

GEOWATCH FOR DETECTING HEAVY CONSTRUCTION IN HETEROGENEOUS TIME SERIES OF SATELLITE IMAGES

Jon Crall, Connor Greenwell, David Joy, Matthew Leotta, Aashish Chaudhary, Anthony Hoogs

Kitware, Inc.

ABSTRACT

Learning from multiple sensors is challenging due to spatio-temporal misalignment and differences in resolution and captured spectra. To that end, we introduce *GeoWATCH*, a flexible framework for training models on long sequences of satellite images sourced from multiple sensor platforms, which is designed to handle image classification, activity recognition, object detection, or object tracking tasks. Our system includes a novel partial weight loading mechanism based on sub-graph isomorphism which allows for continually training and modifying a network over many training cycles. This has allowed us to train a lineage of models over a long period of time, which we have observed has improved performance as we adjust configurations while maintaining a core backbone.

Index Terms— Remote Sensing, Multi-Sensor, Segmentation, Partial Weight Loading, Instillation, Continual Learning

1. INTRODUCTION

The deployment of remote sensing models in realistic settings features a number of challenges: sensor noise, full or partial occlusion from shadows and clouds, and data sparsity due to the fixed revisit rates of many sensor platforms. Taken together, these challenges complicate long term Earth monitoring from a single sensor platform as high quality images are received less frequently. A common strategy to increase the number of viable images is to incorporate readings from additional sensor platforms. To overcome spatial, resolution, and spectral misalignment between multiple sensor platforms, harmonized products such as HLS [1] have been introduced. However, this limits training and inference to a fixed set of sensors. In this paper we introduce *GeoWATCH*, a software package for remote sensing computer vision tasks which is designed to handle multiple sensors, long time frames, be robust to missing and noisy data, and more.

While *GeoWATCH* is designed to be a general use platform, it was initially built to solve the IARPA SMART large-scale heavy construction detection task. The goal of the SMART program is to detect, characterize, and monitor anthropogenic or natural processes using a multiple sources of satellite imagery collected over time. The motivating use-case is the detection of heavy construction over a broad area of space and time.

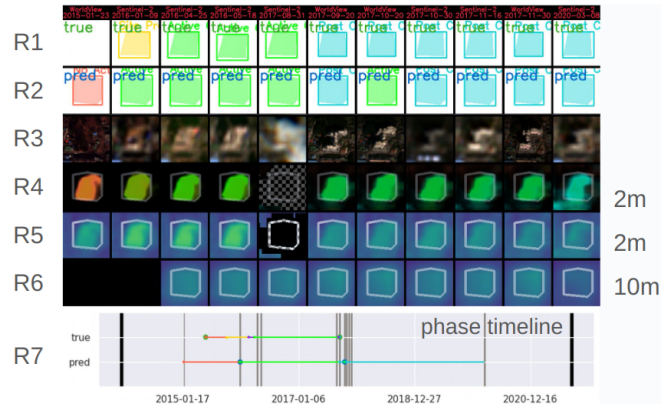


Fig. 1: GeoWATCH Detection Example.

Example prediction for a heavy construction site in the validation dataset. Rows 1 and 2 display true and predicted polygons. Row 3 presents the image data. Rows 4 and 5 feature the 2m GSD phase and saliency heatmaps. Row 6 displays the 10m GSD saliency heatmap. Row 7 compares true and predicted timelines. Category colors include red for “No Activity”, yellow for “Site Preparation”, green for “Active Construction”, and blue for “Post Construction.”

Specifically, we construct a data cube of Landsat, Sentinel-2, and WorldView imagery covering several large spatial regions over a period of almost eight years. The goal is to predict a polygon on construction sites along with a start and end date.

Using *GeoWATCH* our algorithms are designed to make a prediction at every pixel in space time (*i.e.* the output heatmaps are the same resolution as the input images) of 2 heads: “saliency” which indicate if any construction is happening, and “class”, if it is one of 2 phases: “Site Preparation” or “Active Construction”. We then extract polygons from these heatmaps and assign a start and end date. We operate in two stages: (1) broad area search for candidate sites at 10m GSD and (2) activity characterization and validation of candidate sites at 2m GSD. An example of a site detected with our system is shown in Figure 1.

Our contributions are: 1) An extension of MS-COCO [2] called KWCoco, designed to be more suitable for geospatial data, and which provides the ability to sample large multispectral images at a virtual resolution using a novel “Video View”.

2) A new open source framework for training and predicting with AI models on geospatial data with support for continual learning regularization [3], which we call *GeoWATCH*. 3) A method for transferring part of a network to another network with similar structure by finding a maximum subtree embedding / isomorphism [4], termed “Partial Weight Loading”. We present preliminary observations of a phenomena that we call “instillation”, where our models trained with one set of input features retain performance even after that input is removed. and 4) We have publicly released our code¹ via GitLab and model weights via IPFS[5].

2. RELATED WORK

Our GeoWATCH framework is related to other libraries. TorchGEO [6] - Defines specific dataloaders for individual standardized datasets, whereas our system opts to define a data interchange standard and use the same dataloader on all problems. MMSegmentation [7] - is a powerful semantic segmentation library, but cannot handle large images. RasterVision [8] - Is another open source geospatial deep learning library with similar capabilities, but to the best of our knowledge it does not have the ability to produce native resolution batches on time sequences from multiple sensors.

We also explore new techniques related to existing algorithms. Loss-of-plasticity [3] is the observation that networks gradually degenerate while training with SGD, and that selectively re-initializing neurons can mitigate this. In our work we accomplish this re-initialization with partial-weight-loading, wherein the weights of a partially loaded network are reinitialized. Distillation [9, 10] is a technique to transfer knowledge from one model to another. This is done by training a student network to match predictions of a teacher network. In this work we observe a new way to transfer knowledge, which we call “instillation”. This involves a partial weight transfer of a larger network to a smaller network followed by fine-tuning.

3. DATA INTERCHANGE

The main data interchange in our system for vision tasks is a KWCoco file, which is an extension of the well-known MSCOCO format [2] that better handles a datacube of geospatial data. Specifically, COCO images are extended to allow for multi-spectral imagery stored across multiple assets, which could exist at different resolutions. Each COCO image can be registered as a frame in a video. Affine transformations are stored to warp between assets at different resolutions.

To prepare a region for training or prediction, our system runs STAC [11] query and the results are indexed in a KWCoco file, which abstracts away geospatial information and allows computer vision algorithms to reference the data based on a

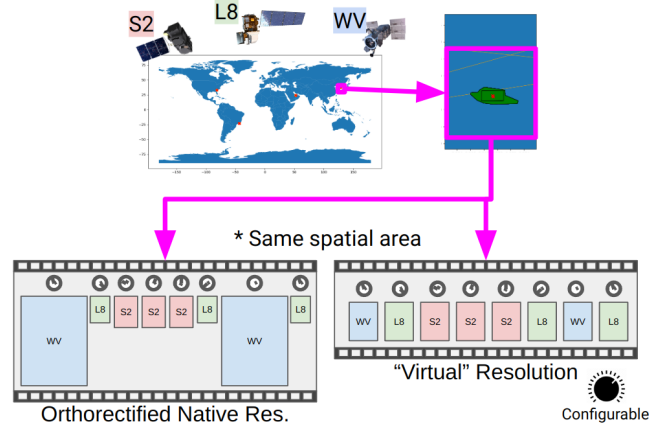


Fig. 2: STAC-to-KWCoco. Given a region of interest, our system runs a STAC query and registers paths to the original images in a KWCoco file. The images are stored natively on disk and we can request heterogeneous subregions of space-time at arbitrary resampled (or native) resolutions.

virtual pixel space where all images area aligned. We call this virtual resolution a **video view**. This process is illustrated in Figure 2.

Once raw GeoTIFFs are available on disk, and indexed by a KWCoco file, we can efficiently access them via the delayed image package [12]. This tool utilizes the COG format [13] to allow efficient access subregions of the cropped images (which might still be quite large — *e.g.* $4K \times 4K$ pixels is common), or coarser resolutions via overviews. It allows the developer to build a tree of image operations (*e.g.* resampling / concatenation) and it is able to optimize this tree by fusing linear operations (*e.g.* affine transforms) and replace downscales by 2 with an overview operation.

4. TRAINING AND INFERENCE

Given a dataset in KWCoco format, a network is trained by specifying an input window and resolution, a time kernel, sensor/channel specification that will be used to construct batches. At creation time, the dataset estimates dataset statistics like mean/std and class frequency and cached. The network also establishes trainable classes, mean/std, and class frequency by receiving them from the dataset. For regularization we have implemented a restricted variant of the generate-and-test algorithm for continual learning [3] as well as the full shrink-and-perturb [14] algorithm. Our networks states are saved as checkpoints, and packaged with `torch.package`, which allows us to bundle the weights with the network topology and other metadata like train-time parameters. All training details are logged in the the package metadata. For network details see previously published work [15].

At test time, we construct a regular grid of sample targets, and produce an output for each input location. These outputs

¹<https://gitlab.kitware.com/computer-vision/geowatch>

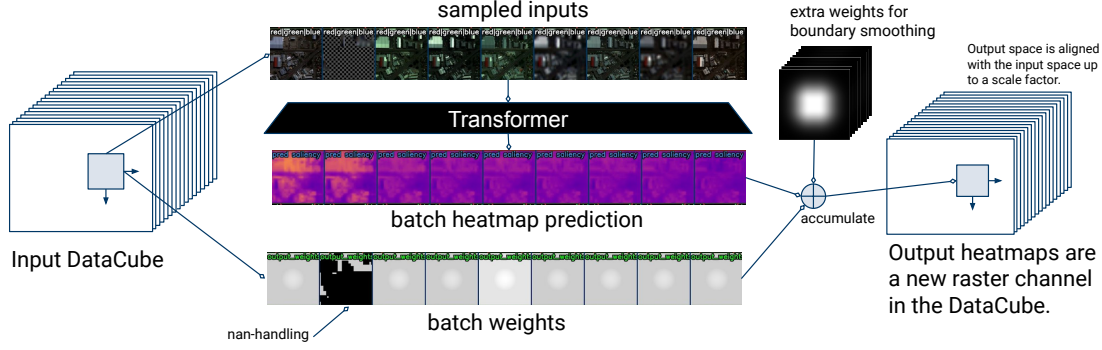


Fig. 3: Prediction Pipeline. Given an enumeration of spacetime sample grids, the input is prepared and passed to a model, which predicts a corresponding set of heatmaps. With the input is an associated set of weights for each pixel, which is zero if the pixel is NaN or low quality as indicated by the QA mask. The heatmap predictions are accumulated into a pre-allocated buffer for for each frame in the larger video. Boundary smoothing weights are used to down-weight edges of each predicted window. When combined with overlapping windows, this results in a smooth final heatmap corresponding to each larger frame in the original video.

are stitched together in pre-allocated memory corresponding to the spacetime extent of the input dataset. The stitcher maintains an accumulation array, allowing overlapping windows to make multiple predictions for the same location in space time. The final pixel value at a location is the weighted average of all inputs accumulated there. Each prediction window augments the weights provided by the dataset (which indicates NaN locations) with an additional filter that down-weights spatial edges and smooths boundary artifacts produced by overlapping windows. The prediction process is illustrated in Figure 3.

The result of this prediction process is either a new 1 channel saliency map, or a new per-class heatmap. The outputs are quantized to int16 and written to disk as geo-registered COGs and registered as new bands in the output KWCoco file.

5. PARTIAL WEIGHT INITIALIZATION

Our training pipeline has a unique initialization process. The network is either initialized from scratch by default or, if specified, weights are transferred using partial-weight-initialization [16]. The establishment of a partial matching between similar networks involves finding a maximum subgraph isomorphism [4], as shown in a simplified version in Figure 4.

Our networks possess extensive training history, with each network having a lineage of initialization events. This lineage comprises a chain of training events facilitated by partial weight loading, representing the model’s training history and initialization states. We have leveraged this tool to evolve our architectures by adjusting depth, re-configuring input stems for different modalities, and modifying and re-initializing network heads for various objectives. Notably, training on semantically rich input features, such as activations of a pretrained MAE, enhances model performance. Interestingly, even after removing this feature and continuing training with RGB inputs, the

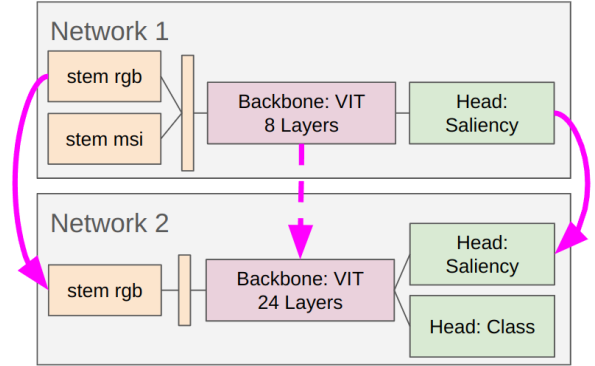


Fig. 4: Partial Weight Loading. A partial matching between similar networks is established by finding a maximum common subtree embedding [4]. Unmatched destination weights are reinitialized. In this example the input stem MSI data is dropped, the backbone is extended from 8 layers to 24 layers and partially initialized, a new class head is initialized, and the existing saliency head and RGB stem are exactly copied.

model retains its improved performance — a phenomenon we term “instillation.”

6. DETECTING HEAVY CONSTRUCTION EVENTS

The motivating application of our system is the detection and classification of heavy construction events for the IARPA SMART challenge. Using the techniques described in this paper we continuously trained models that improved over time as shown in Figure 5. Our system works in two phases: broad-area-search (BAS) and activity characterization (AC). In BAS we search for candidate regions using data sampled at 10m GSD. In AC we zoom into candidate detections at 2m GSD,

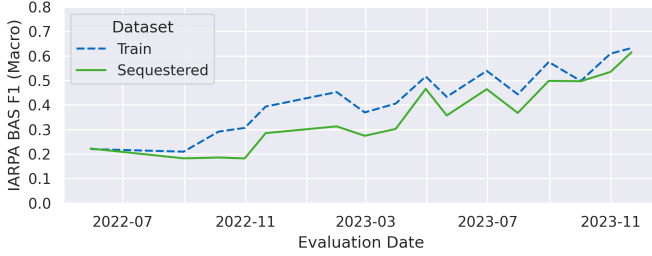


Fig. 5: Instillation Improved Scoring Over Time. Over an 18-month period, our F1 scores for the IARPA SMART BAS task ($\uparrow$ is better) improved on both the training regions (dashed-blue) and the sequestered test regions (solid-green). Each model is finetuned under different training conditions from its most recent ancestor, initializing each step using our partial weight loading approach. The scores are reported to us from an external evaluation of our system.

refine spacetime boundaries, and classify the phase of construction.

Broad Area Search For BAS our best network was trained with a combination of red, green, blue, and near-infrared bands from Landsat, Sentinel, and Worldview as well as a 36-band derived feature using the COLD algorithm [17]. We found that an important step was time-averaging (using the median) our input KWCoco files such that all images per-sensor per-year were averaged together to produce 1 image per sensor per year. An important note is that COLD features are only produced once per-year for Landsat 8 and Sentinel 2 data. Instead of averaging these features together we simply assign them to the nearest average image in time.

Given the trained BAS saliency model and a dataset, we predict saliency heatmaps for each frame. We then extract the polygons from these heatmaps, which will become our initial predicted site boundaries. First, we find the maximum saliency response over all time. We use a threshold (0.375) to binarize this max-image, and produce polygons. These are our BAS spatial bounds. Any polygon outside an area threshold of under $7200m^2$ or above $8km^2$ is removed. We then assign temporal extents by looking again at the heatmaps. For each 1 year window (which may contain multiple predictions for different sensors) we average predictions together. For each of these frames, average saliency response under the polygon and assign it as the “score” for that spacetime observation. Any observation with a score under 0.3 (chosen via grid search) is removed, which defines the start / end time for the site proposal.

Activity Characterization Given a set of site proposals, we cluster them into a set of smaller regions-of-interest. We build a KWCoco dataset by pulling Sentinel 2 and Worldview data with a maximum resolution of $2m$ GSD, downsampling all high-resolution sources. We pull all available Worldview imagery, but we consume only least cloudy Sentinel 2 image per month.

Predicting class heatmaps works almost exactly as in BAS. For each video we define a target grid, push each input through the network and we output a new dataset with *both* 4-channel class heatmaps and high resolution saliency heatmaps.

To extract high-resolution polygons, we employ a tracking process for each site cluster. We load predicted heatmaps for each cluster, multiply the “Active Construction” and “ac-salient” channels, and construct a volume. Pixels inside each BAS polygon are considered, and all other pixels are zeroed. The tensor is binarized based on a threshold (e.g., 0.3), and connected components are found. The maximum response in each component serves as seed points for a watershed algorithm, filling the BAS polygon area with smaller polygons. Small polygons are removed. The maximum AC-salient score is used as a singular filter for each site. A site is rejected if its singular score is below 0.3, and if the maximum per-class score is not above 0.3, the observation is labeled as “No Activity.” The start date for a site is determined by the first “Site Preparation” observation, and the end date is determined by the last “Active Construction” observation. Any site without “Site Preparation” or “Active Construction” predictions is rejected.

7. DISCUSSION

We have introduced GeoWATCH, a software framework for training and predicting AI models on heterogeneous raster time sequences. The system handles tasks like image classification, activity recognition, object detection, or object tracking based on MS-COCO-compatible input data encoding. We demonstrated its effectiveness in detecting heavy construction events in diverse satellite image sequences. Our training system features a novel partial weight loading mechanism, utilizing sub-graph isomorphism, enabling continuous network training and modification over extended periods. This facilitates the creation of a lineage of models, showing improved performance over time by adjusting configurations while maintaining a core backbone. Notably, training models with many features and initializing new networks with fewer input features retains performance, suggesting an “instillation” of knowledge from previous models into new ones. Future research will explore experimental verification of these observations.

Acknowledgement This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), 6 Intelligence Advanced Research Projects Activity (IARPA), via 2021-2011000005. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

8. REFERENCES

- [1] Martin Claverie, Junchang Ju, Jeffrey G Masek, Jennifer L Dungan, Eric F Vermote, Jean-Claude Roger, Sergii V Skakun, and Christopher Justice, “The harmonized landsat and sentinel-2 surface reflectance data set,” *Remote sensing of environment*, vol. 219, pp. 145–161, 2018.
- [2] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick, “Microsoft COCO: Common Objects in Context,” in *Computer Vision – ECCV 2014*, David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, Eds., Cham, 2014, Lecture Notes in Computer Science, pp. 740–755, Springer International Publishing.
- [3] Shibhansh Dohare, J. Fernando Hernandez-Garcia, Parash Rahman, Richard S. Sutton, and A. Rupam Mah-mood, “Loss of Plasticity in Deep Continual Learning,” Aug. 2023, arXiv:2306.13812 [cs].
- [4] Valiente Feruglio and Gabriel Alejandro, “On the maximum common embedded subtree problem for ordered trees,” Mar. 2003, Accepted: 2016-11-29T09:25:22Z.
- [5] Juan Benet, “IPFS - Content Addressed, Versioned, P2P File System,” July 2014.
- [6] Adam J. Stewart, Caleb Robinson, Isaac A. Corley, Anthony Ortiz, Juan M. Lavista Ferres, and Arindam Banerjee, “TorchGeo: Deep Learning With Geospatial Data,” Sept. 2022, arXiv:2111.08872 [cs].
- [7] MMSegmentation Contributors, “OpenMMLab Semantic Segmentation Toolbox and Benchmark,” <https://github.com/open-mmlab/mms Segmentation>, July 2020.
- [8] Robert Cheetham and Azavea/Element 84, “Raster vision: An open source library and framework for deep learning on satellite and aerial imagery (2017-2023),” <https://github.com/azavea/raster-vision>.
- [9] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, “Distilling the Knowledge in a Neural Network,” Mar. 2015, arXiv:1503.02531 [cs, stat].
- [10] Chengming Hu, Xuan Li, Dan Liu, Haolun Wu, Xi Chen, Ju Wang, and Xue Liu, “Teacher-Student Architecture for Knowledge Distillation: A Survey,” Aug. 2023, arXiv:2308.04268 [cs].
- [11] “Spatiotemporal asset catalog (stac) specification,” <https://stacspec.org/>, 2021, Accessed: 2023-12-04.
- [12] Jon Crall and Delayed Image contributors, “Delayed image,” https://gitlab.kitware.com/computer-vision/delayed_image, Dec. 2023.
- [13] “Cloud optimized geotiff,” <https://www.cogeo.org/>, Accessed: 2023-12-04.
- [14] Jordan T. Ash and Ryan P. Adams, “On Warm-Starting Neural Network Training,” Dec. 2020, arXiv:1910.08475 [cs, stat].
- [15] Connor Greenwell, Jon Crall, Matthew Purri, Kristin Dana, Nathan Jacobs, Armin Hadzic, Scott Workman, and Matt Leotta, “WATCH: Wide-Area Terrestrial Change Hypercube,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2024, pp. 8277–8286.
- [16] Jon Crall and Torch Liberator contributors, “Torch liberator,” <https://gitlab.kitware.com/computer-vision/torch-liberator>, Dec. 2023.
- [17] Zhe Zhu, Junxue Zhang, Zhiqiang Yang, Amal H. Aljadani, Warren B. Cohen, Shi Qiu, and Congliang Zhou, “Continuous monitoring of land disturbance based on Landsat time series,” *Remote Sensing of Environment*, vol. 238, pp. 111116, Mar. 2020.