

ONE HACK 2026

AI Track · Problem Statements

8-Hour Hackathon by Hackers Cult · 13 September · NSUT Delhi · Intermediate–Advanced Level

Each problem below names a specific hard sub-problem, not just a general theme — solutions that skip the hard part will not meet the bar. None of these have a live data feed realistically available in an 8-hour window (live satellite tasking, live orbital tracking, live drone footage), so each is explicitly scoped to public/historical datasets or clearly labeled simulated data. Teams are expected to demo a working prototype against the stated core requirements, not a slide deck.

AI-1. Satellite-Based Real-Time Natural Disaster Prediction

Domain: Remote Sensing / Geospatial AI

Background

Floods, cyclones, and landslides cause massive damage in high-risk regions, and early-warning systems that rely on sparse ground sensors often miss or lag the event. Satellite imagery offers wide, repeated coverage but is underused for genuinely predictive (not just after-the-fact) risk mapping because turning raw satellite bands into an actionable spatial risk score is non-trivial.

Problem Statement

Build a model that uses satellite data to predict flood, cyclone, or landslide risk in a chosen high-risk region, producing an actionable spatial risk map rather than a single classification score. Use a public satellite dataset and a defined historical time window — not a live feed.

Core Requirements

- Pick exactly one hazard type (flood OR cyclone OR landslide) and one specific region — do not attempt all three generically.
- Use a publicly available satellite dataset (e.g. Sentinel Hub/Copernicus, NASA MODIS/Landsat, ISRO Bhuvan open data) and disclose which one, its resolution, and its revisit time.
- Go from raw/preprocessed satellite bands relevant to the hazard (e.g. NDWI for flood extent, cloud-top temperature for cyclone intensity, slope + precipitation for landslide susceptibility) to a spatial risk score across the region — not a single number for the whole area.
- Validate the model against at least one real historical disaster event present in the dataset's time window: does the model's risk score actually rise before or during that known event?
- Produce a clear visual output — a risk heatmap overlaid on a map of the region — and be explicit about the lag between a satellite pass and your prediction (this is early-warning value from repeated passes, not literal real-time streaming).

Bonus / Stretch Features

- Fuse in a second public data source (rainfall gauge data, elevation/DEM data, or historical disaster records) for a multi-modal risk score.
- Add a simple threshold-based alert system (e.g. flag when predicted risk crosses a set level).

Suggested Tech Stack

Python, Google Earth Engine or Sentinel Hub API, rasterio/GDAL, PyTorch/TensorFlow (CNN or U-Net for spatial prediction), Folium/Leaflet for map visualization.

AI-2. Space Debris Detection & Collision Avoidance AI

Domain: Aerospace / Orbital Mechanics + AI

Background

The growing debris field in Earth orbit creates real collision risk for active satellites. Assessing that risk requires physics-based orbit propagation as a baseline, with an additional layer to flag genuinely dangerous close approaches and suggest a response — not just raw trajectory math.

Problem Statement

Build an AI-assisted system that ingests publicly available orbital data for a defined set of objects, propagates their trajectories, predicts potential close approaches/collisions between them, and recommends a concrete avoidance maneuver for at least one at-risk satellite.

Core Requirements

- Use publicly available TLE (Two-Line Element) data for a defined set of tracked objects (e.g. from CelesTrak or Space-Track.org) — do not simulate orbital physics from scratch.
- Implement or use an established propagation model (e.g. SGP4) to project future positions of the chosen objects — this is the physics baseline the AI/risk layer sits on top of.
- Build a risk-scoring layer on top of the propagated trajectories that flags close-approach events above a defined threshold — an ML-based anomaly/risk classifier, or a probability-of-collision calculation refined with learned corrections.
- For at least one flagged high-risk conjunction, compute and justify a concrete recommended avoidance maneuver (delta-v magnitude/direction or timing) and show the resulting increase in miss-distance.
- Present a clear before/after visualization: the predicted trajectory intersection versus the corrected trajectory after your suggested maneuver.

Bonus / Stretch Features

- Rank multiple simultaneous conjunction events by urgency/severity.
- Estimate the fuel-cost tradeoff of the maneuver against the collision probability it avoids.

Suggested Tech Stack

Python, Skyfield or python-sgp4, CelesTrak/Space-Track APIs, NumPy/SciPy for orbital mechanics, Plotly or Three.js for 3D orbit visualization, optionally scikit-learn/PyTorch for the risk-scoring layer.

AI-3. AI-Based Border Surveillance Using Drones

Domain: Computer Vision / Security Tech

Background

Manual border monitoring is manpower-intensive and prone to fatigue-driven misses, especially at night. Drones equipped with thermal or low-light cameras could provide continuous automated monitoring, but the real challenge is distinguishing genuinely suspicious human activity from routine wildlife, vehicles, or environmental noise — not simply detecting "something is there."

Problem Statement

Build a computer vision model that processes thermal and/or low-light video to detect and flag suspicious human movement patterns in a defined perimeter, distinguishing them from non-threat activity. Use public thermal/low-light datasets or footage — not live drone/border camera feeds, which are not accessible in an 8-hour hackathon.

Core Requirements

- Use a publicly available thermal or low-light video/image dataset (e.g. FLIR ADAS Thermal Dataset, OTCBVS, or a similar open thermal-imaging dataset) and clearly disclose which one and its limitations.
- Detect and track human figures across a video sequence, not just classify single frames — must handle motion over time, including partial occlusion.
- Distinguish at least 3 categories of movement pattern (e.g. normal walking, crouching/crawling to avoid detection, group movement) versus non-human heat sources (animals, vehicles) — a model that only outputs "human detected" does not meet the bar.
- Explicitly address thermal/low-light-specific challenges: thermal blooming, low resolution, and the absence of color/texture cues that RGB-trained models rely on.
- Produce a bounding-box-and-alert overlay on the video output with a confidence/suspicion score, not just a terminal log.

Bonus / Stretch Features

- Fuse multiple camera angles/feeds for a single tracked target.
- Add false-alarm reduction logic that learns to filter out recurring wildlife/livestock patterns from the dataset.

Suggested Tech Stack

Python, OpenCV, YOLOv8/v9 (or a thermal-adapted detector), DeepSORT/ByteTrack for multi-object tracking, PyTorch.

Notes for Participants

- All problem statements are scoped to be buildable by a team of 1–4 within the 8-hour window: a working core build plus a minimal frontend/demo interface, not a finished product.
- A working, narrower prototype will consistently outscore a broad, non-functional one. If you're running out of time, cut scope early — a half-built, well-defended solution beats a full mockup.
- Teams should be ready to explain and defend their core technical design decisions to judges, not just walk through the demo happy-path.
- All satellite, orbital, and drone/thermal datasets used must be public and clearly cited in the final write-up — teams are judged on modeling and detection quality against real (if historical) data, not on data acquisition.