

Proposal #2



Approver/ Mentors:

[@bw4sz](#)

[@jveitchmichaelis](#)

[@henrysenyondo](#)

[@ethanwhite](#)

Summary

self link:

[Proposal 2: Recovering historical ...](#)

Recovering historical image data using automated ortho-registration and image-matching

Owner: Muhammad Saqlain

2mesaqlain@gmail.com

Contributors:

Muhammad Saqlain

Status: **Submitted**

Title & Summary

Title: Recovering Historical Forest Health Annotations via Automated Ortho-Registration and Weak Augmentation

Core Problem Summary:

- ADS polygon annotations are collected from aircraft at approximately 1:24,000 scale.
- GPS imprecision, flight geometry errors, and temporal mismatch between survey year and NAIP collection (often 1 to 3 years apart) cause systematic spatial offsets.
- The U.S. Forest Service ADS program is one of the longest-running airborne forest health monitoring programs in the world. Recovering usable training data from this archive is high-impact and largely unsolved.
- Manual alignment is impractical at scale. An automated, learned correction is the only feasible path.

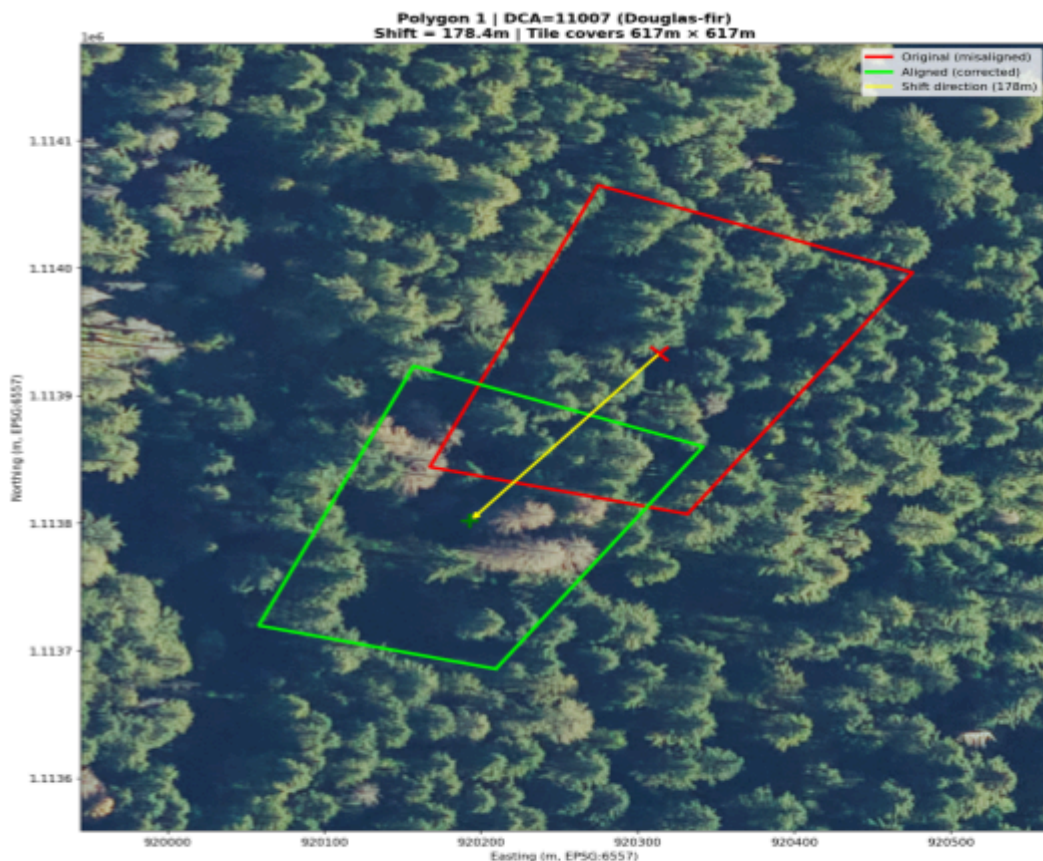
Pre-GSoC Engagement & Problem-Solving Journey

I started working on this project in late February 2026, drawn by the intersection of remote sensing, deep learning, and biodiversity monitoring. Over the past four weeks, I worked through the full problem space, from raw data access to a working alignment prototype, encountering (and solving) problems at every stage. Each challenge below changed the direction of the project.

Challenge 1: Building the Seed Dataset from Scratch

The obstacle: No ground-truth dataset of correctly aligned (ADS polygon, NAIP imagery) pairs existed. I had to create one.

- The ADS data (U.S. Forest Service Aerial Detection Survey) was not directly downloadable, I used a VPN set to Chicago to access the USFS server, then spent days reading the GDB schema: DCA_CODE, HOST1, SURVEYYEAR, geometry projections in Albers NAD83 (ESRI:102039).
- First automated attempt, NDVI contrast scoring. NAIP provides 4 spectral bands (R, G, B, NIR). I built `auto_select_gold_standard.py` to rank 2,500 ADS polygons by the contrast between NDVI inside vs. outside each polygon. A correctly-aligned polygon over dead trees should have low NDVI inside and high NDVI outside.
- Why it failed: The top-ranked results were parking lots and residential driveways surrounded by trees, achieving the same ~0.5 NDVI contrast as a real beetle kill. The metric was mathematically correct but ecologically wrong.
- The lesson: I had jumped straight to automation without first understanding what "correctly aligned" looks like visually. [Automation is the final step](#), not the first.
- How I adapted: Manual verification in QGIS software. I focused on DCA code 11007 (Douglas-fir beetle), the most prevalent code in Oregon with the most distinctive aerial signature. I learned QGIS, loaded both ADS layers and Oregon NAIP, and studied bark beetle biology to recognize dead stands from imagery.
- I watched videos on bark beetle biology and aerial identification: [Mountain Pine Beetle overview](#), [Bark-Beetle causing forest die-off from the air](#), [identifying a beetle-killed tree](#).
- Result: 11 manually confirmed polygon-NAIP pairs, each verified at 30cm resolution by confirming visible dead Douglas-fir under the polygon boundary.



Challenge 2: Six Years of Wrong Imagery, The NAIP Temporal Mismatch

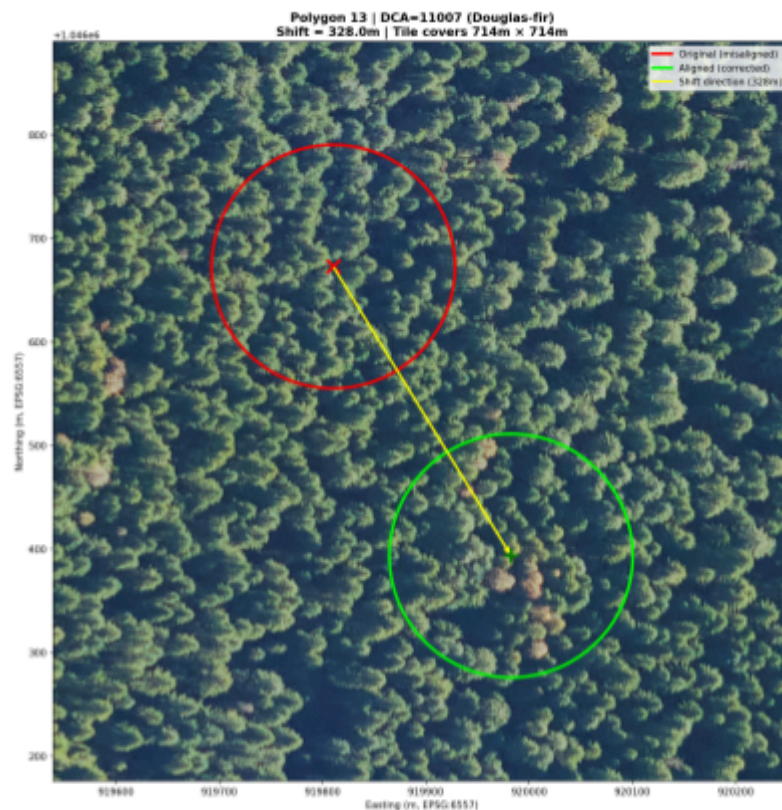
The obstacle: I fetched NAIP from the USDA CONUS PRIME ImageServer for 2016 ADS polygons, but the returned tiles were clearly not from 2016. For 3–4 days I could not explain why, the code and request both looked correct.

- Root cause: The CONUS PRIME endpoint is a "best available mosaic" that silently ignores the year parameter, always returning the most recent data, for Oregon, locked at 2022.
- Using 2016 ADS polygons against 2022 imagery creates a 6-year mismatch: trees surveyed as dead in 2016 may have decomposed by 2022; new mortality may have appeared with no polygon. This is a scientifically unjustifiable pairing that would silently corrupt any model trained on it.
- Resolution: *Oregon GEOHub* maintains separate year-specific image servers, each containing only that year's data, making year drift impossible. I built a year-matching table:

ADS Data File	Resolution	Correct NAIP Server
ADS_2016.gdb	1m, 4-band	NAIP 2016
ADS_2018.gdb	60cm	OSIP 2018
ADS_2020.gdb	60cm	NAIP 2020
INTERIM_R6_ADS_DATA2024.gdb	30cm	OSIP 2024

Key insight: Temporal matching between ADS survey year and NAIP acquisition year is non-negotiable. Without it, the entire data foundation is scientifically invalid.

The CRS reprojection bug followed immediately. After building the pipeline, overlay images showed polygons displaced by hundreds of meters from where QGIS confirmed they should be. I debugged it for some hours and after debugging revealed the coordinate reference system was being passed as the integer 3857 instead of the string "EPSG:3857", Python accepted it silently, and the converter treated coordinates as already correct. One character change fixed all 11 polygons. This bug would never have been caught without prior manual verification in QGIS.



After resolving the silent CRS reprojection bug: polygon boundary now correctly overlays the target beetle-killed stand. Without manual QGIS verification, this bug would have silently corrupted the entire training dataset.

Challenge 3: The CNN-Only Approach, Deceptive Convergence

The obstacle: I built a pure CNN pipeline, ResNet-18 backbone + Spatial Transformer Network (STN) predicting 6 affine parameters, based on three papers I independently found: [He et al. \(KDD 2022\)](#), [Jaderberg et al. \(NeurIPS 2015\)](#), and [Jiang et al. \(KDD 2021\)](#). *The model did not converge.*

- 2 bugs identified, ranging from code errors to fundamental design flaws:
 1. Starting displacement (200px) too large, the model's identity prediction saw zero overlap, so the loss gradient was uninformative.
 2. Fundamental: 11.3M parameters (ResNet-18) trained on 18 images. The model could not learn both tree detection AND alignment from this few samples.
- The deceptive convergence: The loss appeared to improve across curriculum phases, but only because each phase used smaller displacements. The model learned "do nothing" (predict identity) and was rewarded for it. The improvement was an artifact of the evaluation protocol, not actual learning.

The paradigm shift: I re-read the mentor's proposal and found: "Using automated template matching and orthorectification approaches." I tried to approach this problem again from First Principle Thinking and after understanding it, and reading papers for some days, I come to a hybrid pipeline, instead of a pure CNN approach. I shifted from my three papers to the mentor's three reference papers as primary guidance, DFA-Net, GLU-Net for historical maps, and ProximityAlign, and adopted a new design philosophy:

"Use the structure of the problem to constrain the search space, rather than hoping a CNN will discover it from raw pixels."

Challenge 4: From NCC Failure to ProximityAlign, Finding the Right Signal

The obstacle: After the paradigm shift, I implemented NCC (Normalized Cross-Correlation) template matching on vegetation stress maps. Result: 0/11 correct alignments, it made every sample worse, increasing mean error from 134px to 227px.

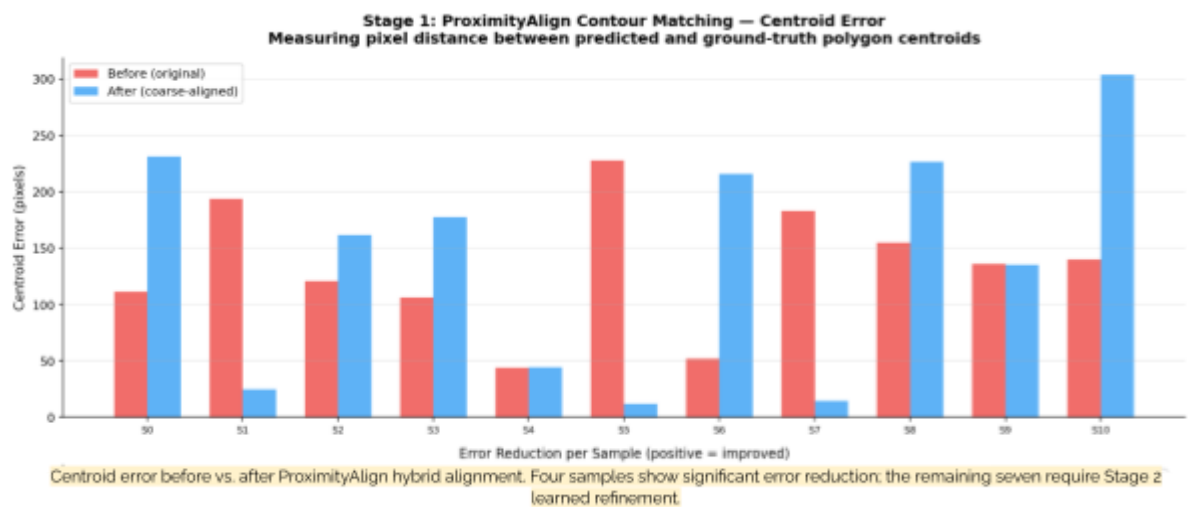
- Root cause: NCC computes "average stress under polygon", it averages away the polygon's unique contour shape, reducing it to a single number. It matches intensity, not structure.
- Signal analysis finding: I built `diagnose_signal.py` and discovered that the spectral difference between correct and incorrect polygon positions is ~0.02 (2% of signal range), effectively within noise. Classical spectral methods are at their theoretical limit for this forest type. *This is a genuine scientific finding.*
- DeepForest limitation: DeepForest detects only 2–11% of pixels as tree crowns on Oregon mortality tiles. The resulting gap map is 89–98% "gap" everywhere, nearly uniform and useless as a spatial alignment signal.

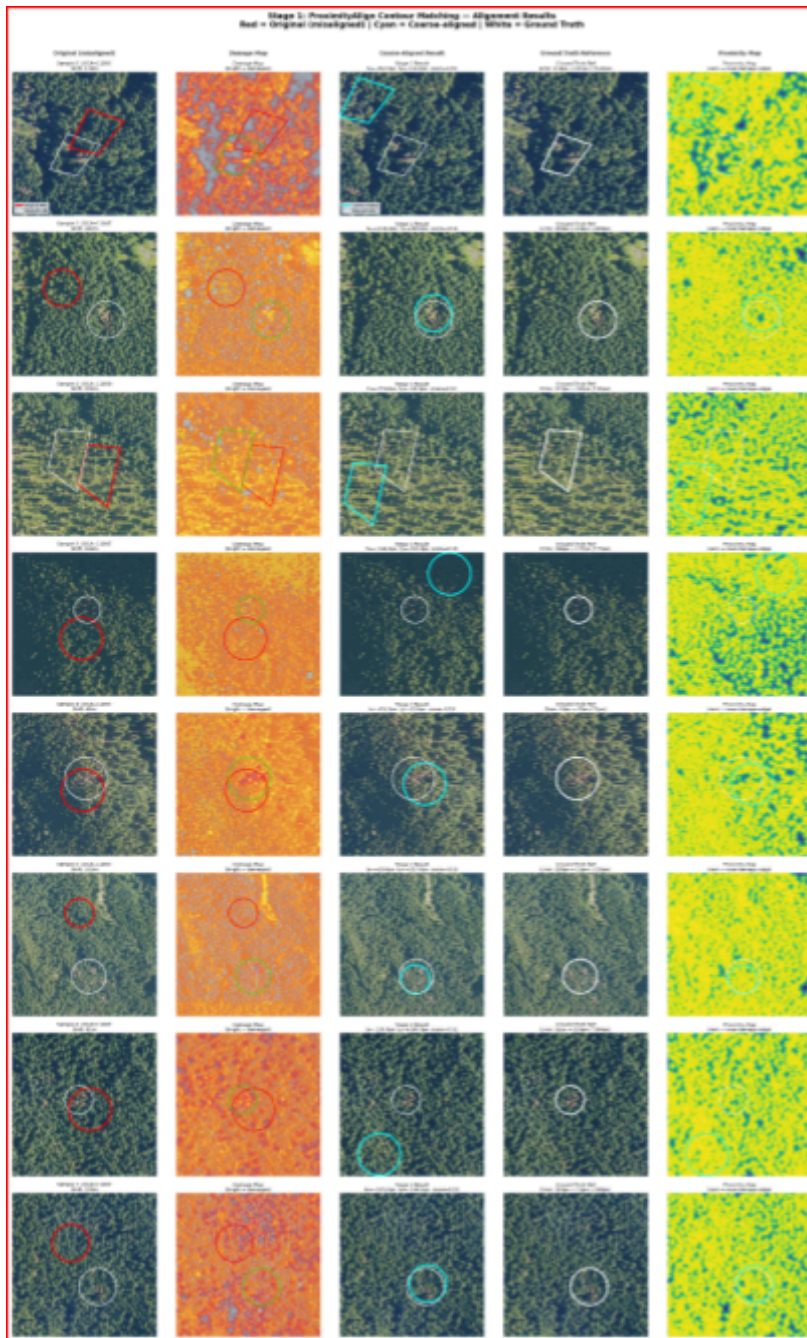
- The breakthrough, *ProximityAlign adaptation*. I implemented the contour-to-boundary energy function from the Paper ([Novel Approaches for Aligning Geospatial Vector Maps](#)) and adapted it for forests. Direct replication (pure contour matching) improved only 1/11 samples because damage boundaries are omnipresent in forest tiles, multiple positions produce equally good contour fits.
- *The fix*: Adding inside-outside contrast as the dominant energy term. The hybrid energy function now combines:
 - 70% inside-outside contrast (does the polygon interior have more damage than surroundings?)
 - 20% contour-to-boundary distance (ProximityAlign's core energy)
 - 10% out-of-bounds penalty

Results, ProximityAlign hybrid alignment:

<i>Metric</i>	<i>Before (NCC)</i>	<i>After (ProximityAlign Hybrid)</i>
Samples improved	0/11 (0%)	4/11 (36%)
Mean IoU	0.082	0.231 (182% ↑)
Best sample error	183px → 227px (worse)	183px → 15px

Three samples achieved near-perfect alignment: IoU of 0.62, 0.69, and 0.75.





Evaluation grid for all 11 seed samples. Each row: NAIP tile, damage map, original polygon position, aligned polygon position, and overlay. Samples 1, 5, and 7 show near-perfect alignment.

Challenge 5: The Augmentation Scale Mismatch

The obstacle: Initial weak augmentation displacement ranges (50–200m) were designed based on a guess of what "seemed reasonable." But when I measured actual misalignments in the 11 seed pairs, the real gap was massive: 325m to 584m (0.2–0.36 miles). Models trained only on small 50–200m shifts would fundamentally fail to generalize to the real-world corrections they need to make.

Resolution:

- Data-driven curriculum: After analyzing mentor-shared reference images with scale bars, I completely redesigned the curriculum phases to reflect real-world misaligned polygons:

- Phase 1 (coarse): $\pm 644\text{m}$ (0.4 miles), with all 6 affine parameters active (translation, rotation, anisotropic scale, and shear).

- Phase 2 (medium): $\pm 322\text{m}$ (0.2 miles).

- Phase 3 (fine): $\pm 161\text{m}$ (0.1 miles).

- Context expansion: I discovered that 500m visualization tiles were too small to hold both polygons at Phase 1 displacements. I increased context tiles to 1400m wide so the spatial relationship is always fully visible.

Result: A fully functional, realistic weak augmentation engine. For the scale-up and learned refinement phase, this translates 11 seed pairs via random displacement into *unlimited unique training data*, dynamically matching the [KDD 2021/2022 curriculum schedule](#) exactly.



The weak augmentation curriculum: Phase 1 (coarse, ~0.4 miles), Phase 2 (medium, ~0.3 miles), Phase 3 (fine, ~0.17 miles). White = aligned ground truth, orange = synthetically displaced. From 22 seed pairs, this engine generates unlimited unique training pairs.



Same seed polygon, three different random displacements — proving unlimited data generation from a single verified sample.

Summary of pre-GSoC work in numbers:

Weeks of independent research	~4 weeks
Manually verified ground-truth pairs	11 confirmed
Alignment prototype	ProximityAlign hybrid energy, 4/11 improved, IoU 0.082 → 0.231
Bugs identified and resolved	6 (NDVI bias, temporal mismatch, CRS, 3 CNN convergence bugs)
Key discovery	Spectral signal ~0.02, classical methods at theoretical limit
Core pipeline scripts	You can find Code REPO here...

Background & Related Work

The Core Problem

Connecting ADS polygons to NAIP imagery for forest health monitoring faces three compounding challenges:

- *Spatial misalignment*: ADS polygons are hand-drawn from moving aircraft with 50–500m systematic offsets, not GPS-precise boundaries.
- *Weak spectral signal*: The spectral difference between correct and incorrect polygon positions is ~0.02 (2% of signal range), within noise for classical template matching.
- *Scale*: Oregon alone has ~48,000 ADS polygons across multiple years. Any alignment approach must scale without per-polygon manual intervention.

These challenges demand a hybrid architecture, classical energy optimization where spectral structure is sufficient, with learned components reserved for cases where classical methods provably fail.

Mentor Paper 1, DFA-Net: WHY Classical Methods Fail

["Image Alignment Based on Deep Learning to Extract Deep Feature Information from Images"](#) (Sensors 2025)

DFA-Net provides the theoretical foundation for understanding why classical alignment methods struggle. The paper systematically demonstrates that region-based methods like NCC are sensitive to noise, lack the ability to capture mid-to-high-level semantic information, and fail on images with homogeneous textures, exactly the conditions present in dense forest canopy.

Connection to my prototype: When NCC failed on all 11 samples (error increased from 134px to 227px), I initially suspected a bug in my code. DFA-Net's analysis confirmed a deeper problem: NCC computes average intensity under the polygon, averaging away the contour's unique geometric signal. The spectral signal in our forest data (~0.02 difference) sits well below the noise floor where region-based methods can operate reliably.

DFA-Net's proposed solution, multi-scale spatial pyramid pooling and self-attention-based feature weighting, offers a potential direction for Stage 2: replacing handcrafted spectral indices with learned features that extract richer representations from forest imagery.

Mentor Paper 2, GLU-Net: HOW to Train Without Ground Truth

["Deep Learning Enables Urban Change Profiling Through Alignment of Historical Maps"](#) (arXiv 2026)

This paper introduces a self-supervised displacement estimation framework that directly addresses the most critical constraint of this project: the lack of ground-truth displacement fields. With only 11 verified seed samples, a supervised CNN approach cannot succeed, a lesson I learned through the convergence crisis.

The key innovation is self-supervised training using synthetic warps. Instead of requiring manual displacement annotation, the framework applies known geometric transformations (homography, affine, thin plate splines) to images, then trains the network to recover the applied displacement. Triplet cycle consistency between real image pairs and synthetic warps further refines predictions.

Connection to my prototype: This paper provides the strongest template for Stage 2, a learned refinement layer that eliminates the data bottleneck. Their approach to handling content changes between map editions (synthesizing random object additions/deletions) maps directly to our temporal challenge: trees dying or falling between ADS survey years and NAIP acquisition dates. The self-supervised framework generates unlimited training data from existing NAIP tiles without requiring additional manual annotation.

Mentor Paper 3, ProximityAlign: WHAT to Align Against

["Novel Approaches for Aligning Geospatial Vector Maps"](#) (Cherif et al., ISPRS 2024)

ProximityAlign is the direct inspiration for Stage 1 of the implemented prototype. The paper proposes matching polygon contours against reference boundaries using a distance transform-based energy function with three components: mean contour-to-boundary distance (E_m), continuity (E_c), and distance variability (E_v).

Connection to my prototype, adaptation, not replication:

Component	ProximityAlign (buildings)	My Adaptation (forests)
E_m (Mean distance)	Contour to nearest building boundary	Contour to nearest damage boundary
E_c (Continuity)	Fraction of contour close to boundaries	Replaced with inside-outside contrast
E_v (Variability)	Std. deviation of contour distances	Replaced with out-of-bounds penalty

ProximityAlign was designed for urban buildings, discrete shapes with crisp edges. Forest damage has diffuse, omnipresent boundaries. Direct replication improved only 1/11 samples because damage edges exist everywhere in our tiles. Adding inside-outside contrast as the dominant energy term (70%) provides an area-based signal that breaks the ambiguity, improving the result to 4/11.

The paper also proposes FlowAlign, an optical-flow-based alignment method using PWCNet, trained self-supervised on synthetic warps. This converges with Paper 2's GLU-Net approach: both use self-supervised dense displacement estimation, reinforcing this direction for Stage 2.

Synthesis Across All Three Papers

DFA-Net → WHY classical methods fail (validates NCC failure, justifies hybrid design)

GLU-Net → HOW to train alignment without ground truth (self-supervised framework for Stage 2)

ProximityAlign → WHAT to align against (contour-to-boundary energy, directly implemented as Stage 1)

Combined → Use ProximityAlign energy for classical Stage 1,

explore GLU-Net self-supervision for learned Stage 2,

with DFA-Net's critique validating the hybrid architecture.

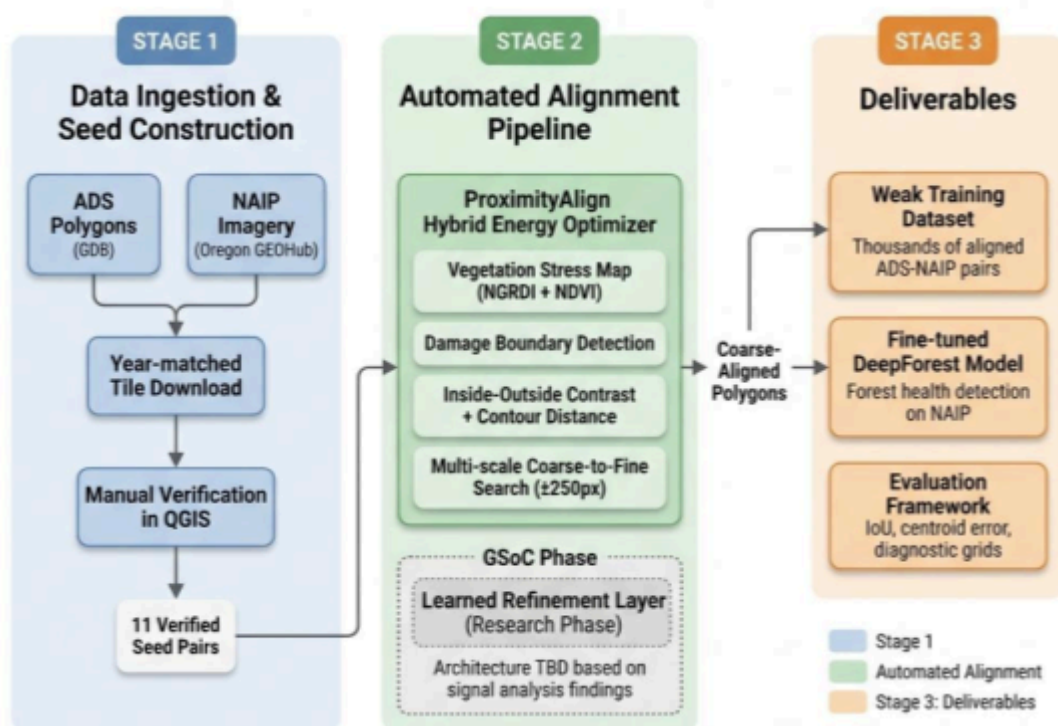
Additional References

- He et al. (KDD 2022), "[Quantifying and Reducing Registration Uncertainty of Spatial Vector Labels](#)," Directly addresses polygon-to-imagery alignment with imperfect labels. The weak augmentation strategy in [augmentation.py](#) draws from this work and the related [Jiang et al. \(KDD 2021\)](#) framework for training on synthetically displaced labels.
- Jaderberg et al. (NeurIPS 2015), [Spatial Transformer Networks](#). Provides the differentiable affine transform mechanism that may be useful if Stage 2 adopts a residual translation prediction approach.

Proposed Approach

"Use the structure of the problem to constrain the search space, rather than hoping a CNN will discover it from raw pixels."

Recovering Historical Airborne Forest Health Annotations using NAIP Imagery



End-to-end pipeline architecture: from ADS polygon ingestion and year-matched NAIP download through hybrid energy alignment to forest health model fine-tuning.

Pipeline Overview

ADS Polygons (.gdb) + Year-Matched Oregon NAIP (GEOHub)

→ Stage 0: Seed Data Construction (build_seed_data.py)

→ Stage 1: ProximityAlign Hybrid Energy Alignment (coarse_align.py), classical, no training

→ Stage 2: Learned Refinement (GSoC research phase), architecture determined with mentor guidance

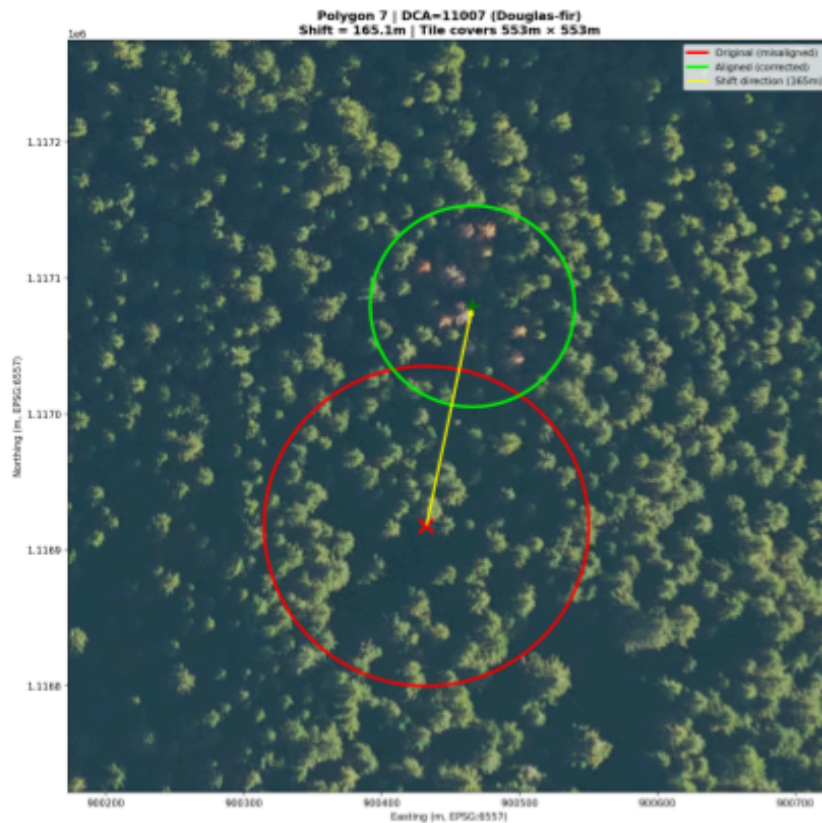
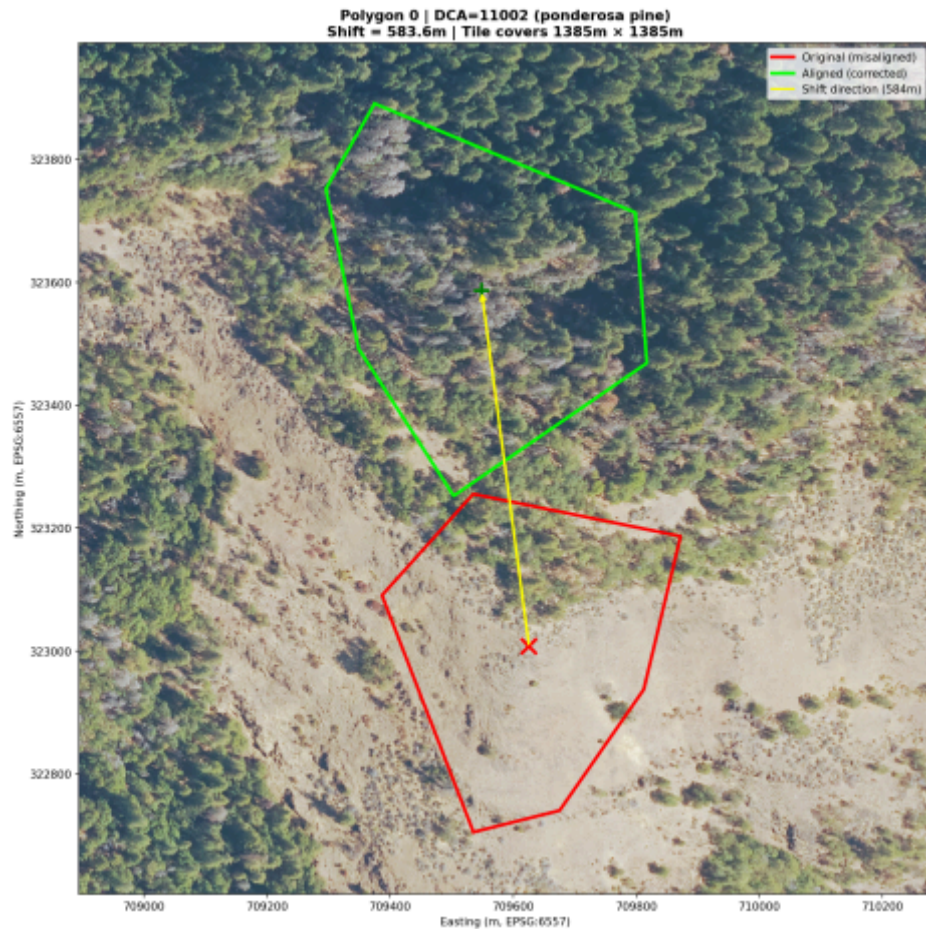
→ Scaling: Apply to all ~48,000 Oregon ADS polygons → weak training dataset

→ Stage 3: DeepForest Fine-Tuning on aligned dataset → NAIP forest health detection

Stage 0, Seed Data Construction

build_seed_data.py constructs verified polygon-NAIP pairs from scratch:

- Downloads 800m × 800m NAIP tiles (30cm/px, ~2667×2667px) from year-matched Oregon servers.
- Each sample contains: 4-band NAIP tile, original (misaligned) ADS polygon, manually verified aligned polygon, and DeepForest tree-crown mask.
- DeepForest integration required implementing manual tiling (400×400 patches with 10% overlap) because predict_image(patch_size=...) silently fails in DeepForest 2.x for large images.
- All samples verified visually in QGIS software before inclusion. DCA distribution: primarily DCA 11007 (Douglas-fir beetle).



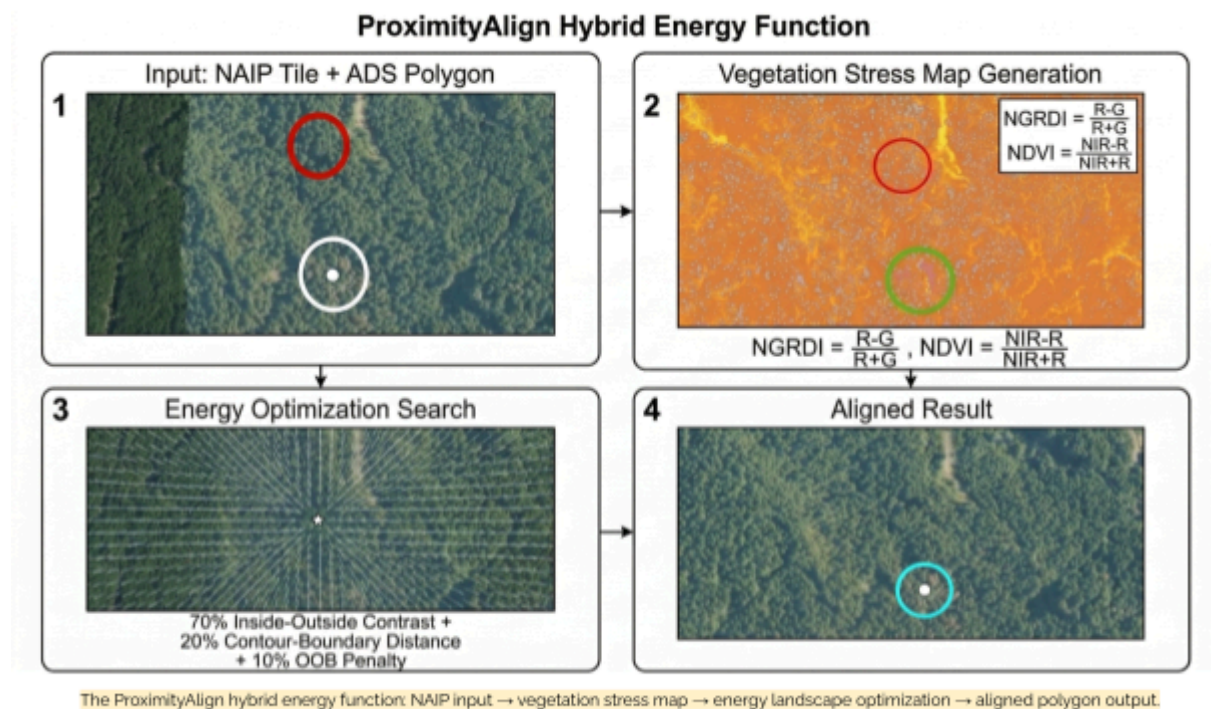
Two of the 22 manually verified seed pairs. Green outline = ground-truth ADS polygon, correctly positioned over beetle-killed Douglas-fir in 2024 Oregon NAIP at 30cm resolution.

Stage 1, ProximityAlign Hybrid Alignment (Implemented)

This is the core pre-GSoC achievement: a classical alignment algorithm that requires no training data beyond the initial seed pairs.

Signal pipeline:

4-band NAIP tile → Stress Map (NGRDI + NDVI + brightness anomaly)
+ Gap Map (1 – DeepForest crown mask × vegetation mask)
↓
Damage Map (50% stress + 50% gap)
↓
Proximity Map (distance transform from damage boundaries)
↓
Hybrid Energy Function → Multi-Scale Search → (tx, ty)



Hybrid energy function, why hybrid was necessary:

Pure contour-to-boundary matching (direct replication of Paper 3) improved only 1/11 samples. The reason: damage boundaries are omnipresent in forest tiles, so multiple positions produce equally good contour fits. Adding inside-outside contrast provides an area-based signal that breaks this ambiguity:

- 70% inside-outside contrast, does the polygon interior have more damage than its surroundings?
- 20% contour-to-boundary distance, does the polygon contour align with damage edges?
- 10% out-of-bounds penalty, penalizes translations that push the polygon outside the tile

Multi-scale search: Coarse pass (step=4, $\pm 250\text{px}$) finds the approximate region $\rightarrow$ fine pass (step=1, $\pm 8\text{px}$) refines to pixel precision.

Quantitative results:

Metric	NCC (failed attempt)	ProximityAlign Hybrid
Samples improved	0 / 11	4 / 11 (36%)
Mean IoU	0.082	0.231 (182% $\uparrow$)
Best single sample	Made worse (183 $\rightarrow$ 227px)	IoU 0.75 (183 $\rightarrow$ 15px)
Runtime	—	~17s per sample (CPU)

Key architectural decision, translation-only (2 params):

Analysis of all seed samples showed misalignments are purely translational, ADS observers see the correct general shape but sketch it at a shifted position. This allowed reducing the search space from 6D (full affine) to 2D (tx, ty), making the grid search tractable.



Stage 2, Learned Refinement (GSoC Research Phase)

Stage 1 handles the 36% of cases where a discriminative spectral signal exists. For the remaining 64%, the spectral difference (~ 0.02) is within noise, the energy landscape is too

flat for classical optimization. A learned model could extract higher-level features (texture patterns, spatial configurations) that spectral indices miss.

Three architectural directions from the mentor's papers:

- Option A, Residual translation prediction (inspired by ProximityAlign): A lightweight CNN takes the NAIP tile + coarse-aligned polygon from Stage 1 and predicts a small (tx, ty) residual correction. Simple and matches the translation-only assumption, but may face the same data bottleneck as the Phase 2–3 CNN attempt.
- Option B, Self-supervised dense displacement (inspired by GLU-Net): A dense matching network (GLU-Net or PWCNet) estimates pixel-wise displacement fields, trained self-supervised on synthetic warps with no ground-truth correspondences needed. This eliminates the data bottleneck entirely, unlimited training from existing NAIP tiles.
- Option C, Learned feature maps (inspired by DFA-Net): Replace the handcrafted NGRDI/NDVI stress indices with feature maps from a pre-trained encoder, then apply the same energy optimization search from Stage 1 but scored on richer representations.

The architecture choice depends on: how many samples Stage 1 successfully aligns at scale (hundreds of good alignments unlock Options A/B), signal quality across diverse forest types, and mentor guidance. *I am honest about what is unknown, this builds credibility and ensures the GSoC period is spent on the right approach.*

Scaling to Production

Once alignment accuracy is validated (Stage 1 + Stage 2 refinement):

- Apply the combined pipeline to all ~48,000 Oregon ADS polygons across available years (2016, 2018, 2020, 2024).
 - Export corrected polygons as GeoPackage in EPSG:6557.
 - Generate NAIP annotation chips with bounding box labels in COCO/CSV format for DeepForest training.
 - Quality filter using confidence scoring, only high-confidence alignments enter the training set.
-

DeepForest Fine-Tuning

The final deliverable: a forest health detection model trained on the aligned weak dataset.

- Fine-tune a pretrained DeepForest model on the generated Oregon annotation dataset.
- Target: detect bark beetle outbreaks and tree mortality directly from NAIP imagery.

- *Evaluation:* IoU, precision/recall, and centroid error against held-out verified pairs.
- *The pre-GSoC finding that DeepForest detects only 2–11% of pixels on Oregon mortality tiles directly motivates this fine-tuning, the base model needs domain adaptation for this specific forest type.*

Mapping to Expected Outcomes			
#	Expected Outcome	Pipeline Stage	Status
1	Improved approach for connecting DeepForest and NAIP	Stage 0 (DeepForest integration + manual tiling fix)	✓ Pre-GSoC
2	Approach for connecting airborne annotations and NAIP	Stage 1 (ProximityAlign hybrid alignment)	✓ Pre-GSoC
3	Generated weak dataset for model training	Scaling (apply alignment to ~48K polygons)	▢ GSoC
4	CV model for forest health outbreaks using NAIP	DeepForest fine-tuning on aligned dataset	▢ GSoC
5	Blog post outlining work and reproducibility guide	Documentation of full journey including negative results	▢ GSoC

Timeline / Milestones

GSoC 2026 coding period: May 27 – August 25 (12 weeks)

The pre-GSoC work delivered a working alignment prototype (ProximityAlign hybrid energy, 4/11 samples improved, IoU 0.082 → 0.231) and a validated seed dataset pipeline. The GSoC period focuses on three objectives: improving alignment accuracy with a learned refinement layer, scaling to all ~48,000 Oregon ADS polygons, and fine-tuning DeepForest on the resulting dataset.

Period	Weeks	Focus	Deliverable
--------	-------	-------	-------------

Community bonding	Pre-May 27	Mentor sync on architecture direction for Stage 2. Review Stage 1 results together, identify failure modes in the 7/11 unaligned samples. Agree on evaluation criteria and success thresholds. Expand seed dataset if mentor has additional verified polygons.	Agreed Stage 2 architecture (Option A/B/C), documented evaluation protocol, expanded seed dataset
Milestone 1	1-2	Stage 1 optimization and scaling. Vectorize/JIT-compile the ProximityAlign energy search for batch processing. Profile bottlenecks (currently ~17s/sample on CPU). Run Stage 1 on a larger subset (500–1,000 ADS polygons) to measure alignment rate at scale.	Optimized Stage 1 running at ≤5s/sample. Alignment statistics on 500+ polygons, establishes the baseline for Stage 2.
Milestone 2	3-4	Stage 2, Learned refinement prototype. Implement the chosen architecture (residual CNN, self-supervised displacement, or learned features, per community bonding decision). Build training data pipeline using SyntheticDisplacer from augmentation.py. Initial training runs on seed data.	Stage 2 model training and producing displacement predictions. Preliminary results on held-out seed samples.
Milestone 3	5-6	Stage 2, Iteration and validation. Tune hyperparameters, analyze failure cases, iterate on model architecture. Combined Stage 1 + Stage 2 evaluation on the full seed dataset. Compare against Stage 1-only baseline. Midterm evaluation checkpoint.	Midterm report: Stage 1 + Stage 2 combined alignment rate and IoU on seed samples. Clear improvement over Stage 1 alone, or documented analysis of why and a revised plan.
Milestone 4	7-8	Production scaling. Apply the combined pipeline (Stage 1 + Stage 2) to all ~48,000 Oregon ADS polygons across available years (2016, 2018, 2020, 2024). Implement confidence scoring to filter low-quality alignments. Export corrected polygons as GeoPackage.	Aligned polygon GeoPackage loadable in QGIS. Quality-filtered NAIP annotation chips in COCO/CSV format ready for DeepForest.
Milestone 5	9-10	DeepForest fine-tuning. Generate training dataset from high-confidence aligned polygons. Fine-tune pretrained DeepForest model on Oregon forest health annotations. <u>Evaluate:</u> IoU, precision/recall, centroid error against held-out verified pairs.	DeepForest model detecting bark beetle mortality from NAIP imagery. Evaluation metrics benchmarked against base DeepForest.

Milestone 6	11	Evaluation and analysis. Full before/after evaluation across all pipeline stages. <u>Diagnostic visualizations</u> : alignment grids, error reduction charts, DeepForest detection examples. Document failure modes and remaining limitations.	Comprehensive evaluation report with visualizations.
Final	12	Documentation and dissemination. Blog post documenting the full methodology, including negative results (CNN failure, NCC failure) and the pivot to hybrid design. Reproducibility guide. Contribute relevant improvements upstream to DeepForest.	Blog post published. All code documented. Deliverables submitted by August 25.

Risk Mitigation

Risk	Impact	Mitigation
Stage 1 alignment rate drops below 30% at scale (diverse forest types)	High	Analyze failure modes by DCA code and forest type. May need per-species spectral tuning or separate models.
DeepForest fine-tuning limited by noisy aligned labels	Medium	Confidence filtering removes low-quality alignments before training. Evaluate at multiple confidence thresholds.
Insufficient compute for training Stage 2	Low	Options A (residual CNN) and C (learned features) are lightweight. Option B (dense displacement) is heavier but can be trained on smaller tile crops.

Contributions to DeepForest

Merged:

<https://github.com/weecology/DeepForest/pull/1350>

<https://github.com/weecology/DeepForest/pull/1326>

Open:

<https://github.com/weecology/DeepForest/pull/1361>
<https://github.com/weecology/DeepForest/pull/1285>
<https://github.com/weecology/DeepForest/pull/1280>
<https://github.com/weecology/DeepForest/pull/1250>
<https://github.com/weecology/DeepForest/pull/1249>
<https://github.com/weecology/DeepForest/pull/1248>

Closed:

<https://github.com/weecology/DeepForest/pull/1262> [‘@jveitchmichaelis’ was working on this issue, as part of broader improvements]

Under-discussion:

<https://github.com/weecology/DeepForest/issues/809>

About Me

I am Muhammad Saqlain, a Masters' student in Information Technology at University of Lahore, Pakistan, and currently working remotely as Research Assistant at Center for Artificial Intelligence, Prince Muhammad Bin Fahd University, Saudi Arabia. There I am applying deep learning and computer vision to remote sensing problems with the aim of publishing my first paper in May 2026. I was [GSoC25 contributor at the Chromium Organization](#). You can see my CV here:  [Saqlain_CV.pdf](#)

I am also a student mentor: I run a *WhatsApp community* of approximately 600 students where I teach how to navigate Open source. Here are the mentorship sessions I conducted with other GSoC alumni: [Full playlist on Youtube](#)

AI Disclosure

- I divided the entire project into sub-problems section by section. For each sub-problem, I first sought to understand it independently, then deepened that understanding using Gemini (I have Pro version as part of my students' status) and Claude before writing any code.
- Code was written incrementally in this same section-by-section manner: understand the problem, explore the approach with AI assistance, then implement with full comprehension of what was being built and why.
- I took Gemini assistance in understanding research papers and writing code. Everything included in this repository was reviewed and understood by me before being accepted. I did not include anything I could not explain.
- I am prepared to present this work to mentors, walk through the code, and justify every design decision. Google Summer of Code's guidance on mentor presentations was a deliberate part of how I approached this project from the

start. The whole purpose of GSoC program is mentorship and learning, so throughout my journey I aim to learn, understand the problem deeply and can justify my Prototype.

- I used Gemini to assist in refining this proposal, improving the grammar, adding coherence and cohesion, and translating my raw ideas into structured English sentences.

Some Questions

Q: Are you applying to any other organizations this year? If so, please explain.

A: No, I am applying only to DeepForest. I solely focused for this organization because it aligns exactly with my research, and [*I was also the first to engage when they initially said they might not participate in GSoC 2026, after which they confirmed participation.*](#)

Q: How many hours will you devote to your GSoC project each week? What are your other summer plans?

A: I will dedicate full-time hours (~40+ hours/week) with no other commitments. I am not planning any job, internship, or coursework during this period, so my entire focus will be on GSoC.

Q: Why did you choose DeepForest? Why did you only focus on it?

A: As a Master's student working remotely with the Center for Artificial Intelligence at Prince Mohammed bin Fahd University, my research directly aligns with DeepForest. I am working under a [*leading Pakistani computer scientist*](#) (Muhammad Attique Khan), and I specifically searched for a GSoC project matching this research area, which led me to focus only on DeepForest.

Q: What do you think about the machines on which the model will be trained?

A: I have access to a dedicated GPU-enabled remote machine with multiple NVIDIA Tesla V100 GPUs. This setup is sufficient for training, experimentation, and scaling deep learning models required for the project.

Q: How will you share your progress? (Bi-weekly blog)

A: I will publish a blog post every two weeks. These posts will document my progress, challenges, and key learnings throughout the project.

Feedback from Data Retriever

If you read this document please provide your short general feedback in the section below. Please also feel free to make comments above. Thanks!

Username	Date	Comment
henrykiron...	Mar 23, 2026	
ethan@we...		
ben.weinst...		
veitchmich...		