ChatGPT

Day 4, Session 3: Emerging AI/ML Trends in Earth Observation (GeoFMs, SSL, XAI)

Learning Objectives

- **Understand Geospatial Foundation Models (GeoFMs):** What foundation models are, how they are pre-trained on massive unlabeled Earth data, and why they enable powerful transfer learning for EO tasks [1] [2]. Identify examples like Prithvi, Clay, SatMAE, and DOFA and their potential applications.
- **Grasp Self-Supervised Learning (SSL) for EO:** Learn how models can **train themselves** using *pretext tasks* (e.g. masked autoencoding, contrastive learning) to leverage abundant unlabeled satellite images. Recognize EO-specific SSL techniques (masking spectral bands, patch prediction, temporal augmentation) and why SSL is valuable for data-scarce Philippine contexts [3] [4].
- **Appreciate Explainable AI (XAI) in EO:** Understand the need for model interpretability in Earth Observation – for debugging, transparency, and accountability in critical applications. Get introduced to XAI methods (SHAP, LIME, Grad-CAM) and see examples of how they explain model decisions (e.g. feature importance plots, saliency heatmaps) to build trust in AI results.

Module 1: Introduction to Foundation Models in EO

What are Foundation Models? Foundation models (FMs) are very large AI models **pre-trained on broad, unlabeled data** at scale, then fine-tuned for specific downstream tasks [2]. In Earth Observation, GeoFMs ingest huge amounts of satellite imagery (potentially petabytes) and learn general representations of Earth's surface. They **"distill and synthesize"** diverse environmental patterns into a versatile understanding of our planet [1]. As a result, a single GeoFM can be adapted to many tasks (land cover mapping, disaster detection, crop yield prediction, etc.) without training each from scratch. Crucially, these models leverage self-supervised learning during pre-training (no human labels needed), for example by *filling in masked portions* of images [5] [6]. This makes them extremely powerful "generalists" – **like an AI that has learned to *"see"* the Earth in general**, which we can then specialize for a specific job with relatively little additional data [7]. In practical terms, using a pre-trained GeoFM can save tremendous labeling effort and achieve strong performance even with limited new training samples [7].

Examples of GeoFMs: Several foundation models for geospatial data have recently emerged:

- **Prithvi** (NASA/IBM) – ~300M-parameter Vision Transformer model pre-trained on NASA's Harmonized Landsat-Sentinel-2 (HLS) imagery [8]. Prithvi learns by masking parts of satellite images and predicting them [5]. It embeds NASA's remote sensing expertise into a model that can be fine-tuned for tasks like land-use change detection, burn scar mapping, flood mapping, and crop type classification [9] [10]. (It's described as *"like having a powerful assistant"* analyzing petabytes of data to yield insights [11].)
- **Clay** (Radiant Earth) – An open-source GeoFM (~500M parameters) that uses a Vision Transformer with special geospatial tokens (location & time) [12]. Clay is trained via Self-Supervised Learning (masked autoencoding) on multi-sensor imagery (Sentinel-1 SAR, Sentinel-2 optical, Landsat 8/9,

NAIP aerial, etc.), producing rich **embeddings** of any location+time on Earth [12] [13]. These embeddings can be used for downstream tasks by fine-tuning small models on top. Clay's goal is to provide general-purpose representations for nature and climate applications, and it's **openly available** for adaptation [14] [15].

- **SatMAE** (Stanford) – A 300M-parameter foundation model designed specifically for multi-spectral **and** temporal satellite data [16]. SatMAE extends the Masked Autoencoder (MAE) approach to satellite imagery by encoding multi-band inputs with spectral position embeddings and masking patches across time [17]. By reconstructing missing pixels, it learns powerful features that boosted land-cover classification accuracy by up to 14% over previous methods [17]. In essence, SatMAE learns both the *spectral* signatures and *temporal* dynamics of EO data via self-supervision.
- **DOFA** (Dynamic One-For-All) – A cutting-edge multimodal foundation model that **adapts to various sensor types** (optical, radar, hyperspectral) within one unified network [18]. Inspired by neural plasticity, DOFA uses a dynamic hypernetwork to adjust to different input modalities on the fly [18]. It was jointly trained on data from five sensors and can handle tasks across 12 Earth observation datasets, even ones not seen during training [19]. This "jack-of-all-trades" approach shows how foundation models are pushing toward *universal* EO models that integrate all data sources [18].

How GeoFMs Work – Pretraining and Fine-Tuning: A GeoFM is typically trained in two phases. **First, pretraining:** the model ingests vast unlabeled datasets (e.g. global satellite archives) and learns general features via a self-supervised objective (for instance, masking 50% of input pixels and learning to reconstruct them, as in MAE [4]). After pretraining, the model can produce **embedding vectors** that capture high-level information about any input image (like latent features describing land cover, textures, etc.). **Second, fine-tuning:** for a specific task, we take the pretrained model (or its embeddings) and train a smaller head or adjust the model weights using a labeled dataset for that task [20] [21]. Because the GeoFM already learned general Earth features, fine-tuning requires far fewer labeled samples than training from scratch [7]. This is especially useful for countries like the Philippines, where labeled EO data might be limited for certain themes. We can fine-tune a GeoFM on as few as dozens or hundreds of local examples to get a performant model.

A simplified GeoFM workflow: **(1)** Large-scale **pretraining** on unlabeled satellite imagery (using a method like masked autoencoding) produces a general-purpose model. **(2)** At inference, the GeoFM **encodes** new images into *embeddings* – compact vectors summarizing their content. **(3)** These embeddings power various applications: e.g. *similarity search* (find areas with similar features), *change detection* (track how an area's embedding shifts over time), or serve as inputs to downstream models [4] [22]. **(4)** For a specific task (classification, segmentation, etc.), we perform **fine-tuning** by adding a lightweight prediction head on the GeoFM and training it with a small labeled dataset [23] [24]. The GeoFM's rich knowledge means we need much less new data, and it ensures consistent performance across time and locations [25] [7].

Philippine Use Case: Imagine you want to map land cover in a Philippine province (e.g. classify rice paddies, forests, mangroves, urban areas) but you only have, say, 50 labeled samples for mangroves. A foundation model like Clay or Prithvi could be a game-changer. Because the GeoFM was exposed to global coastlines and mangrove patterns during pretraining, it has a notion of what **mangroves look like** from space. By fine-tuning it on your limited Philippine samples, the model can quickly adapt to local conditions (perhaps learning the particular spectral signature of Philippine mangroves) with high accuracy. In essence, the GeoFM transfers knowledge from other regions to jump-start your task [2] [7]. This few-shot learning capability means even small organizations in the Philippines can build competent AI models for EO tasks – from coral reef mapping to landslide detection – by leveraging these "pre-trained eyes in the sky" rather than starting blank.

Think-Through: *Why do you think a model pre-trained on global data (like one that has seen images of forests, cities, water from all over the world) might perform better in the Philippines than a model trained from scratch on Philippine data alone? Consider issues like data scarcity and diversity.*

Mini-Challenge: *Suppose you are tasked with mapping mangrove forests across all Philippine islands, but you only have a very small labeled dataset (a handful of example patches labeled as mangrove vs. non-mangrove). Outline a strategy to tackle this using a GeoFM. What steps would you take to fine-tune the foundation model (e.g. Clay or Prithvi) for mangrove detection? What kind of data preparation and additional training would you need, and how would you validate that the model works well?*

Module 2: Self-Supervised Learning in EO

What is Self-Supervised Learning? Self-Supervised Learning (SSL) is a training paradigm where the model learns from **unlabeled data** by solving made-up proxy tasks (called *pretext tasks*). The key idea is to generate supervision *from the data itself*. In computer vision, this often means hiding or perturbing part of the input and tasking the model to predict it. By doing so, the model learns useful representations without any human-provided labels [26] [3]. SSL has become a dominant approach in contexts like remote sensing where **labels are scarce but unlabeled images are plentiful** [3]. Instead of requiring thousands of annotated satellite images, we can let the model **"make its own puzzles"** from raw images and learn from solving them.

Common SSL Pretext Tasks: Two popular families of SSL tasks in EO are **masked reconstruction** and **contrastive learning**:

- *Masked Image Modeling (MIM):* This is like a high-tech jigsaw puzzle. We **mask out** a portion of an image (could be random patches, or certain pixels) and ask the model to reconstruct the missing content [27] [28]. The model must capture context from visible parts to fill in the blanks. Vision Transformers with a Masked Autoencoder approach follow this technique – e.g. in SatMAE and Clay, the model sees only ~30% of the image pixels and has to predict the rest [4]. By doing so across millions of images, the model internalizes general structure: e.g. "if the surrounding area is blue and wavy, the masked part is likely ocean." In remote sensing, Masked Modeling is especially powerful because images have a lot of redundancy and patterns (textures, spatial structures) that can be learned. It also helps with issues like clouds or occlusions – the model learns to **impute or recover** information, a bit like how we use multiple images to guess a cloudy pixel [29] [30].
- *Contrastive Learning:* This can be seen as a **matching game**. We take an image and create two different augmented versions of it (e.g. crop it, rotate, change colors slightly). The model's task is to recognize that these two views *represent the same place*, and distinguish them from images of different places [31]. Essentially, the model learns to make representations of the same scene *close* in feature space, and different scenes *far apart*. In remote sensing, contrastive methods like SeCo (Self-Supervised Contrastive Learning) use spatial and temporal augmentations – for instance, one view might be a Sentinel-2 image in dry season, another in wet season; the model should learn they're the same location despite differences [32]. Contrastive SSL leverages techniques like random rotations, flips, spectral band dropout, etc., which are particularly pertinent for satellites (e.g. a satellite image rotated 180° is still the same scene, just a different angle). By training on such objectives, the model learns invariant features (e.g. it will recognize a rice field as a rice field regardless of rotation or lighting).

EO-Specific SSL Techniques: Beyond generic tasks, Earth Observation data opens up creative self-supervised tasks:

- *Masking Spectral Bands:* Unlike natural images, satellite images have multiple spectral bands (RGB, infrared, radar, etc.). An SSL task could hide one or more **bands** and have the model predict them from the others. For example, mask the Near-IR band and let the model infer it from the RGB bands. To do this, the model must understand the relationships between spectra (e.g. vegetation is bright in NIR but somewhat green in RGB). This teaches spectral feature learning. In essence, the model learns to **"guess" a missing sensor** using the remaining data – analogous to how humans can guess what a color-infrared image would look like from a true-color photo.
- *Patch Location Prediction:* This is like a puzzle where pieces are removed or shuffled. We could cut an image into patches and remove one patch – the model must predict which patch is missing or its content (a context prediction task). Or shuffle patches and task the model to order them correctly (like solving a sliding puzzle). This forces understanding of spatial context – e.g. the model learns that *beach* usually lies next to *water*, not in the middle of a forest.
- *Temporal Consistency or Prediction:* Many EO datasets have time series (e.g. monthly images). An SSL task here: give the model frames from different times and mask one timestamp, asking it to predict the missing time's image (like future frame prediction or interpolation). Another approach is *temporal contrastive* learning: treat the same location at Time A and Time B as a positive pair (they are related), whereas different locations or very different times are negatives [32]. This way, the model learns features that are stable over time or evolves in known ways (e.g. seasonal vegetation cycles). For instance, if we have satellite images through a year, the model might learn what seasonal changes to expect for rice paddies vs. evergreen forests.

All these pretext tasks exploit the unique structure of EO data (multi-band, multi-time) to learn rich representations without labels. They effectively turn an unlabeled satellite archive into a **treasure trove of training signals**. As a concrete example, the SatMAE model mentioned earlier does both spectral and temporal masking: it randomly hides patches in each date of a time series and also encodes spectral groups separately [17] [33]. The model then reconstructs the missing parts, achieving stronger features for downstream tasks than simple ImageNet pretraining would.

Why SSL Matters (Philippine context): The Philippines has plenty of satellite data (from Diwata microsatellites, to free Sentinel-1/2 imagery, etc.), but only a small fraction is labeled for AI. SSL allows us to harness **all those terabytes of unlabeled images** to pretrain models that understand local geography. For instance, we might take a year's worth of Sentinel-2 images over the Philippines and train a model via masked autoencoding – the model would learn features like the texture of coconut plantations, the spectral signature of coral reefs, the shape of rice terraces, etc., without any human labels. Later, if we need to classify a particular phenomenon (say, *identify volcanic lahar-affected areas* or *find fishponds in coastal areas*), we can fine-tune this pretrained model with a handful of examples. The self-supervised pretraining would have given the model a "head start" by teaching it generally what Philippine landscapes look like. This is crucial for a country where labeled datasets for EO might be limited to small academic projects or specific agencies. SSL effectively **leverages quantity (unlabeled data) to make up for the lack of quality labels**. As another example, Diwata-2 images could be pretext-trained by tasking the model to predict one spectral band from the others (since Diwata has unique bands for agriculture, etc.) – thereby creating a Diwata-specific representation that could then be used for, say, crop stress detection with minimal labeled data.

Think-Through: *You have 10 years of satellite images of a certain region in the Philippines, but very few ground-truth labels. What are some creative self-supervised tasks you could design to help a model learn useful*

information from this dataset? (Hint: How could you make the model predict one part of the data from another? Consider space, time, and spectral channels.)

Mini-Challenge: *Identify an EO dataset of interest that has no labels – for example, imagery from a new satellite or drone survey in your area. Propose a self-supervised learning plan to pretrain a model on it. What specific pretext task would you use and why? (For instance, will you mask pixels, drop whole spectral bands, rotate images, create synthetic pairs…?) Describe how you would later test if the learned representation is useful (what downstream task could you fine-tune on as a proof of concept?).*

Module 3: Explainable AI (XAI) for EO

Why Do We Need Explainability? In Earth Observation applications, AI models are often used for high-stakes or decision-support tasks – e.g. identifying flood zones for disaster response, mapping deforestation for enforcement, or monitoring urban growth for planning. If these models are "black boxes," it's hard for practitioners or stakeholders to trust their outputs. We've all heard *"AI is only as good as its data"* – if a model makes a bizarre or wrong prediction, we need a way to **peek inside and understand why**. Explainable AI techniques aim to shed light on the model's reasoning: which features or parts of the image influenced its decision? This is crucial for **debugging** models (finding out if it's looking at clouds instead of floods), for **transparency** (justifying decisions to end-users or officials), and for **accountability** (ensuring the model isn't discriminating or making errors that go unchecked). In short, XAI bridges the gap between complex model predictions and human interpretability, helping build **fairness and trust** in AI deployments [34].

Let's introduce three common XAI techniques relevant to EO workflows:

- **SHAP (SHapley Additive exPlanations):** This is a model-agnostic method grounded in game theory. In simple terms, SHAP asks: *for a given prediction, how much did each feature contribute to pushing the prediction towards the outcome?* It assigns each input feature (e.g. a spectral band value, or a derived index) a Shapley value indicating its influence – positive or negative – on the model's output. For example, if you have a Random Forest model predicting "forest" vs "not forest" from spectral bands, SHAP might tell you that for a particular pixel, the NDVI feature had the highest positive contribution to the model saying "forest," while an urban index had a negative contribution [35] [36]. SHAP values can be aggregated to show global importance (which features are most important on average) or examined per sample to explain individual predictions. One nice aspect is SHAP is **consistent and local** – it explains each prediction with a set of feature attributions that sum up to the difference between the model's output and a baseline. In EO, we often use SHAP for models like gradient boosting trees or random forests used on tabular data (like classifying land cover from reflectance values). It produces intuitive visualizations – e.g. bar plots of feature importance or summary dot plots.

Example – SHAP global feature importance: This bar chart shows the top features influencing an ML model (example from a non-EO dataset). Each bar's length represents the mean absolute SHAP value of that feature over many predictions, i.e. how much that feature contributes on average [37] [38]. In an EO context, features could be things like "Green band reflectance" or "Elevation" or "NDVI index." A similar plot for a land-cover classification model might reveal, for instance, that **NDVI** and **NIR band** have the highest SHAP values (most influence) for differentiating vegetation classes, whereas **urban mask presence** might be top for built-up areas. Such insights help us verify the model is using sensible features (e.g. NDVI should matter for vegetation!). If something odd shows up (say, "Latitude" is a top feature for forest vs. urban

classification), it might hint the model is picking up a spurious correlation (maybe all your forest samples happened to be from the north). SHAP thus provides both **interpretability and a diagnostic tool**.

- **LIME (Local Interpretable Model-Agnostic Explanations):** LIME focuses on explaining a single prediction by approximating the model *locally* with a simple interpretable model. The idea: take the input (e.g. an image), perturb it in many slight ways (for images, one common approach is to segment the image into superpixels and randomly hide or alter some segments), and see how the model prediction changes [39] [40]. By observing which parts of the input most affect the prediction, LIME fits a small linear model that approximates the big model's behavior around that instance. The output might be, for an image classification, a heatmap highlighting the regions that positively contributed to a certain class. For example, if a CNN predicts a satellite image patch as "flooded", LIME could highlight the actual water-covered pixels as having the strongest positive weight to that prediction, whereas other areas (like buildings or clouds) might have negative weights. In a tabular case, LIME might say: *"For this specific prediction of high forest cover probability, the factors were: high NDVI (++, drove prediction up), low built-up index (++), low Red band reflectance (+), etc."* LIME is powerful because it doesn't require model internals – you treat the model as a black box, poke it with inputs, and see how it reacts. This is useful when you cannot easily inspect the model (say it's an ensemble or proprietary model). The downside is that LIME explanations are local and approximate – they are valid in a small neighborhood of that data point, not globally. Still, for EO, LIME can be used on any classifier or even a regression to explain individual outcomes. For instance, you could use LIME on a building footprint detection model to see which parts of the image led it to decide "building" – it might highlight the rectangular roof structures and not the surrounding vegetation, which aligns with our expectations.

- **Grad-CAM (Gradient-weighted Class Activation Mapping):** This is a technique specifically for convolutional neural networks (CNNs) commonly used in image tasks. Grad-CAM generates a **visual "heatmap"** over an input image to show which regions were most important for the model's prediction [41] [42]. It works by taking the gradient of the target class score with respect to the feature maps of the last convolutional layer, essentially finding which neurons (spatial regions in the feature map) had the largest influence on the final score [43]. These gradients are averaged to get importance weights, and a weighted sum of the feature maps (up-sampled to image size) gives us the heatmap overlay. In simpler terms: *Grad-CAM highlights "where the CNN is looking."* For an EO example, suppose we have a deep CNN that classifies 256×256 images as **"Flood" vs "No Flood."** If we apply Grad-CAM for the "Flood" class on an image the model labeled as flooded, we might see bright regions (hot colors) corresponding to areas of standing water (e.g. inundated fields or rivers) on the image [44] [45]. This would confirm that the model's decision was based on the presence of water in those areas – which is good. If instead Grad-CAM highlighted an unrelated region (say, a cloud or an area that's not actually flooded), that would be a red flag that the model might be misled by some confounding feature (perhaps it learned to associate clouds or shadows with floods incorrectly). Grad-CAM is especially popular for **interpretability in vision tasks** because it directly shows on the image what the model deemed important, making it very intuitive.

Grad-CAM example – Flood Detection: The images above (from a research study on urban flooding) illustrate Grad-CAM explanations [44] [45]. The top row shows original scenes, and the bottom row shows Grad-CAM heatmaps for a CNN that classifies each scene. In the left pair (a "Normal" class example), the heatmap is diffuse, focusing on the road (dry areas) rather than water – aligning with the model recognizing a non-flooded scene [46] [45]. In the middle (a "Flood" example), the bright colored regions in the heatmap coincide

with waterlogged areas on the road, indicating the model is rightly attending to flood evidence [46] [45]. The right pair ("Unknown" or irrelevant example) shows the model's attention is scattered on unrelated objects, which is expected since it should ideally find no strong flood or normal signal [47] [48]. This kind of visualization is incredibly useful: it validates when the model is doing the *right* thing, and it helps diagnose when the model might be keying off the *wrong* features. In production, one might run Grad-CAM on a sample of results to ensure, for example, a building-detection CNN is lighting up on building rooftops (not on adjacent trees or shadows), or a cloud detection model is highlighting actual clouds (not snow or bright rooftops by mistake).

Philippines Use Case for XAI: Consider a flood mapping model applied after a typhoon – it flags certain areas as flooded. By applying Grad-CAM or similar saliency maps, disaster managers can **see what the model "thinks" is water**. If the highlighted areas align with actual riverbanks and submerged fields, great – the model is trustworthy. If instead it's highlighting cloud shadows or mountain valleys (where there is no water, but maybe dark shadows confused it), the users know the model may be making an error. They can then correct or be cautious with those outputs. Another scenario: an AI model identifies *illegal fishing platforms* in satellite images for maritime authorities. Using XAI (perhaps SHAP or LIME on the features it uses, or Grad-CAM on imagery if it's CNN-based) could provide an explanation like: "The model focused on these bright rectangular shapes in the open water." This could help authorities verify those are indeed the bamboo structures of interest, rather than the model accidentally picking up on something irrelevant (like a boat or wave pattern). In essence, XAI provides **human oversight and insight** into the model's logic, which is invaluable for critical environmental monitoring tasks in the Philippines or anywhere.

Think-Through: *You have a black-box model that predicts areas of high landslide risk from satellite imagery and topographic data. What are some reasons it would be important to make this model explainable, especially if government agencies will use its predictions to prioritize evacuations or infrastructure projects? What could go wrong if the model is not interpretable?*

Mini-Challenge: *Take one of the EO models you encountered earlier (e.g. the flood mapping CNN or a land cover classifier) and describe how you would apply an XAI method to it. For instance, if it's a CNN, how would you use Grad-CAM to verify it's identifying flooded areas correctly? If it's a decision tree or random forest, how would you use SHAP values to understand which environmental factors are driving its predictions? Write down what steps you would follow to generate the explanations and what specific insights you'd be looking for in the context of a Philippine application (floods, land use, etc.).*

Activity: XAI Demo (Conceptual)

In this session (theory), we won't be running code, but let's **conceptualize an XAI demo** that ties together what we learned:

- **Grad-CAM on a U-Net Flood Mapping Model:** Recall the U-Net model some of you trained on Day 3 for segmenting floods in Sentinel-1/Sentinel-2 imagery. Imagine overlaying a Grad-CAM heatmap on the satellite image for a predicted flood region. How would it look? We'd expect the U-Net's "flood" class activation to be high (hot colors) exactly where floodwater is present (along rivers or inundated fields). As a demo, one would take a test image that the U-Net marked as flooded in certain parts, compute Grad-CAM for those flooded pixels or for the flood class, and visualize the result as a colored heatmap on the image. The **interpretation**: if the heatmap aligns with water bodies, the model is correctly basing its segmentation on water signals; if it highlights, say, clouds or irrelevant

regions, that reveals an issue. This conceptual demo would show students *how* Grad-CAM can validate and build trust in the model's predictions before they are used in real disaster management. It connects the theory from Module 3 with the practical model from Day 3.

- **SHAP on a Random Forest Land Cover Model:** On Day 2, suppose we built a Random Forest to classify land cover types (e.g. forest, built-up, agriculture) using input features like spectral indices (NDVI, NDWI, etc.), elevation, and slope. A great exercise is to apply SHAP values to one of its predictions. For example, take a specific location the RF predicts as "Built-up" and compute SHAP values for all features for that prediction. In a live demo (conceptually), we'd show a bar chart of feature contributions: perhaps "Nighttime Lights = high, NDVI = very low, Red Roof Index = high" contributing to the "built-up" classification, whereas "Elevation" and "NDWI" might have negligible or negative contributions. This aligns with expectation that built-up areas have lights at night, low vegetation, certain roof reflectance. By showing this, trainees can see **why** the model said "built-up" – not just *what* it predicted. We could also examine a "Forest" prediction and see SHAP highlighting high NDVI and high texture, etc. The demo would reinforce how XAI helps interpret classic ML models in EO.

These conceptual demos emphasize that for any AI model we develop, we should plan for an explainability step. By mentally walking through Grad-CAM and SHAP on our models, we prepare ourselves to actually implement these in practice during the hands-on sessions or future projects.

---

1 12 14 15 20 21 Clay Foundation Model — Clay Foundation Model
https://clay-foundation.github.io/model/index.html

2 7 NASA and IBM Openly Release Geospatial AI Foundation Model for NASA Earth Observation Data | NASA Earthdata
https://www.earthdata.nasa.gov/news/nasa-ibm-openly-release-geospatial-ai-foundation-model-nasa-earth-observation-data

3 26 27 28 29 30 31 32 MIMRS: A Survey on Masked Image Modeling in Remote Sensing
https://arxiv.org/html/2504.03181v2

4 22 23 24 Revolutionizing earth observation with geospatial foundation models on AWS | Artificial Intelligence
https://aws.amazon.com/blogs/machine-learning/revolutionizing-earth-observation-with-geospatial-foundation-models-on-aws/

5 6 9 10 11 Expanded AI Model with Global Data Enhances Earth Science Applications - NASA Science
https://science.nasa.gov/science-research/ai-geospatial-model-earth/

8 13 16 25 Using Foundation Models for Earth Observation — Development Seed
https://developmentseed.org/blog/2024-11-01-geofm/

17 33 SatMAE: Pre-training Transformers for Temporal and Multi-Spectral Satellite Imagery
https://sustainlab-group.github.io/SatMAE/

18 19 [2403.15356] Neural Plasticity-Inspired Multimodal Foundation Model for Earth Observation
https://arxiv.org/abs/2403.15356

34 Transformer-based land use and land cover classification with explainability using satellite imagery | Scientific Reports
https://www.nature.com/articles/s41598-024-67186-4?error=cookies_not_supported&code=8e426fac-2d74-4feb-b331-7b56b194d53c

35 36 SHAP global explanations on the spectral bands of Sentinel-2. | Download Scientific Diagram
https://www.researchgate.net/figure/SHAP-global-explanations-on-the-spectral-bands-of-Sentinel-2_fig2_368968203

37 38 bar plot — SHAP latest documentation
https://shap.readthedocs.io/en/latest/example_notebooks/api_examples/plots/bar.html

39 40 41 42 43 Opening the Black-Box: A Systematic Review on Explainable AI in Remote Sensing
https://arxiv.org/html/2402.13791v2

44 45 46 47 48 Example of the decision attention of the trained model. The upper and... | Download Scientific Diagram
https://www.researchgate.net/figure/Example-of-the-decision-attention-of-the-trained-model-The-upper-and-lower-rows-show-the_fig3_354515821