Sarvalanche -S1 Avalanche Detection Pipeline

Dr. Zachary Hoppinen

February 2026

What is sarvalanche

- Python package that handles data finding, reprojection, differencing, and detection for avalanche debris. Pytest implemented.
- How does it work? Combines static probabilities of avalanches debris (slope, fcf, runout probability, water mask) with SAR features (local resolution weighted backscatter change currently)
- IO – we get S1 (opera rtcs, local resolution), forest cover, slope angles, UCLA snowmodel SWE, and run the FlowPy debris flow model over a dem to generate runout cell counts.

What is sarvalanche

- Runtime – over most of Sawtooth Mountains full run (including flowpy modeling (95% of time) takes 33 minutes. Non-flowpy run time = 2 minutes, data-loaded already = <1 minute.

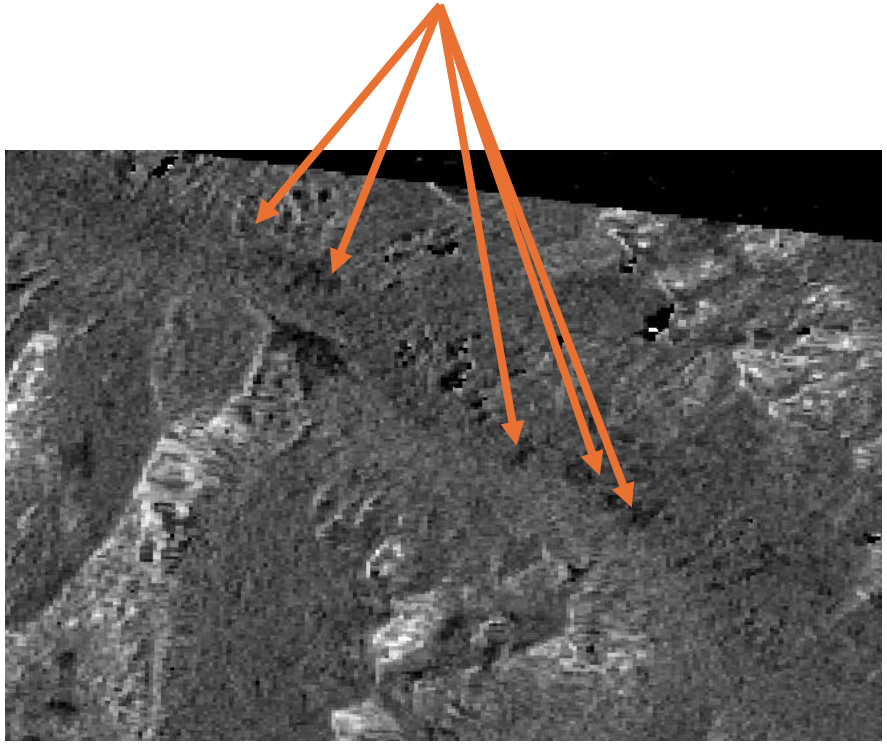
```
2026-02-11 17:11:52 - sarvalanche.utils.timing - INFO - 💣 PIPELINE TIMING SUMMARY
2026-02-11 17:11:52 - sarvalanche.utils.timing - INFO - ================================================================================
2026-02-11 17:11:52 - sarvalanche.utils.timing - INFO -   3_load_assemble_dataset                                32m 45s ( 98.4%)
2026-02-11 17:11:52 - sarvalanche.utils.timing - INFO -   8_export_results                                       25.1 sec (  1.3%)
2026-02-11 17:11:52 - sarvalanche.utils.timing - INFO -   7_group_detections                                     3.6 sec (  0.2%)
2026-02-11 17:11:52 - sarvalanche.utils.timing - INFO -   6_pixelwise_probabilities                              2.3 sec (  0.1%)
2026-02-11 17:11:52 - sarvalanche.utils.timing - INFO -   5_static_probabilities                                  98 ms (  0.0%)
2026-02-11 17:11:52 - sarvalanche.utils.timing - INFO -   4_calculate_weights                                     41 ms (  0.0%)
2026-02-11 17:11:52 - sarvalanche.utils.timing - INFO -   6.5_apply_masks                                         18 ms (  0.0%)
2026-02-11 17:11:52 - sarvalanche.utils.timing - INFO -   1_validation                                            1 ms (  0.0%)
2026-02-11 17:11:52 - sarvalanche.utils.timing - INFO -   2_setup_cache                                           0 ms (  0.0%)
2026-02-11 17:11:52 - sarvalanche.utils.timing - INFO - --------------------------------------------------------------------------------
2026-02-11 17:11:52 - sarvalanche.utils.timing - INFO -   TOTAL TIME                                            33m 16s
2026-02-11 17:11:52 - sarvalanche.utils.timing - INFO - ================================================================================
```

```
FlowPy calculation: 100%|                                                                                | 120/120 [31:27<00:00, 15.73s/task]
```

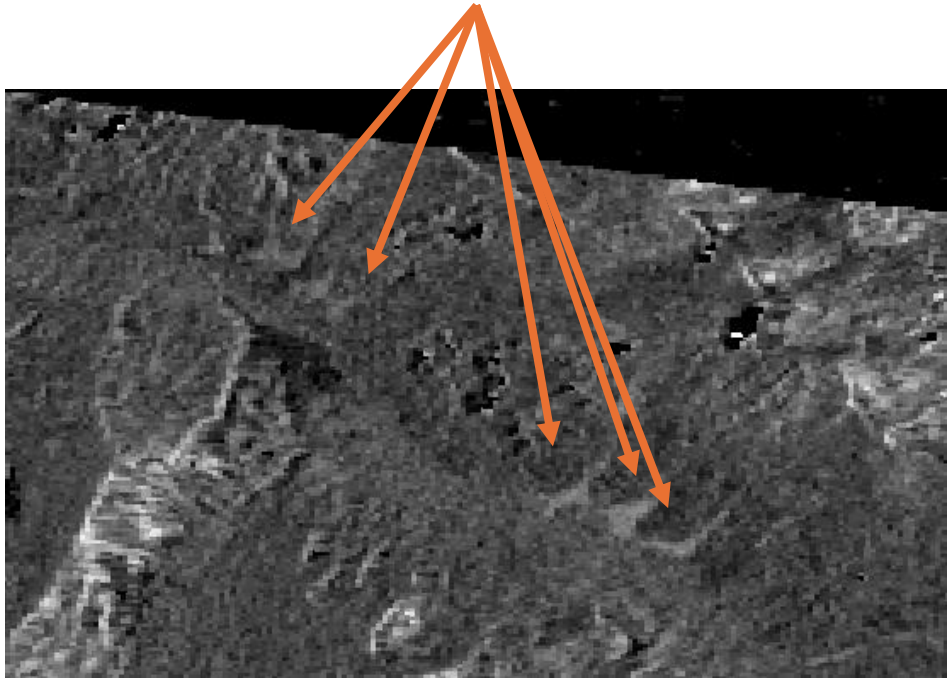
Example

- March 31st – Major earthquake triggered avalanche cycle in the Sawtooths

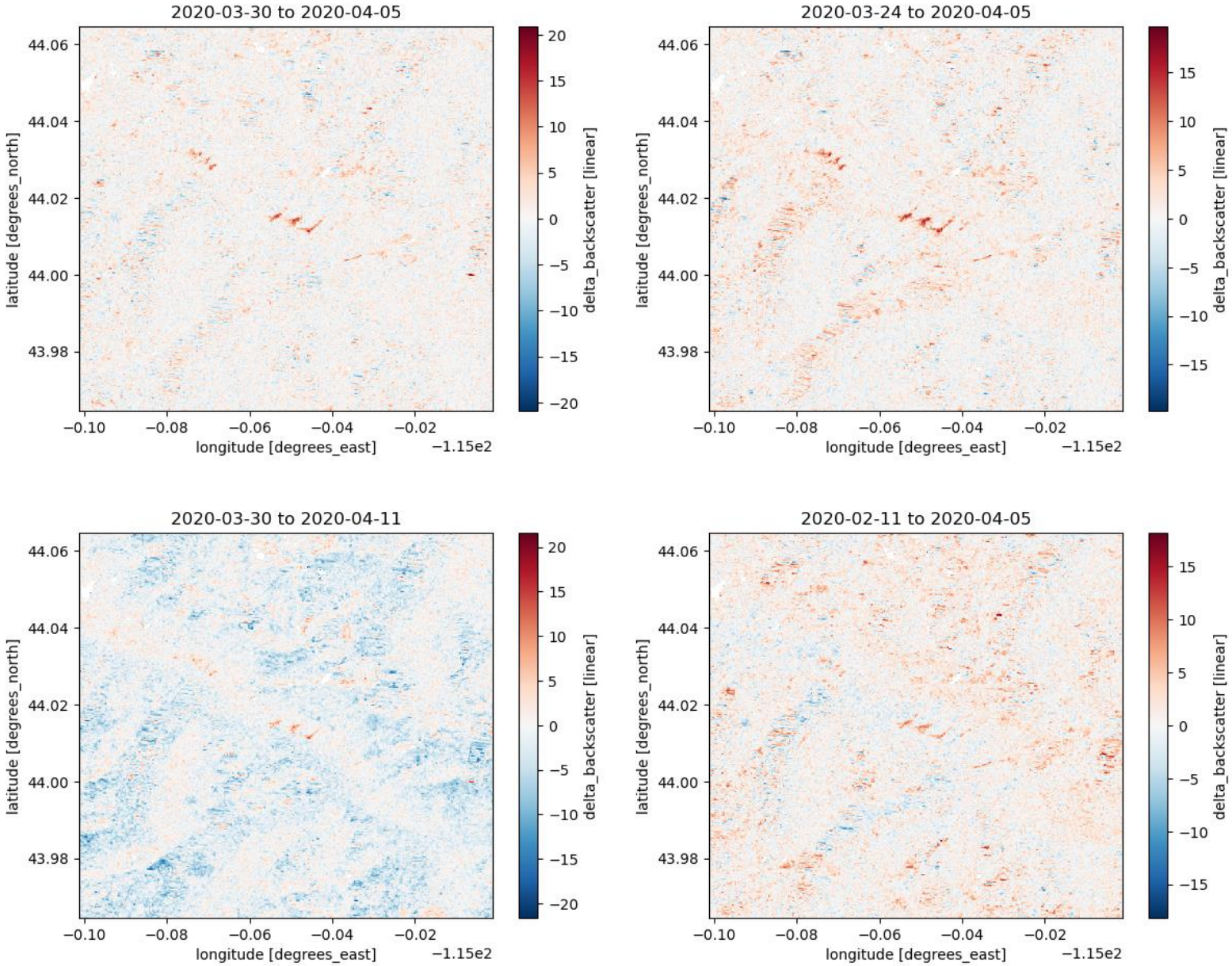
Before – March 30th – Track 71



Avalanches – April 5th – Track 71



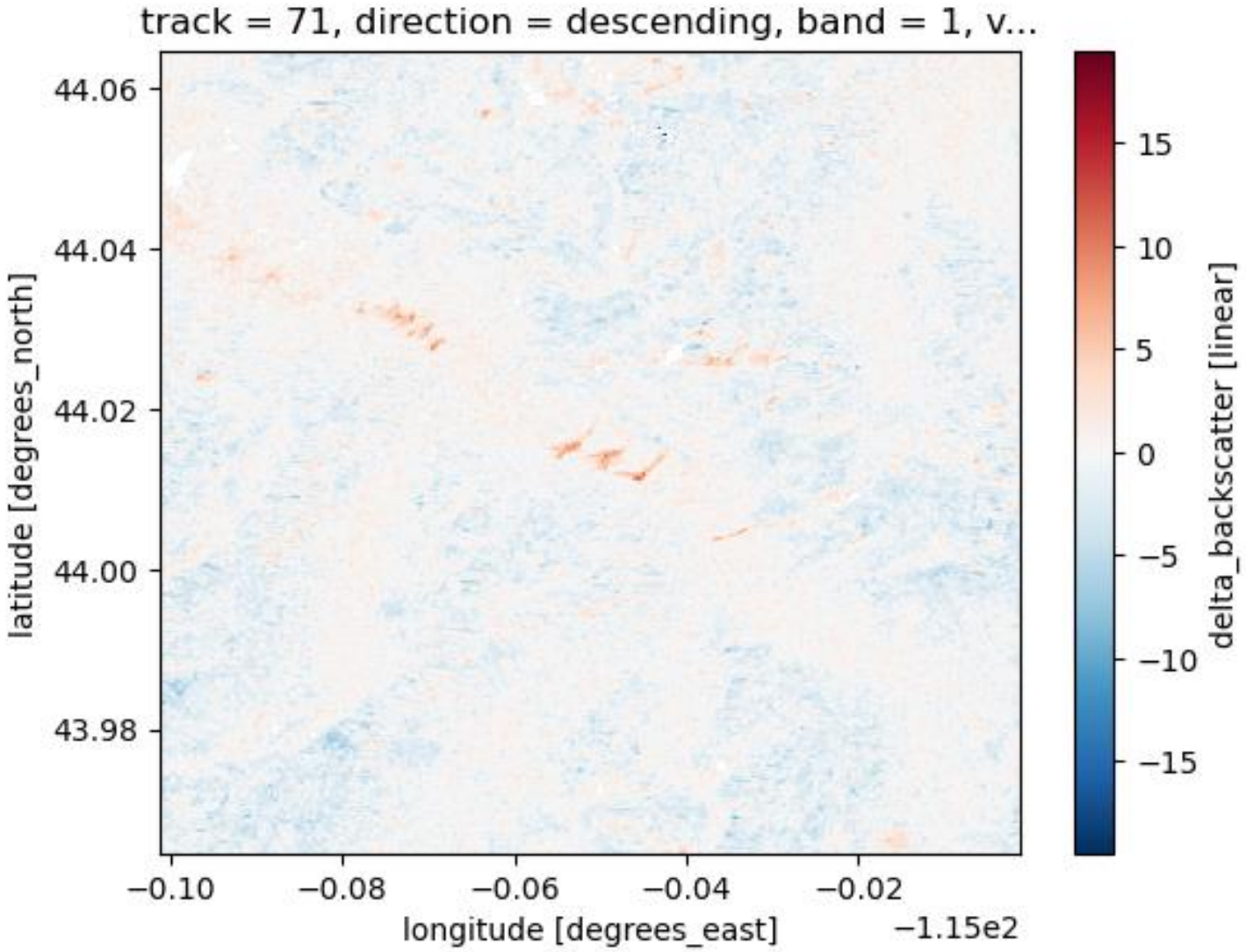
We start with dB of backscatter change over specific date (2020-03-31) from each track from all possible pairs (within a few months). These are all examples crossing 3/31 from same orbit geometry.



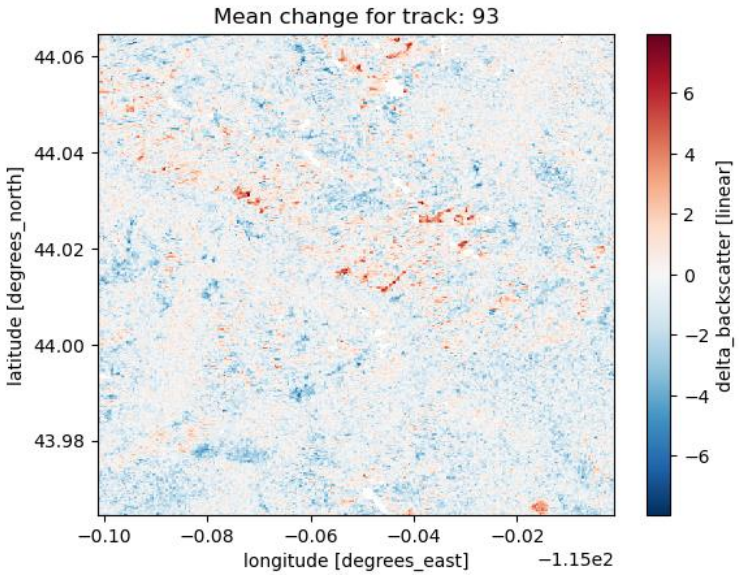
These are weighted by temporal distance between each pair to generate a track based mean change that weights closer pairs more

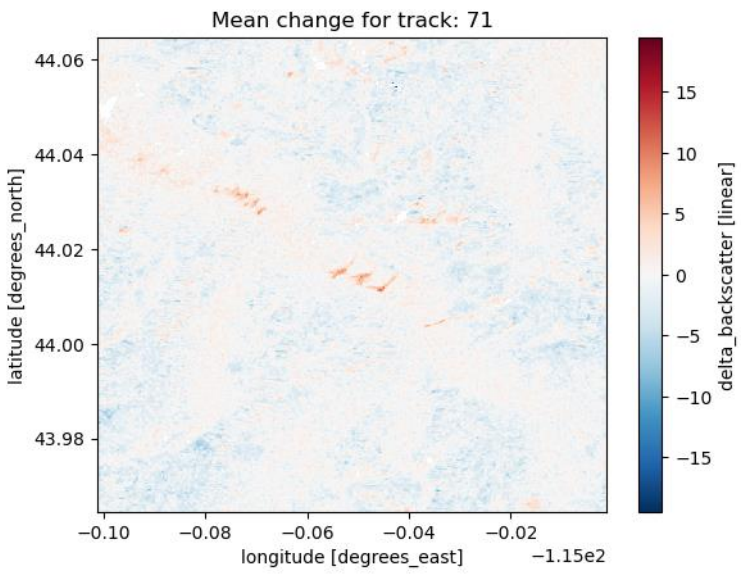
w_temporal = np.exp(-np.abs(dt_days) / tau_days)

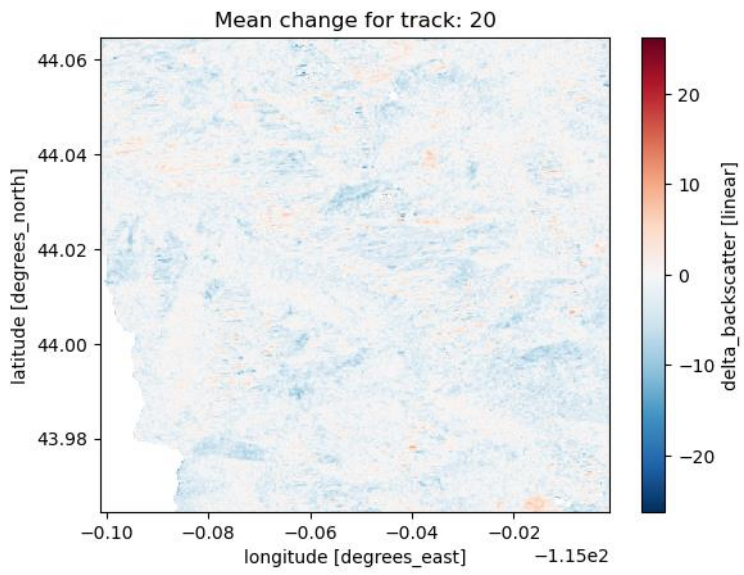
With tau_days = 24 currently



This is repeated for all available tracks



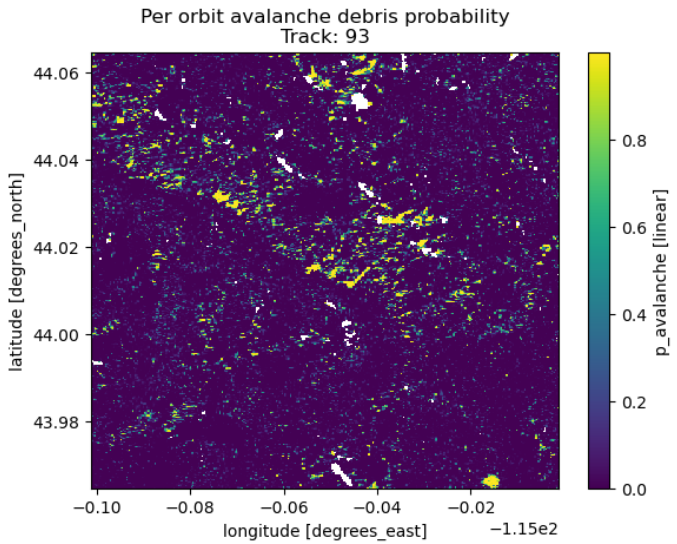


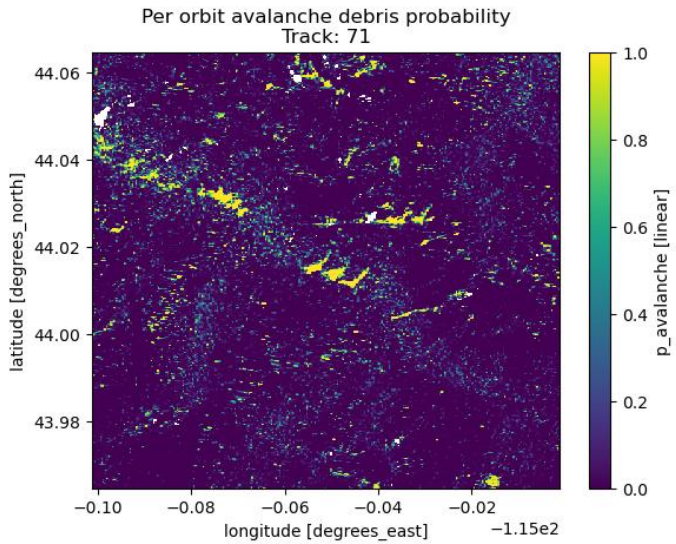


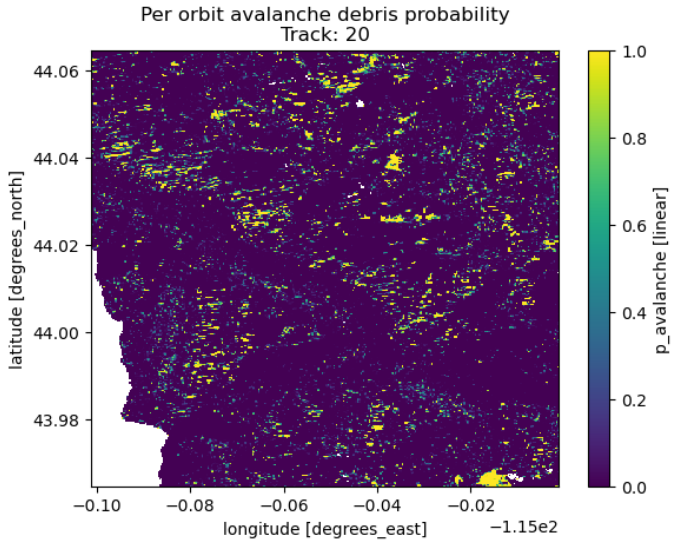
We convert this mean change to a probability

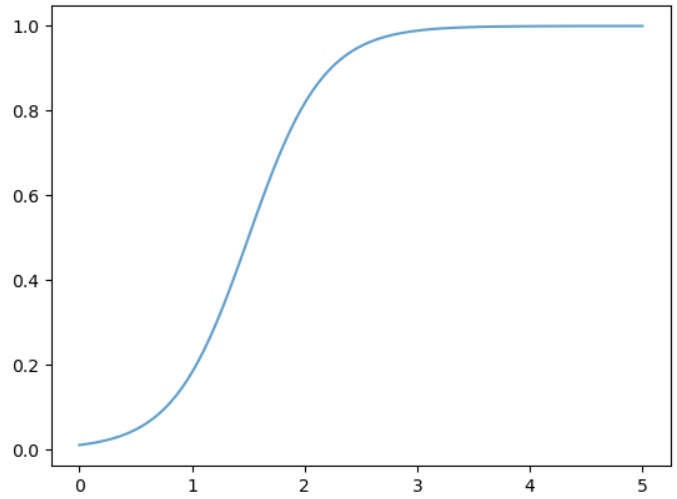
prob = scipy.expit(diff - threshold_db)*logistic_slope

We show the probability to backscatter change to the right



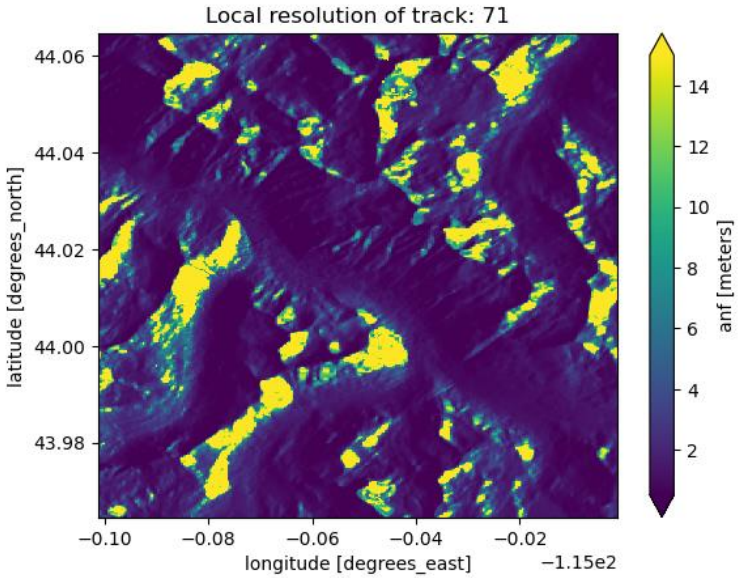


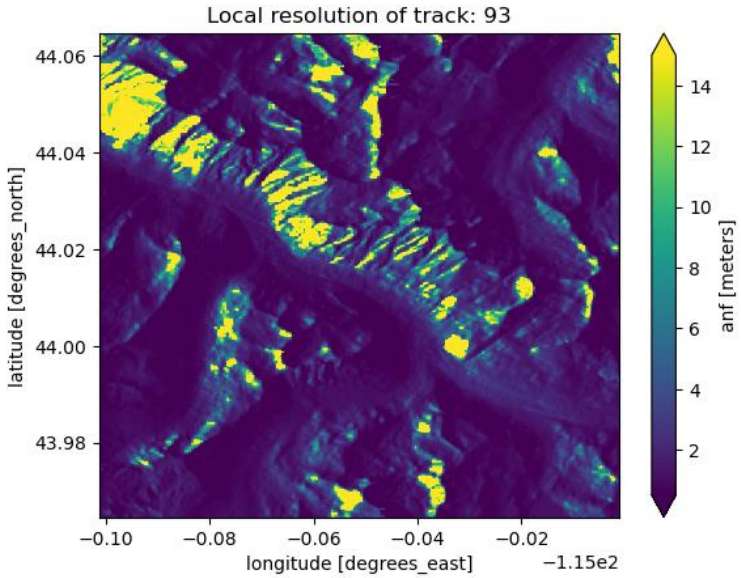


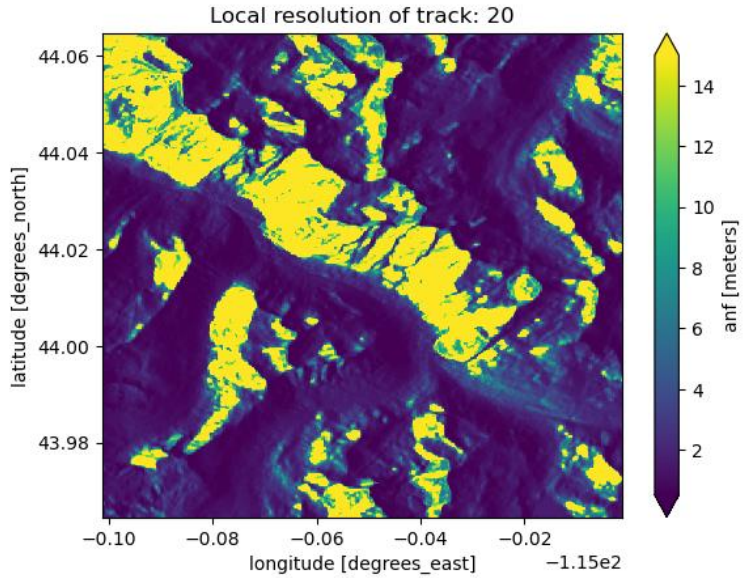


These are then combined using the inverse of local resolution of each pixel to weight the relative tracks.

- So lower local resolution == less weight for final probability
- We also use a relative weighting of VV vs VH (1.0 vs 0.8)







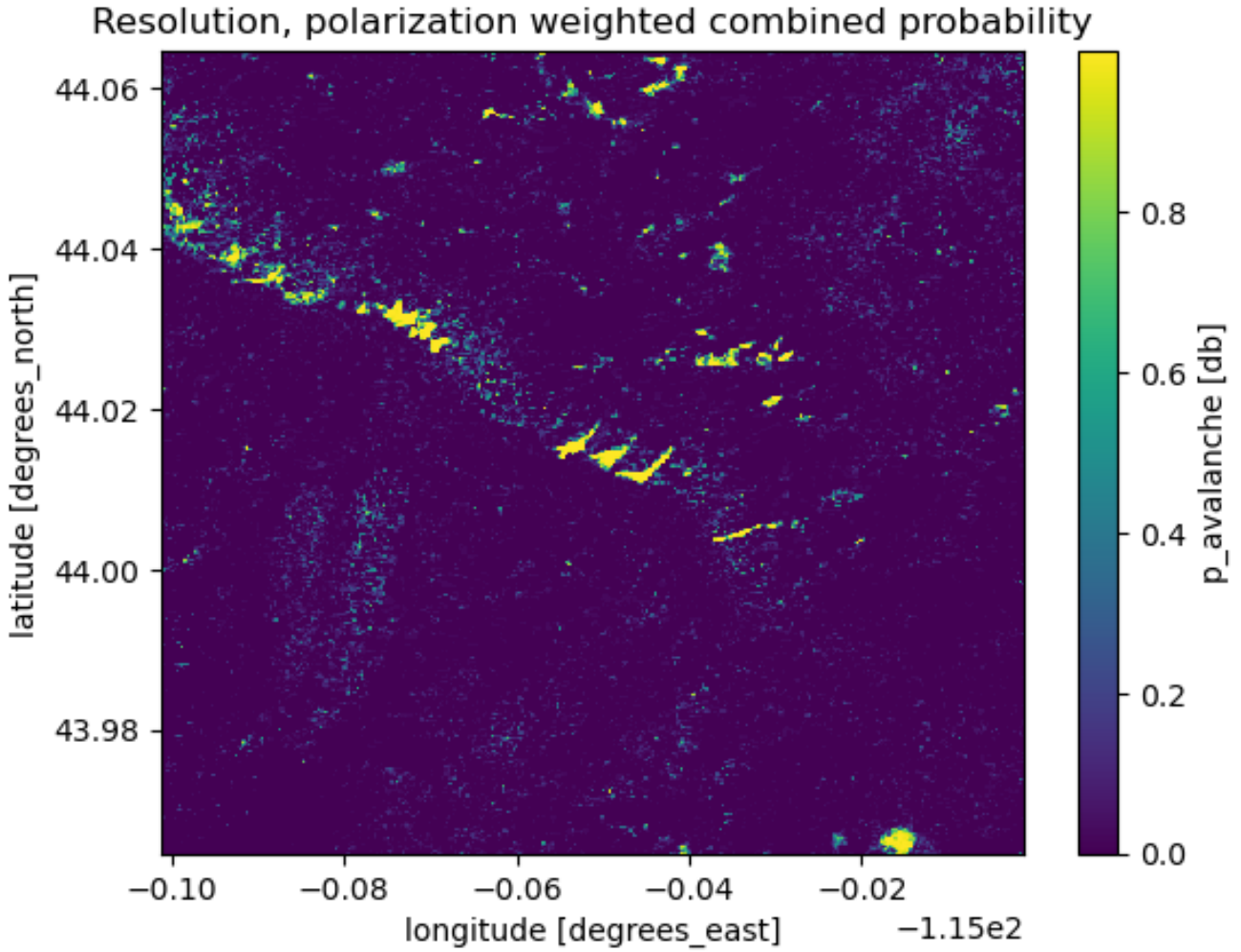
Combining probabilities from multiple tracks

Log-odds combination

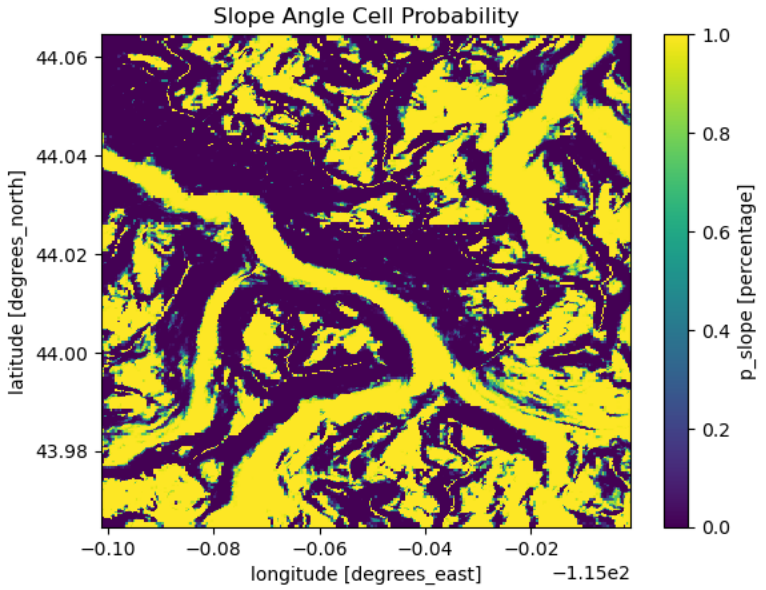
```
log_odds = np.log(probs_safe / (1 - probs_safe))
# Weighted average in log-odds space. Weight is inverse of local resolution
log_odds_combined = (log_odds * weights).sum(dim=dim)
# Convert back to probability
p_combined = 1 / (1 + np.exp(-log_odds_combined))
```

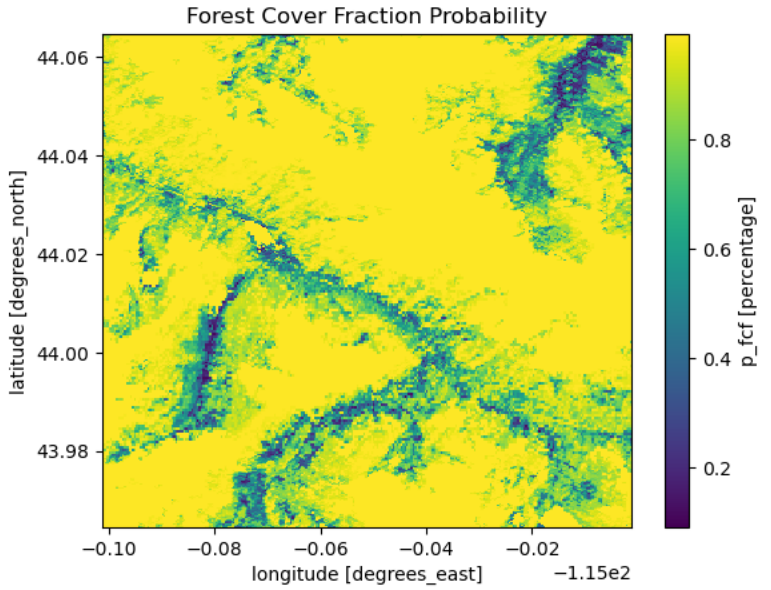
Boosting:

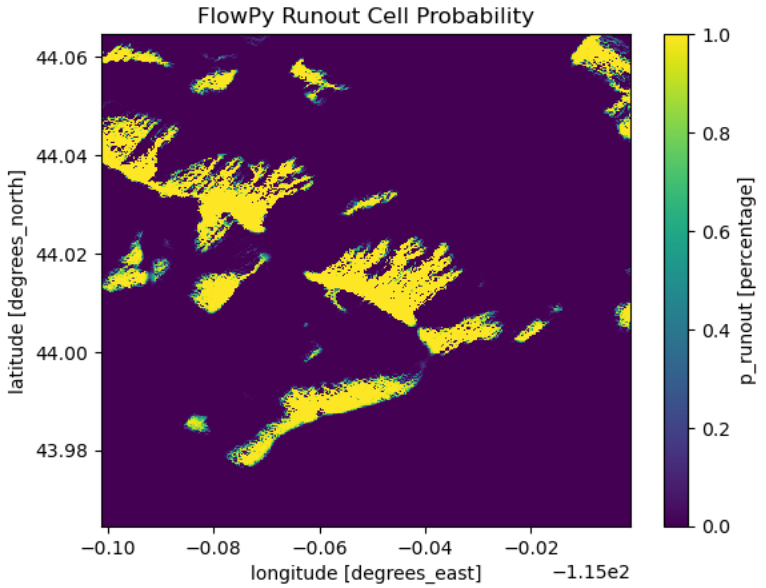
```
probs_filtered = probs.where(probs >= min_prob_threshold)

# Count valid sources (how many detect something)
n_valid = probs_filtered.notnull().sum(dim=dim)
n_total = probs.sizes[dim]
# Agreement: fraction showing detection
agreement = n_valid / n_total

# Boost formula: p_final = p_mean + (1 - p_mean) × agreement² × strength
boost = (1 - p_combined) * (agreement ** 2) * agreement_strength
p_combined = p_combined + boost
```

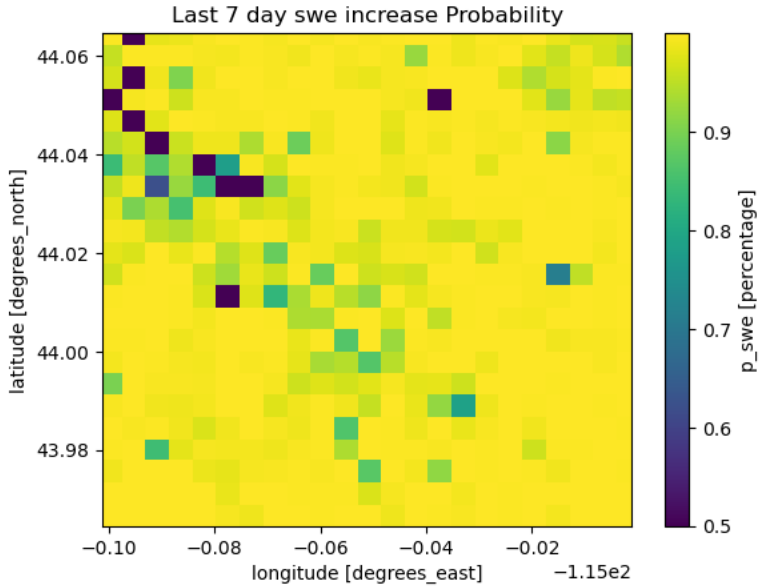


This is then combined (geometric mean this time) with non-SAR probabilities



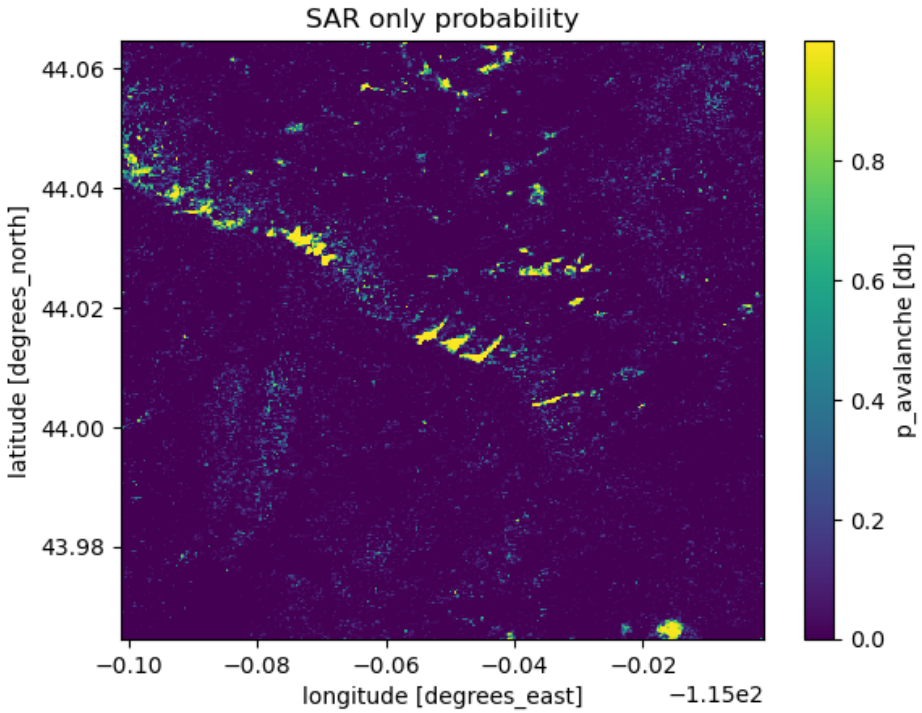


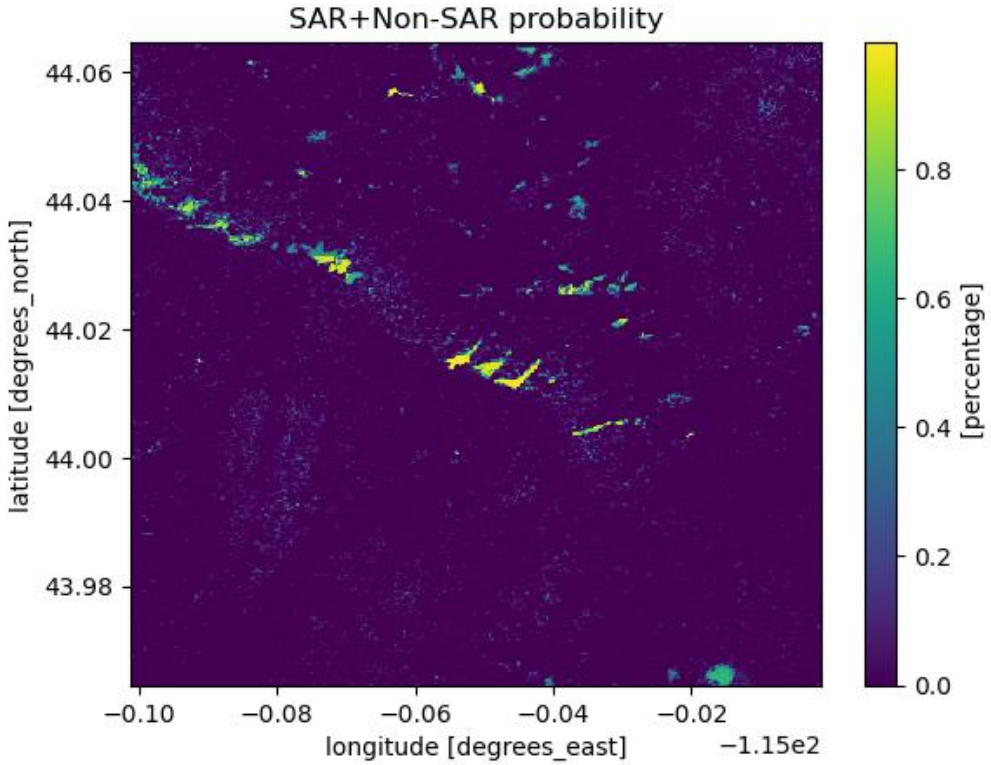




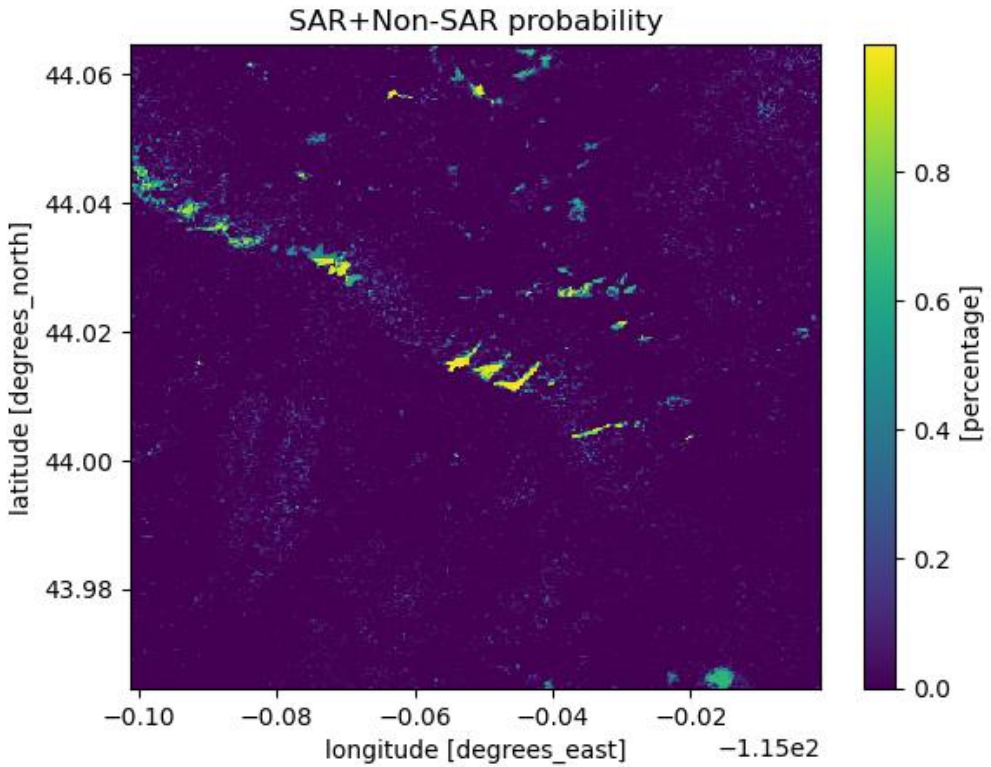
We just multiple pooled non-SAR probabilities * SAR probability.

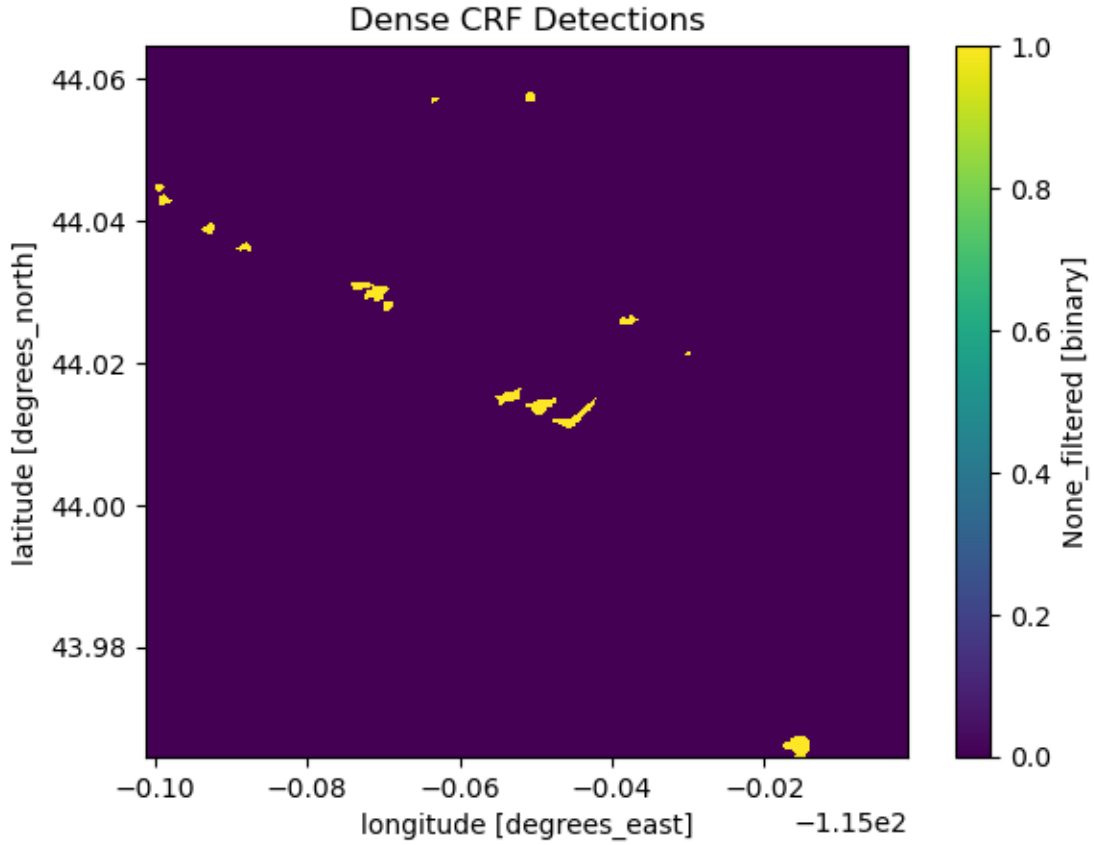
- Non-SAR layers can not increase pixel-wise probability from SAR results but can decease it.
- Reduces probability when 1. not in runout zones, 2. steep angle, 3. high fcf, 4. little recent SWE.
- Completely masks: urban and water





Last step: Use dense conditional random fields (CRF) to give weight to pixels near other debris pixels and remove sharp changes in debris/background

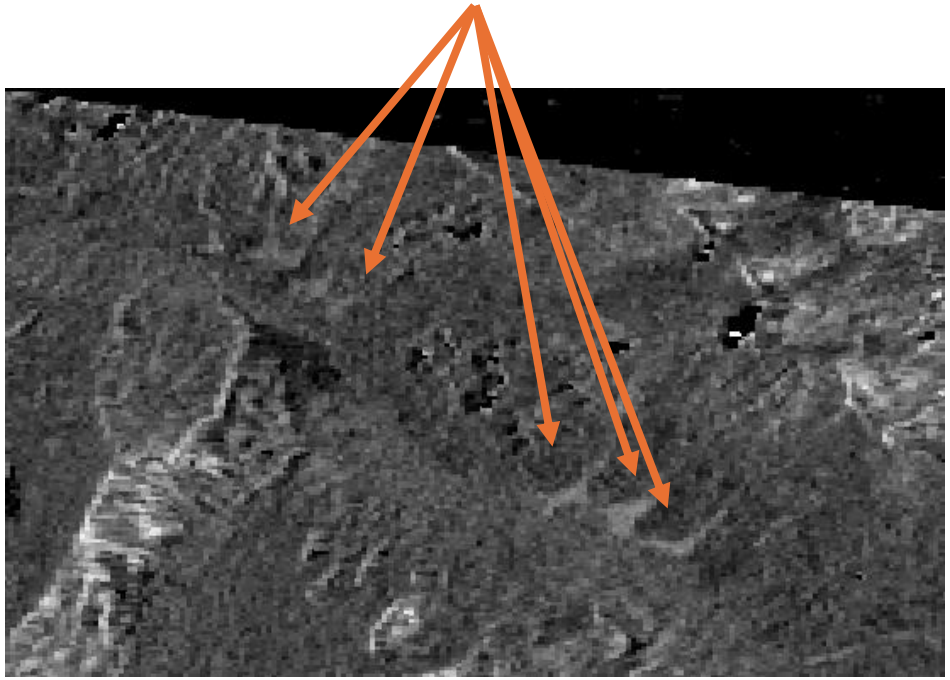


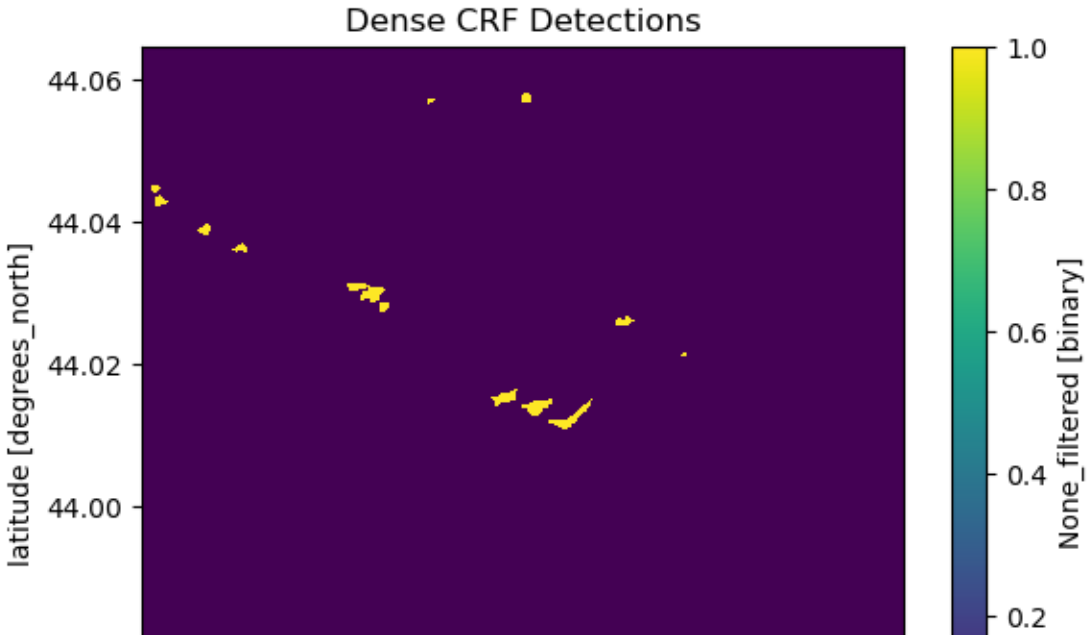


Re state example with dectionsExample

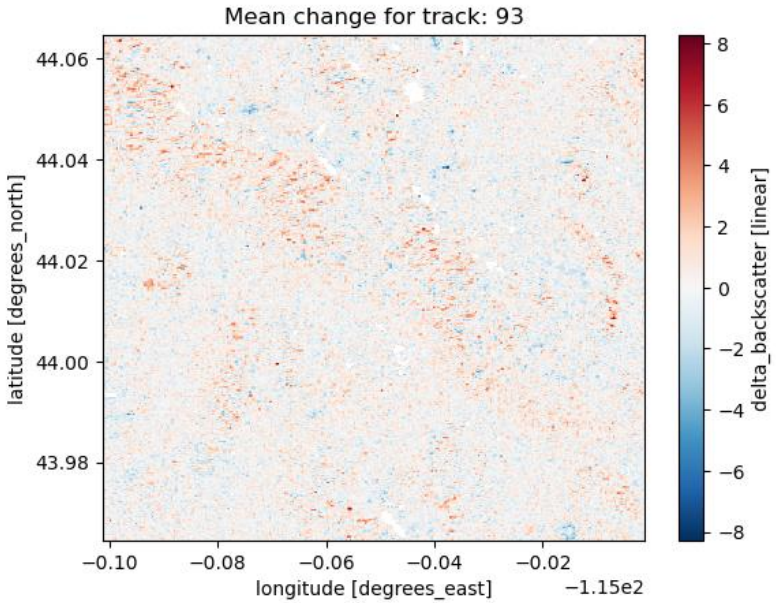
- March 31st – Major earthquake triggered avalanche cycle in the Sawtooths

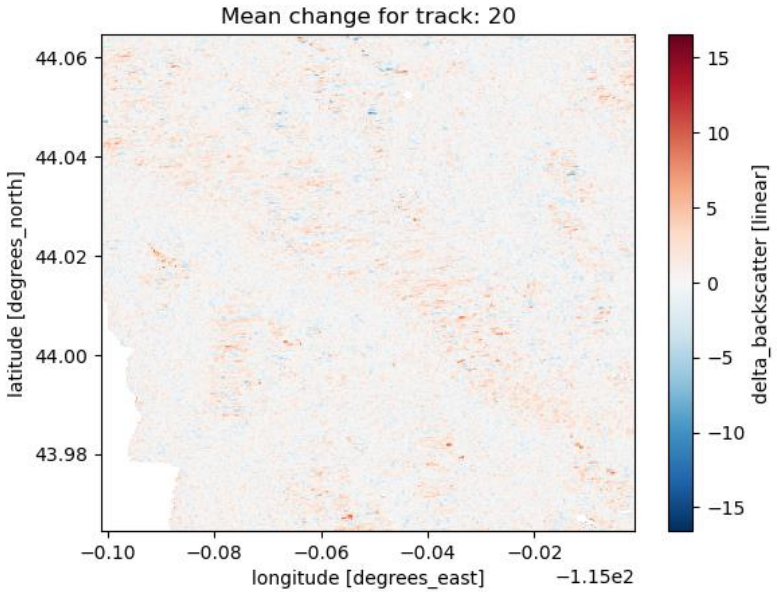
Avalanches – April 5th – Track 71

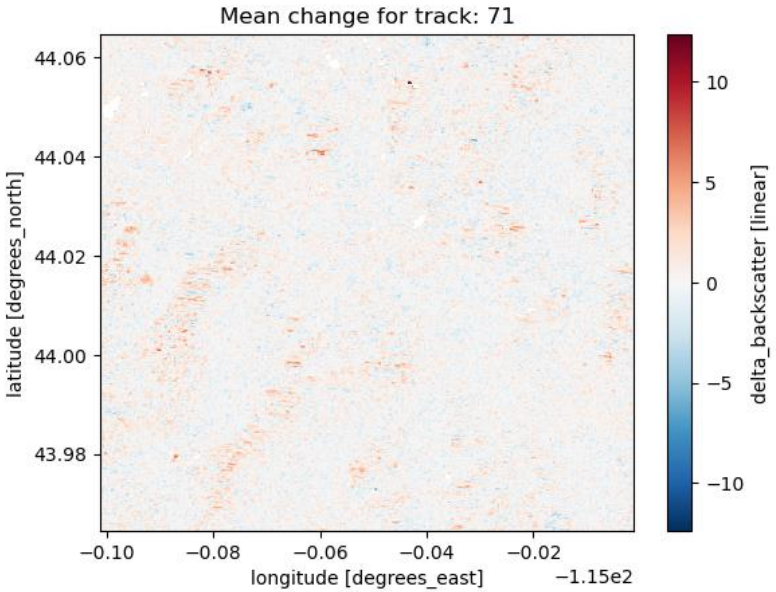




Re-run again on Jan 31st, 2020 for an example of minimal avalanches

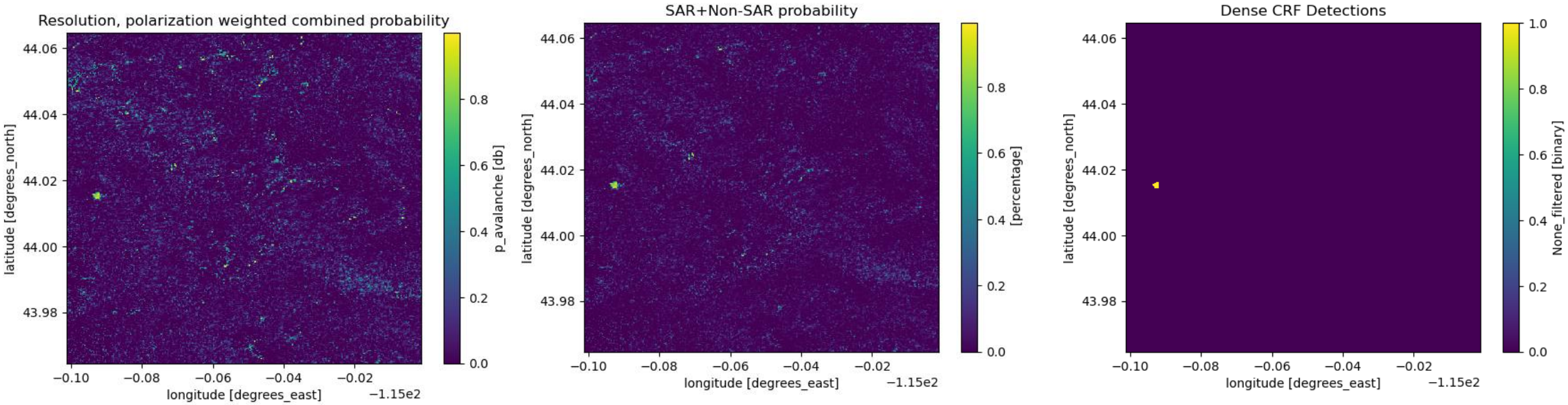






Re-run again on Jan 31st, 2020 for an example of minimal avalanches

Single detection for non-avalanche date



Next Steps?

- **ISSW presentation** – run over last 5 years over each of the western US avalanche regions to generate database for other researchers. What level of temporal specificity? Yearly? Monthly? Daily?
- **Processing improvements**:
  - Multi-temporal. When does an avalanche first appear in a SAR scene = set as detection date.
  - Use backscatter distributional changes in addition to backscatter change
  - Use empirical detections to train ML model (unet) based on empirical detections and then re-run and capture more detections
  - Incorporate other/more sentinel-1 coherence and NISAR based pixel based probability pipelines.