Automatic detection of Landfills with Sentinel 2 data using YOLO-v5

Brian Ament, Sonya Yuechan Chen, Prakhar Maini, Lalit Vedula, Michael Zhu
UC Berkeley School of Information
{brian.ament, sonyasonya345, prakhar.maini, lalitv1, michael.zhu}@berkeley.edu

Abstract

*Methane is a heavily potent greenhouse gas with 16% contribution towards global greenhouse emissions [9]. Methane is estimated [10] to have ~30x stronger greenhouse effect than CO_2. Thus removing Methane is expected to be more effective at reducing temperature. One of the major sources of Methane are landfills, (17% of Methane emissions in the US in 2019 [11]). With the advent of readily available satellite data through EPA's Copernicus and other programs, there have been some efforts [6, 7] to perform automatic detection of landfills and Methane leaks [8, 17]. However, in our literature survey we found that most of these attempts have either been largely qualitative [7] or didn't utilize the state of the art techniques [6]. In our work, we used Sentinel - 2 data [13] in the visual band and used modern computer vision models [2,3,4] to develop an automated pipeline for landfill detection. We also created a manually tagged set of landfill image data covering 2147 landfills across the globe which can be used for both landfill detection and landfill localization tasks. We created a three step pipeline (image extraction, landfill detection, landfill localization) with landfill detection (using Inception v3) ROC-AUC as **0.934** and landfill localization (using YOLO-v5) mean average precision@0.5 as **0.638**. Our dataset and code can be accessed from https://github.com/UCB MethaneTracker.*

Keywords: Sentinel 2, Object Detection, YOLO-v5, Inception v3, Methane Emissions, Climate change

1 Introduction

Convolutional neural networks have been really successful in recent times in detection of real world phenomenon using satellite imagery. In recent times, they have been tried in many real world contexts (e.g. detecting demolished buildings after natural hazard [14], Inundation assessment of Typhoons [15]). Our motivation was to detect the Methane emissions directly. However, the sparsity of Sentinel-5P methane collection data inhibited our attempts to create a global Methane detection model. Instead we focused our attention to detection of landfills in the wild given their large contributions to Methane emissions and unavailability of structured landfill location data especially in the developing world.

We hypothesize that a system capable of automatically sweeping a large geographical region and localize landfills will be helpful in controlling Methane emissions in two fundamental ways:

I. The resulting localizations can be used as focal points for more advanced remote sensing equipment (e.g. Sentinel-5P) for better estimation of Methane emissions
II. In places which lack solid waste disposal tracking, this system can act as an early detection system. The signals generated by this system can help authorities to manage the waste management practices better and plan for Methane capture

In order to motivate more research, we released our manually tagged dataset which should result in improvement in performance of these systems.

2 Project Overview

2.1 Background

Proposing methods to reduce Methane emissions is the primary motivation for this work. Sentinel 5P consists of a Methane data product [16]. However,

Geospatial coverage of this data is very sparse. At present, Methane concentration data from Sentinel 5 is being used for anecdotal [17] reports of Methane gas leaks but it can't be used effectively to detect Methane emissions across the globe with high fidelity. As landfills are one of the large contributors to methane emissions [11] and organic waste management is typically problematic in developing nations, we developed an algorithm for automatic landfill detection.

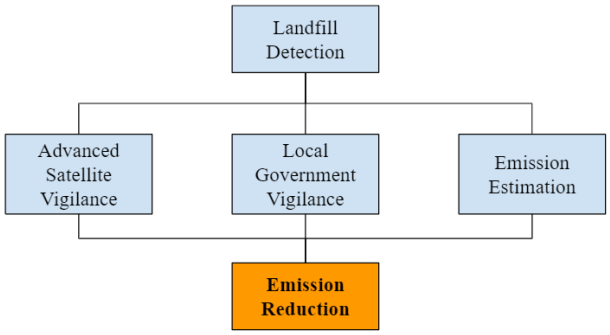


Figure 1: Proposed Action Mechanism

As showcased above, we propose that our algorithm can be used as a first step towards:

- More localized usage of Sentinel 5 based Methane & other GreenHouse Gas (GHG) concentration vigilance
- Vigilance from local government to control
- Estimate Methane and other GHG estimation from landfill sites

We hypothesize that this added vigilance should result in better capture of Methane and consequently significant reduction in Methane emissions from landfills.

2.2 Datasets

In our literature survey, we could not find any curated data sets consisting of Sentinel 2 visual bands focusing on landfill locations. Thus, one of the contributions we make to this problem is the hand curated data set that we generated with landfill images. We started with 2299 US based landfill [18] locations provided by EPA. We augmented this data by adding 507 more landfill locations from around the globe. After further data cleaning and pruning, we were left with 2147 landfill locations which we divided into train, validation and test sets.

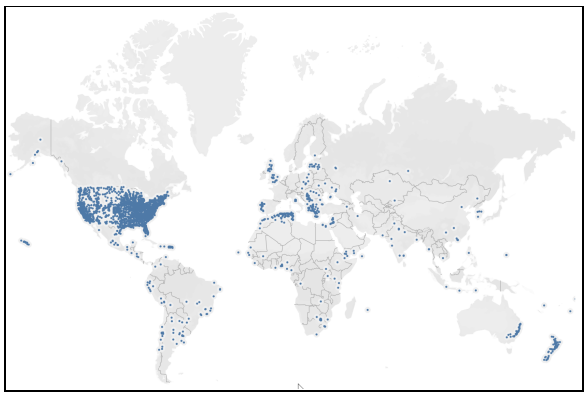


Figure 2: Geospatial distribution of landfill locations in our training, validation and test sets

We further generated negative samples from locations around the landfills by taking a constant offset of 0.4 in both latitude and longitude in both directions. This way we tried to maintain a 1:4 positive to negative ratio across our dataset. After data cleaning and manual validation of images, we ended up with the following full dataset:

Data Type	Positive	Negative	Total
Training	1732	5829	**7561**
Validation	165	501	**666**
Test (Unseen)	250	776	**1026**

Table 1: Distribution of data across Train, Validation and Test

2.3 Data Processing

We have discussed our data collection methodology in section 2.2 where we indicated that we started with 2299 + 507 (= 2806) landfill locations. We used Google Earth Engine API [20] for image extraction. Given the latitude and longitude coordinates of a landfill, we used following filters to extract relevant Sentinel 2 images:

I. We looked at the timeframe between 2020-09-01 and 2021-09-21 for image extraction
II. We focused on Bands B4, B3 and B2 representing red, blue and green colors

III. We focused on a square bounding box of side length 0.024 centered on given latitude and longitude of landfill
IV. We selected the least cloudy available image from the selected time period
V. We clipped the image to get a 512x512 pixel final image

Please refer to the function get_ee_image_S2() in our training data pipeline for a more elaborate look. We added a small perturbation to positive (i.e. with landfill) images so that we can reduce the center bias from our models. In the next step, we manually went through all the landfill images to locate the landfill area using a bounding box [21]. In this step, we faced many challenges and had to perform a lot of data pruning which resulted in 23.5% reduction in the baseline data. Major themes included:

I. Latitude and longitude associated with the landfill didn't generate image with landfill
II. The generated image had a lot of cloud cover thus landfill was not visible

While tagging the bounding boxes, we also indicated the other metadata (e.g. terrain, presence of water body etc.). The distributions across the training + validation and test sets are as following:

Type	Barren	Green Cover	Water body	Residential
Train + Validation	60%	40%	5%	13%
Test	59%	41%	7%	17%

Table 2: Distribution of landfill image metadata

2.4 Approach

Our goal was to have an easy and affordable method to sweep a large land area using satellite imagery and detect landfills. In line with our goal, our approach to landfill detection consists of three major elements:

2.4.1 Automated data extraction & processing

We developed a data pipeline that utilizes Google Earth Engine API [20] and can be utilized to sweep large land masses and extract Sentinel 2 images given latitude and longitude information. It has built-in filters that enable clean data extraction. At present, we support local image downloads but this can easily be changed to a central repository (e.g. Google Drive) for a more scalable solution.

2.4.2 Landfill detection model

In the next step, we used a fine-tuned classification model [33, 34] to detect presence of a landfill in the extracted Sentinel 2 visual image. Using our training data, we obtained a probability threshold that enables us to do binary classification with landfill and no landfill classes. At the end of this step, we are in a position to detect if a landfill exists somewhere in the picture. However, in order to localize the landfill within the image, we move on to the final element of our analytical pipeline (i.e. localization).

2.4.3 Landfill localization model

In the final step, we started with the images that received higher than threshold probability of containing a landfill in the previous step. We run our fine tuned YOLO-v5 [1, 2, 3] model on these images to localize the landfills within the image using a bounding box. This also helps us understand the actual size of the landfill using the size of the resulting bounding box and the geospatial zoom of the base image.

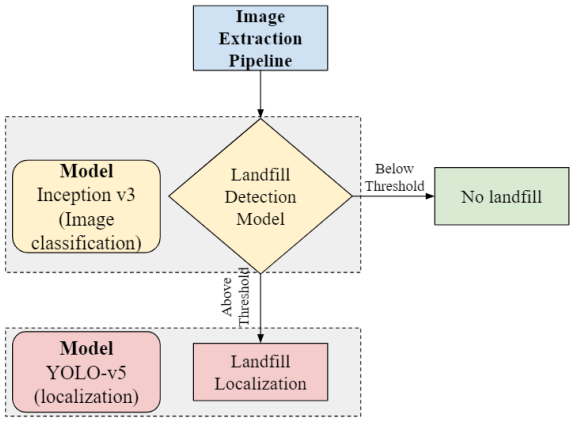


Figure 3: Approach for Landfill Detection & Localization

2.5 Evaluation Method

To evaluate the performance of our classification model, we will look at the ROC-AUC of our fine-tuned

model and F1-score on the unseen test set. Similarly, to evaluate the performance of our localization model, we will look at Average Precision @0.5 on the unseen test set. To further validate the generalizability of our results, we performed a systematic sweep across 10 cities in India and showcased the results of our model in section 4.3.

3 Models

3.1 Landfill Classification Model

Deeper neural networks naturally integrate low, mid, and high level features and classifiers in end-to-end multilayer fashion and the levels of features can be enriched by the number of stacked layers (i.e. depth). This was the intuition behind the first classification model that we tried on our data, VGG-16 [35]. One of the main problems achieving convergence in deep networks are:

- Exploding / vanishing gradient
- Saturation and degradation of accuracy

The exploding and vanishing gradient problem is addressed by either normalized initialization [23, 24] or Batch Normalization [25]. In order to solve the degradation problem, He et. al. (2015) proposed [4, 5] a deep residual learning framework (as noted below)

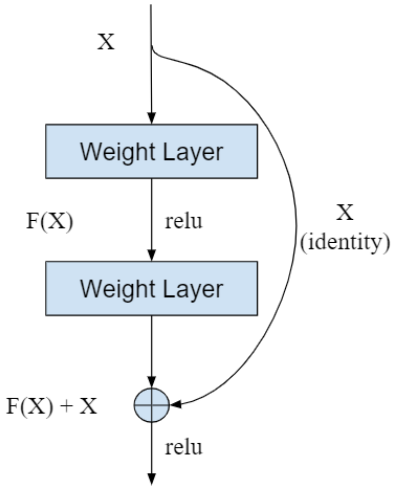


Figure 4: Residual Learning: a building block

The authors hypothesized that it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping. Formally, denoting the underlying mapping as $\mathcal{H}(x)$, we let stacked nonlinear layers fit another mapping $\mathcal{F}(x) := \mathcal{H}(x) - x$. The original mapping is recast into $\mathcal{F}(x) + x$ which is realized by feedforward neural networks with shortcut connections. This constitutes the second broad network category we experimented with for our classification problem (i.e. Resnet 50).

Finally, auxiliary classifiers introduced with Inception v3 [34] are another interesting way to push gradients to the lower layers of deep networks and improve convergence. This creates the core of our third category of classification models (i.e. Inception v3). Introduced in 2016, Inception v3 is 42 layers deep and is still much more efficient than VGG-16 due to the improved architecture and focus on reducing representational bottlenecks using factorization, label smoothing and efficient grid size reduction. Our experiments on landfill classification resulted in Inception v3 as our champion model (Section 4.1).

3.2 Landfill Localization Model

YOLO was introduced [2] as a unified model for object detection. Unlike classifier-based approaches, YOLO is trained on loss function that directly corresponds to detection performance and the entire model is trained jointly. Many fundamental improvements [26] were proposed by Redmon et. al. which enabled the next iteration YOLO-v2 to outperform state-of-the-art methods (76.5 mAP at 40 FPS) while being one of the fastest models and being able to run at varying sizes. Bochkovskiy et. al. offered YOLO-v4 which is a state- of-the-art detector faster and more accurate [27] than all available alternative detectors. For our localization task, we selected YOLO-v5 [3] which is a PyTorch implementation of YOLO-v4. Many state-of-the-art approaches repurpose classifiers to perform detections. More recent approaches like R-CNN use region proposal methods to first generate bounding boxes in an image and then run a classifier on these proposed boxes. After classification, post processing is used to refine, deduplicate and rescore boxes based on other objects in the scene. YOLO-v5, on the other hand, reframes object detection as a single regression problem thus you only look once at an image to

predict what objects are present and where they are. This means:

- YOLO is very fast because there is no need for complex detection pipelines
- YOLO reasons globally about the image

YOLOv5 consists of three parts [27]:

1. **Backbone**: It uses CSPDarknet53 [28] as the backbone trained on Imagenet
2. **Neck**: It uses SPP [29] as the additional block and PAN [30] as path aggregation block collecting feature maps from different stages
3. **Head**: It uses the YOLO-v3 [31] head for one stage dense prediction

The system divides the input image in a grid (S x S) and for each grid cell predicts B bounding boxes, confidence for those boxes and C class probabilities. Each bounding box consists of five predictions (*x, y, w, h, confidence*). The (*x, y*) represent the center of the box relative to the center of the grid cell. Width (*w*) and height (*h*) are predicted relative to the whole image and confidence represents the IOU (*Intersection Over Union*) between the predicted box and any ground truth label. Overall, predictions are encoded as an S x S x $(B * 5+C)$ tensor. Bounding boxes are predicted using dimension clusters as anchor boxes. To improve the priors to facilitate quick learning, YOLO-v5 runs k-means clustering (k=5) on training set bounding boxes with distance metric ***d(box, centroid) = 1 - IOU(box, centroid)*** as our goal is to have priors that lead to good IOU scores.

4 Results

4.1 Discussion of classification results

For landfill classification we tried three pre-trained convolutional neural networks, VGG-16, ResNet-50, and Inception v3. In each case we used the base model without the top classification layer, but instead added a fully connected layer with ReLU activation, a 20% dropout layer, and a final sigmoid activation layer for landfill prediction. The initial weights in each model were imported from ImageNet. The input images had size 512x512x3. The training set consisted of 7561 images while the validation set had 666 images. Both the VGG and ResNet models had good landfill Precision, but both struggled with landfill Recall, as each had a score of 0.22. The Inception model, however, had good landfill Precision and Recall, resulting in an f1-score of 0.63 and an accuracy of 83% on the validation data.

Using the Inception v3 model, we explored the learning rate, optimizer, and the number of epochs. Larger learning rates led to over-fitting of the training data. The best result with the validation data occurred with a learning rate of 5E-7. Three optimizers were tested, Adam, RMSprop, and SGD. The first two were much better than the last one while Adam was slightly better than RMSprop. Lastly, we explored the number of epochs. As expected, an increased number of epochs resulted in better performance, but 90 epochs was only minimally better than 60 epochs. Using this final model on the *unseen test set*, we observed **0.934** ROC-AUC and **79%** F1-score. For detailed discussion, please refer to *Appendix A1*.

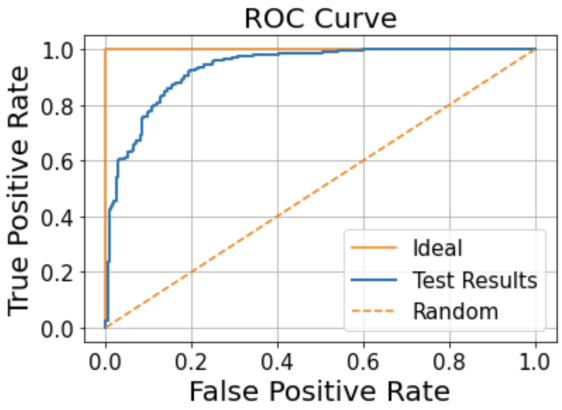


Figure 5: ROC curve indicating our final model performance on the unseen test set (**ROC-AUC = 0.934**)

4.2 Discussion of localization results

We used YOLO-v5 to detect bounding boxes around the landfills. We completed six iterations, each with a different combination of technique and model size. The input image size was 512 x 512 for our model training. Early in the model exploration, we found that the Adam optimizer performed better than the SGD optimizer with smaller datasets. We also discovered that the results improved when we

unfroze the backbone. For instance, for the mid sized model the best mAP increased from 0.625 to 0.638 when we unfroze the backbone while holding all other parameters the same (5th versus 6th iteration).

In our training iterations, we realized that predicted images could have several bounding boxes whereas in our labeled training dataset, we restricted labeling to a single bounding box. To resolve this disparity, we restricted the prediction to a single bounding box with highest confidence. According to a discussion on Github [32], NMS stands for non_max_suppression. Parameter max_nms sets the maximum number of bounding boxes. So we set max_nms to 1 to limit the number of bounding boxes to 1 in file yolov5/utils/general.py. With max_nms, the small size YOLO-v5 best mAP@.5 increased from 0.201 to 0.259. Finally, we shifted from using a small size YOLO-v5 to a mid size model and noticed that the best mAP increased significantly. Best mAP@0.5 increased from 0.259 to 0.625 when changing from small size model (3rd iteration) to mid size model (5th iteration) while holding all other factors the same. The detection model with best mAP@0.5 is middle size YOLO-v5 with max_nms=1, unfrozen backbone, Adam optimizer, which has mAP@.5 of **0.638** (5th iteration). For detailed results, please refer to the Appendix (A2).

4.3 Testing for model generality

As we created the negative images in our classification dataset by choosing 4 images at a constant offset from the base landfill image and most of the landfills are from the United States, it is very likely that our dataset is biased and our resulting models don't generalize well to other parts of the world and terrains. In order to test the generalizability of our approach, we started with a broad geographical survey of 10 cities in India (not included in our initial data survey). Our survey resulted in 2,561 images and the model predicted 138 of the images to contain landfill. Manual inspection suggested good precision among predicted images. The model seems to generalize well given model performance. More work is needed to ensure generalizability of our approach but the initial results are promising. The list of 10 cities from India (part of our generalization study) can be referred to in Appendix (A4).

Figure 6: Example of landfill detected by our pipeline in Hyderabad, India.

4.4 Discussion of Errors

Overall, YOLO-v5 did a better job in detecting landfill than we expected with only 3 channel images (R, G, B). Going through the detection results, we noticed that two types of images did not perform well. Majority of the cases that had either no bounding box predictions or mis-labeled boxes are ones that are hard to tell even for the human eye from the satellite images. Full results can be seen here [36]. Some comparisons of the predicted vs labeled images are noted in Appendix (A3).

4.5 Limitations & Future Work

4.5.1 Sampling of negative images

In order to generate the negative sample images for the classification model, we used a constant offset of 0.4 and applied it to the latitude and longitude of the landfill. Our rationale was that such a large shift will ensure that landfill is not present in the negative images however the overall terrain will be similar to that of the landfill. This choice introduces a bias in our training data and as a potential solution, we can add randomly sampled images from landmass across continents to reduce bias and improve generalizability.

4.5.2 Restricting on only visual bands (B4, B3, B2)

Sentinel-2 provides us with 13 bands of data [22] in the VNIR and SWIR range. For simplicity, we utilized the visual bands B2, B3 & B4 in our pipeline. There might be scope for significant improvement if we are able to synthesize more information using the remai-

ning 10 bands especially focusing on including infrared region data.

4.5.3 Single bounding box prediction

To keep the manual tagging consistent, we chose to use a single bounding box per landfill image. However, we noticed that in many cases, the landfills were clearly segmented geographically. In order to remain consistent in our modeling, we restricted the YOLO-v5 to produce a single bounding box per prediction. In the future iterations, we may look to remove this constraint.

5 Conclusions

In this paper, we explored the problem of landfill detection using satellite imagery. We showcased that using the Inception v3 model, we can detect the presence of landfills in the Sentinel - 2 visual image with high F1-Score (**0.79**) on the unseen test set. In addition, we also showed that using the YOLO-v5 model, we can further improve upon our output by localizing the landfills in the image and getting a sense of their size and position (mAP@0.5 **0.638**). Additionally, we created a manually tagged gold standard dataset that can be used by future researchers to improve upon our work. Overall, our key motivation was to be able to make it easy and affordable to sweep a large area via satellite imagery and detect the Methane sources such as landfills. We demonstrated how to achieve this goal via our end-to-end machine learning pipeline.

6 Acknowledgements

We thank our instructors Alberto Todeschini and Fred Nugen for motivating us to work on this problem. We thank Colorado Reed for helping us debug our classification model and providing valuable feedback on running experiments. We thank Daniel Cusworth from NASA Jet Propulsion Laboratory for helping us narrow down our problem statement and providing valuable guidance early in the project.

References

[1] M. B. Blaschko and C. H. Lampert. Learning to localize objects with structured output regression. In Computer Vision–ECCV 2008, pages 2–15. Springer, 2008

[2] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). "You only look once: Unified, real-time object detection," in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 779-788. DOI: 10.1109/CVPR.2016.91

[3] Ultralytics-Yolov5. Available online: https://github.com/ultralytics/yolov5

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. CVPR, 2016.

[5] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2):157–166, 1994.

[6] Skogsmo, M. (2020), “A Scalable Approach for Detecting Dumpsites using Automatic Target Recognition with Feature Selection and SVM through Satellite Imagery” Master's Thesis, Uppsala University.

[7] E. G. Cadau, G. Laneve , G. Vingione , G. Palumbo (2017), “COPERNICUS SENTINELS EO-DATA FOR URBAN LANDFILLS DETECTION AND MONITORING”. ISRSE 2017 - TSHWANE

[8] https://www.esa.int/Applications/Observing_the_Earth/Satellites_detect_large_methane_emissions_from_Madrid_landfills

[9] https://www.epa.gov/ghgemissions/global-greenhouse-gas-emissions-data

[10] https://www.epa.gov/ghgemissions/understanding-global-warming-potentials

[11] https://www.epa.gov/ghgemissions/overview-greenhouse-gases

[12] Xu, R.; Lin, H.; Lu, K.; Cao, L.; Liu, Y. A Forest Fire Detection System Based on Ensemble Learning. Forests 2021, 12, 217. https://doi.org/10.3390/f12020217

[13] https://sentinel.esa.int/web/sentinel/missions/sentinel-2

[14] Vahid Rashidian, Laurie G. Baise, Magaly Koch and Babak Moaveni, Detecting Demolished Buildings after a Natural Hazard Using High Resolution RGB Satellite Imagery and Modified U-Net Convolutional Neural Networks. Remote *Sens.* 2021, *13*(11), 2176; https://doi.org/10.3390/rs13112176

[15] Liu W., Fuji K., Maruyama Y. and Yamazaki F. Inundation Assessment of the 2019 Typhoon Hagibis

in Japan Using Multi-Temporal Sentinel-1 Intensity Images. Remote *Sens.* 2021, *13*(4), 639; https://doi.org/10.3390/rs13040639
[16] https://sentinels.copernicus.eu/web/sentinel/missions/sentinel-5/data-products
[17] https://www.themoscowtimes.com/2021/11/01/satellites-detect-massive-russia-methane-leak-bloomberg-a75410
[18] https://www.epa.gov/lmop/project-and-landfill-data-state
[20] https://developers.google.com/earth-engine
[21] https://www.makesense.ai/
[22] https://sentinels.copernicus.eu/documents/247904/ 685211/Sentinel-2_User_Handbook
[23] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Muller. Efficient backprop. ¨ In Neural Networks: Tricks of the Trade, pages 9–50. Springer, 1998
[24] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In AISTATS, 2010.
[25] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In ICML, 2015.
[26] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, pages 6517–6525. IEEE, 2017.
[27] Alexey Bochkovskiy, Chien-Yao Wang, and HongYuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934, 2020.
[28] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. CSPNet: A new backbone that can enhance learning capability of cnn. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshop (CVPR Workshop), 2020.
[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 37(9):1904–1916, 2015.
[30] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 8759–8768, 2018
[31] Joseph Redmon and Ali Farhadi. YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767, 2018.
[32] https://github.com/ultralytics/yolov5/discussions/2248
[33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In CVPR, 2015.
[34] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In CVPR, 2016.
[35] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014
[36] https://github.com/UCBMethaneTracker/UCB-Capstone-Methane-Tracker/tree/main/yolo_v5/yolov5/runs/train/iteration5_midModel_adam_imageSize512_unfreezeBackbone_maxNMS1

Appendix

A1. Classification model results

Base models (All results on validation data with frozen backbone, no augmentation, 30 epochs, and prediction threshold of 0.5)

Model	Precision	Recall	F1 - score	ROC-AUC	Accuracy
Resnet50	0.53	0.22	0.31	0.758	0.76
VGG16	0.93	0.22	0.36	0.797	0.80
Inception v3	0.68	0.58	0.63	0.851	0.83

Fine Tuning Learning Rate with Inception v3 Model (All results on validation data with frozen backbone, no image augmentation, 30 epochs, RMSprop optimizer, and prediction threshold of 0.5)

Learning Rate	Precision	Recall	F1 - score	ROC-AUC	Accuracy
1E-3	0.64	0.53	0.58	0.802	0.81
1E-5	0.67	0.58	0.62	0.830	0.83
1E-6	0.76	0.49	0.60	0.852	0.84
5E-7	0.84	0.42	0.56	0.860	0.84
1E-7	0.76	0.42	0.60	0.861	0.82

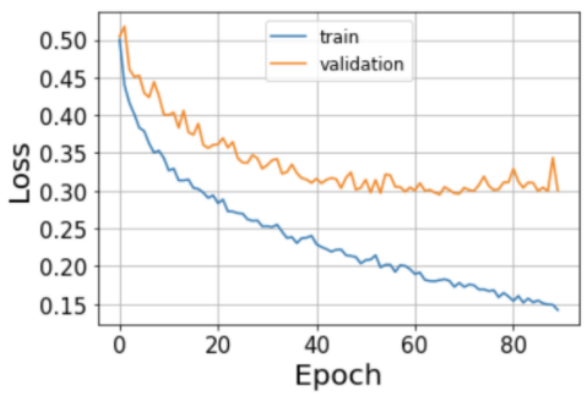
Fine Tuning Inceptionv3 Model (All results on validation data with learning rate of **5E-7**)

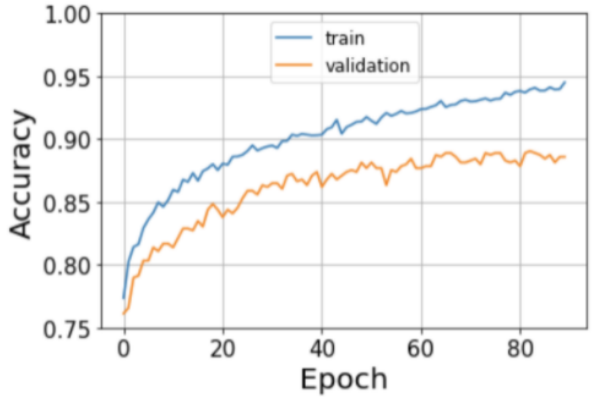
Epochs	Image Augmentation	Frozen Backbone?	Optimizer	Prediction Threshold	Precision	Recall	F1 - score	ROC-AUC	Accuracy
30	No	Yes	SGD	0.5	0.58	0.07	0.12	0.604	0.76
30	No	Yes	RMSprop	0.5	0.79	0.47	0.59	0.857	0.84
30	No	Yes	Adam	0.5	0.75	0.48	0.59	0.873	0.83
30	Yes	No	Adam	0.5	0.78	0.61	0.69	0.899	0.86
60	Yes	No	Adam	0.5	0.85	0.61	0.71	0.920	0.88
90	Yes	No	Adam	0.5	0.82	**0.69**	0.75	**0.922**	0.89
90	Yes	No	Adam	0.366757	**0.87**	0.67	**0.76**	**0.922**	0.89

Performance of the final classification model on unseen test set

Class	Precision	Recall	F1-Score	Support
Positive	0.78	0.80	**0.79**	250
Negative	0.93	0.93	0.93	776

Loss and accuracy curves of the final classification model on validation dataset



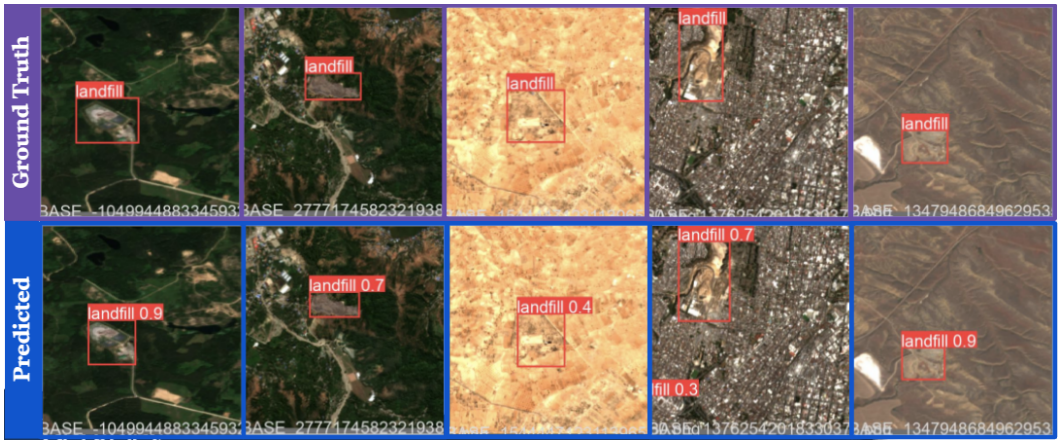


A2. Validation model results

Iteration	max_nms=1	Freeze Backbone (10x)	Best mAP @.5	Best mAP@ .5:.95	Best Precision	Best Recall	Model / Optimizer	Epochs	Image Size	Folder
1	No	Yes	0.201	0.0573	0.273	0.297	Small/Adam	115/300	512 x 512	iteration1_smallModel_adam_imageSize512_freezeBackbone
2	Yes	Yes	0.259	0.0997	0.35	0.319	Small/Adam	122/300	512 x 512	iteration2_smallModel_adam_imageSize512_freezeBackbone_maxNMS1
3	Yes	No	0.584	0.332	0.628	0.563	Small/Adam	249/300	512 x 512	iteration3_smallModel_adam_imageSize512_unfreezeBackbone_maxNMS1
4	Yes	Yes	0.625	0.351	0.613	0.6	Mid/Adam	173/300	512 x 512	iteration4_midModel_adam_imageSize512_freezeBackbone_maxNMS1
5	Yes	No	**0.638**	0.318	0.718	0.585	Mid/Adam	238/300	512 x 512	iteration5_midModel_adam_imageSize512_unfreezeBackbone_maxNMS1

A3. Landfill Localization Results (Examples)

Localization Performance on the validation set (Examples)



Localization performance on the unseen test set (Examples)

A4. List of indian cities included in generalizability study

City	(Latitude, Longitude)
Delhi	28.66,77.23
Mumbai	18.97, 72.83
Kolkata	22.54, 88.34
Bangalore	12.97, 77.60
Chennai	13.08, 80.28
Hyderabad	17.37, 78.47
Pune	18.51, 73.85
Ahmedabad	23.03, 72.58
Surat	21.17, 72.83
Lucknow	26.85, 80.95