational capabilities, we used pretrained Ultralytics YOLOv8 [5] with Google Collab framework. YOLO is popular for object detection and it is generally considered fast and accurate as it is a one stage detector compared to two-stage detectors. YOLO takes an input image and first divides it into a grid. After that, it calculates bounding boxes and class probabilities for each cell. YOLO outputs detected objects (eg: people, animals, cars etc) in a single forward pass through the CNN. Since its introduction in 2016, there have been many different versions of YOLO. We used the YOLOv8 since it performed better compared to the YOLOv7 from the previous year. As shown in Figure 3, our models diverge in the labeling process, but after that, follow the same path for all variations.

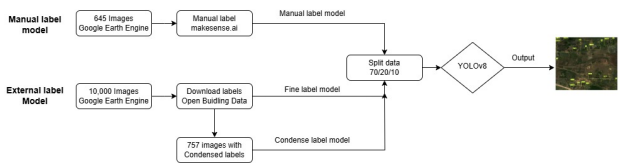


Figure 3: Workflow chart

During the training phase for the automatic labels data, we observed that the external labels were often too small and densely clustered, which significantly obstructed the model's ability to learn

meaningful patterns. As a result, the models trained on this data performed poorly, achieving a maximum precision of only around 1 percent. To address this issue, we developed a script to refine the labels by merging those that were close together into larger bounding boxes. This adjustment aimed to provide clearer object representations for the model. The refined labeling process was applied to a preprocessed version of the dataset from Roboflow, which then served as the basis for subsequent training experiments.

3 Results

In this section, we will report the results of our experiment. Three common metrics are used in object detection to evaluate the performance of a model. They are Recall, Precision, and we additionally show mean average precision (mAP). These metrics are based on True Positive, False Positive, False Negative, and True Negative.

3.1 Manual label model

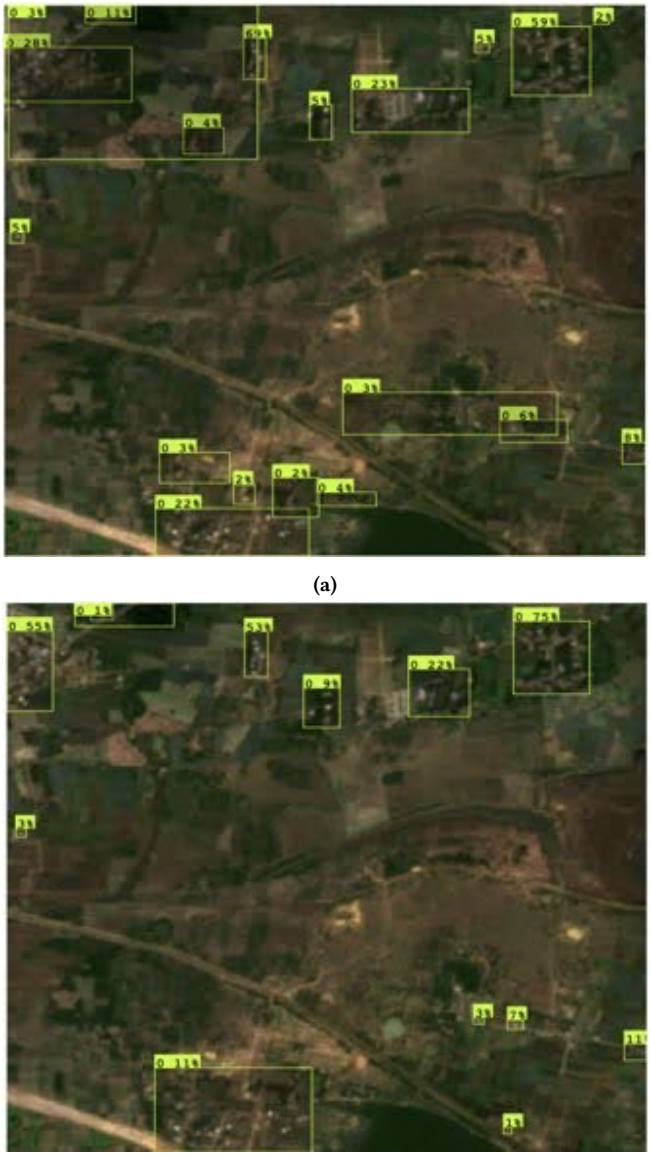
The base model with manual labels contains 646 images from the 1st polygon since manually labeling two polygons was not feasible. We trained the data using different benchmarks. First, with 25 epochs and learning rate of 0.01. As shown in Figure 4a model can detect the groups of buildings, although with less confidence. To improve the accuracy of the bounding boxes, we increased epochs to 30 and decreased the learning rate to 0.001. This led to a slight improvement in our results. As we can see in Figure 4b, bounding boxes are more accuarte compared to Figure 4a. Table 1 provides the key metrics.

Table 1: Performance Metrics of Manual label models at Different Epochs and Learning Rates (percent)

Configuration	mAP	Precision	Recall
Epochs 25, learning rate 0.01	19.3	27.2	29.2
Epochs 30, learning rate 0.001	26.7	34.8	33.0

3.2 External label model

External labels from Open Building data enabled us to cover a wider area compared to the manual label method. In this model, we tested with two different versions of the Building labels. One model was trained with a fine-label dataset with very granular labels that cover the two polygons with 10,000 images. We also trained another model using condensed labels by merging labels that were close together into larger bounding boxes to test how results would vary between the manually labeled model and the fine-labeled model. We used 25 Epochs for the finely labeled model, but increased the Epochs to 100 for the condensed label model to improve our results. Figure 5a shows an example of the results we obtained using fine labels. We can observe a significant reduction in the performance of the model compared to the manually labeled model in Figure 4a or Figure 4b. As reported in Table 2, our results are considerably lower compared to the manual label model. Using condensed labels did improve the results a little, but not significantly, example output is shown in Figure 5b.



(a)

(b)

Figure 4: Exemplary output for manual label model: 4a Epochs 25, learning rate 0.01, 4b Epochs 30, learning rate 0.001. Coordinates: ((79.9249, 13.1419), (79.9479, 13.1649))

Table 2: Performance Metrics for External Label Models (percent)

Configuration	mAP	Precision	Recall
Fine labels	1.3	1.8	0.1
Condense labels	7.9	12.6	6.2

(a)

(b)

Figure 5: Exemplary output for external label model: 5a Fine labels, 5b Condensed labels. Coordinates: ((79.9249, 13.1419), (79.9479, 13.1649))

4 Discussion

Both the manual label model and the external label model improve when we change the epochs and the learning rates. In the case of the manual label model, when the epochs are set to 30 and the learning rate to 0.001, Precision improved by 7.6 percent and Recall by 3.8 percent. Moreover, from Figure 4a and 4b, we can observe a noticeable improvement in the bounding boxes. They are more accurate around the settlements when epochs are increased and the learning rate is decreased. We tried other variations of epochs, but the results were similar. We believe that more extensive pre-processing of the images could improve the results, similar to findings from other studies [9]. Additionally, our sample for the manual labeled model only contains 646 images, which is divided into 70:20:10 ratio for training, validation, and testing. The results could be improved with a larger sample.

In order to overcome the challenges posed by manual labeling, we used the Open Building dataset with a larger area. We expected our metrics to improve with the increase in the number of images and the accurate labels. However, we observed a decline in key metrics. Both the fine label model (Figure 5a) and the condensed label model (Figure 5b) performed worse compared to the manually labeled model (Figure 4a and 4b). One explanation for the decline in metrics may be that the very fine-grained labels were not compatible with our low-resolution data. Footprints are georeferenced, but the satellite imagery might be misaligned, leading the bounding boxes to not match visible features. Furthermore, labels may be outdated with regard to new buildings, or demolished buildings might not have been updated. Similar results have been reported in other studies that uses other label datasets, such as OSM data labels [3].

The models we used for this project have some limitations. One obvious limitation is the dataset size we used in both models. For the first model, manually labeling images was very time-consuming. This resulted in a small dataset of 646 images. Manual labeling also leaves a lot of room for human error compared to using a dataset that has been established and tested through different sources. Especially considering the low resolution. Additionally, many studies that use Sentinel-2 data go through an extended image augmentation to resolve the problem of low resolution, which was difficult given the time and resource constraints. As mentioned previously, using building footprint data did help us cover more area, but it did come with its challenges; we found it to be less reliable than the manual label dataset. Using other label data could lead to different results.

5 Conclusion

This project aims to realize efficient methods for building detection in remote regions where high-resolution data is difficult and costly to obtain. We used Sentinel-2 data with 10m resolution for the object detection task with both manual and external labels. Our results show that manual labels outperform the external label models. Our study also highlights the importance of pre-processing the low-resolution images. Due to time and resource constraints, we performed basic augmentation, but the literature includes many novel methods to improve the quality of the data. If the data is improved in such a way, the external labels could outperform the manual labels, but this will require further testing of the models.

References

[1] Muhammad Aamir, Yi-Fei Pu, Ziaur Rahman, Muhammad Tahir, Hamad Naeem, and Qiang Dai. 2018. A Framework for Automatic Building Detection from

Low-Contrast Satellite Images. *Symmetry* 11, 1 (Dec. 2018), 3. https://doi.org/10.3390/sym11010003

[2] Christina Corbane, Vasileios Syrris, Filip Sabo, Panagiotis Politis, Michele Melchiorri, Martino Pesaresi, Pierre Soille, and Thomas Kemper. 2021. Convolutional neural networks for global human settlements mapping from Sentinel-2 satellite imagery. *Neural Computing and Applications* 33, 12 (June 2021), 6697–6720. https://doi.org/10.1007/s00521-020-05449-7

[3] Haonan Guo, Qian Shi, Andrea Marinoni, Bo Du, and Liangpei Zhang. 2021. Deep building footprint update network: A semi-supervised method for updating existing building footprint from bi-temporal remote sensing images. *Remote Sensing of Environment* 264 (2021), 112589.

[4] Svetlana Illarionova, Dmitrii Shadrin, Islomjon Shukhratov, Ksenia Evteeva, Georgii Popandopulo, Nazar Sotiriadi, Ivan Oseledets, and Evgeny Burnaev. 2023. Benchmark for Building Segmentation on Up-Scaled Sentinel-2 Imagery. *Remote Sensing* 15, 9 (Jan. 2023), 2347. https://doi.org/10.3390/rs15092347 Number: 9 Publisher: Multidisciplinary Digital Publishing Institute.

[5] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. 2023. *Ultralytics YOLOv8*. https://github.com/ultralytics/ultralytics

[6] Ade Febri Sandhini Putri, Wirastuti Widyatmanti, and Deha Agus Umarhadi. 2022. Sentinel-1 and Sentinel-2 data fusion to distinguish building damage level of the 2018 Lombok Earthquake. *Remote Sensing Applications: Society and Environment* 26 (April 2022), 100724. https://doi.org/10.1016/j.rsase.2022.100724

[7] Wojciech Sirko, Emmanuel Asiedu Brempong, Juliana T. C. Marcos, Abigail Annkah, Abel Korme, Mohammed Alewi Hassen, Krishna Sapkota, Tomer Shekel, Abdoulaye Diack, Sella Nevo, Jason Hickey, and John Quinn. 2024. High-Resolution Building and Road Detection from Sentinel-2. https://doi.org/10.48550/arXiv.2310.11622 arXiv:2310.11622 [cs].

[8] Joseph Z. Xu, Wenhan Lu, Zebo Li, Pranav Khaitan, and Valeriya Zaytseva. 2019. Building Damage Detection in Satellite Imagery Using Convolutional Neural Networks. https://doi.org/10.48550/arXiv.1910.06444 arXiv:1910.06444 [cs].

[9] Mohammed J Yousif. 2023. Enhancing the accuracy of image classification using deep learning and preprocessing methods. *Artificial Intelligence & Robotics Development Journal* (2023).

[10] Lixian Zhang, Runmin Dong, Shuai Yuan, Weijia Li, Juepeng Zheng, and Haohuan Fu. 2021. Making Low-Resolution Satellite Images Reborn: A Deep Learning Approach for Super-Resolution Building Extraction. *Remote Sensing* 13, 15 (July 2021), 2872. https://doi.org/10.3390/rs13152872