ChatGPT

Day 4: Time Series Analysis, Emerging Trends, and Sustainable Learning

Session 1: AI for Time Series Analysis in EO – LSTMs (1.5 hours)

Understanding Time Series Data in Earth Observation

Earth observation (EO) often produces **time series data** – sequences of observations over time. For example, satellites like Sentinel-2 capture images of the same area repeatedly, allowing us to track how an index like NDVI (Normalized Difference Vegetation Index) changes month by month. NDVI is a popular measure of vegetation health, computed from red and near-infrared reflectance; **high NDVI values** mean lush, healthy vegetation, while **low NDVI** indicates sparse vegetation or bare ground [1]. By examining NDVI time series, we can observe seasonal cycles (phenology) such as vegetation green-up and senescence, and detect anomalies like drought impacts. **Synthetic Aperture Radar (SAR)** sensors also provide time series of backscatter intensity, useful for tracking changes in soil moisture or surface structure (e.g., flood inundation or deforestation events) over time [2]. Time series data give us a *temporal dimension* to monitor trends and patterns that single images cannot show.

Simulated NDVI time series over 3 years, with a pronounced drop in year 2 due to drought. Such EO time series reveal seasonal vegetation patterns and anomalies like drought-induced declines in plant health. Each point represents NDVI for a time step (e.g., monthly), illustrating how prolonged low values signal stress.

However, analyzing EO time series is challenging. Traditional machine learning models (like random forests) assume independent data points, but time series observations are **autocorrelated** (each value relates to previous values). We need models that consider *temporal dependencies* – how past observations influence future ones. This is where sequence models like Recurrent Neural Networks (RNNs) come in.

From RNNs to LSTMs: Dealing with Long Sequences

Recurrent Neural Networks (RNNs) are designed for sequential data. An RNN processes one time step at a time, carrying forward a **hidden state** that *"remembers"* information from previous time steps. In theory, a basic RNN could learn patterns over arbitrary long sequences. In practice, though, vanilla RNNs struggle with **long-term dependencies**. As we backpropagate errors through many time steps, gradients tend to either **vanish** (shrink towards zero) or **explode** (grow uncontrollably), making it difficult for the model to learn relationships that span more than a few steps [3]. This *vanishing gradient problem* means an RNN might "forget" important early information by the time it reaches later steps.

To address these limitations, Hochreiter & Schmidhuber (1997) introduced the **Long Short-Term Memory (LSTM)** network. LSTMs are a special type of RNN explicitly designed to remember information for long durations *and* to forget irrelevant info when needed [4]. The name "long short-term memory" reflects that it can retain long-term dependencies while still using a short-term memory mechanism [5].

How LSTM Works: An LSTM cell has an internal memory **cell state** and three main gates that regulate information flow [6] :

- **Forget Gate:** Decides what old information to throw away from the cell state. Based on the previous hidden state and current input, it outputs a value between 0 and 1 for each piece of information – 1 means "keep this" and 0 means "forget this." Essentially, *it's like our brain intentionally forgetting irrelevant details*.
- **Input Gate:** Determines which new information to add to the cell state. It uses a similar mechanism (0 to 1 values) to filter the current input and create candidate values (often via a **tanh** activation) that could be added to memory. This is akin to *deciding what new facts to save to memory*.
- **Output Gate:** Controls what information from the cell state is revealed as the output (and thus passed to the next time step's hidden state). It's like *deciding what to recall from memory when you need it*.

These gates are implemented with sigmoid activation functions (outputting 0-1 weights) and element-wise multiplication to **gate** the information flow. The cell state is like a conveyor belt running through time steps, modified by gates to persist important information [7] .

Diagram of an LSTM memory cell. Orange blocks represent activation functions (sigmoid σ for gates, tanh for candidate values); yellow circles are pointwise operations (like element-wise addition). The cell state (horizontal line) runs through, with the Forget gate (f) filtering out old info, the Input gate (i) adding new info (after a tanh layer generates candidate content ${\tilde{C}}$), and the Output gate (o) controlling the released hidden state ${h_t}$ [6] . This gated architecture allows LSTMs to carry relevant information forward while discarding what's not needed.

Analogy: Think of an LSTM as a **smart notebook** you carry when observing a phenomenon. The *cell state* is the notebook's page. At each time step, you decide what to erase from the page (forget gate), what new observation to write down (input gate), and what summary to read aloud to others (output gate). Because you don't erase everything each time – only what seems unimportant – you can accumulate knowledge over many observations without overflowing or forgetting crucial details. This mechanism enables learning long-term patterns (like an annual season cycle or a multi-year trend) far better than a standard RNN [4] .

Applications of LSTM in EO

With LSTMs able to capture long-range dependencies in sequences, they have become powerful for EO time series analysis. Some **practical applications in Earth Observation** include:

- **Drought Monitoring and Forecasting:** Using vegetation index time series (like NDVI) to detect drought onset and predict vegetation stress. For example, an LSTM can learn the normal seasonal NDVI pattern for croplands and flag when the series deviates due to drought conditions [8] . It can even be trained to forecast future NDVI values or drought indices (e.g., SPEI) a few time steps ahead.
- **Crop Yield Prediction:** Combining time series of EO data (vegetation indices, weather data) with LSTMs can improve crop yield forecasts. The LSTM can integrate information across the growing season – e.g., if a drought in early season followed by recovery later – to predict final yield, something hard for models lacking memory.
- **Land Cover Change Detection:** Time series of satellite images or indices can reveal changes like deforestation, urbanization, or crop rotations. An LSTM can be trained on historical sequences to

detect when a significant change is happening (for instance, a forest's greenness pattern abruptly shifts when clearing occurs).

- **Phenological Analysis:** By analyzing multi-year NDVI time series, LSTMs can help identify phenological events (start of season, end of season) and how they shift over years. This is useful for climate change studies (e.g., tracking how El Niño or La Niña affect timing of greening in the Philippines).

Philippine Context: In the Philippines, seasonal patterns and extreme events are critical. For instance, Mindanao's agricultural regions have distinct wet and dry seasons. An LSTM could learn the typical NDVI rise during the monsoon and drop in the dry season, and more importantly, detect anomalies like **El Niño-induced droughts** where NDVI stays depressed beyond the usual dry season. By training on historical data, an LSTM model might forecast drought impacts on vegetation and give early warning for crop failures in provinces like Bukidnon or South Cotabato. (We will explore such a case in the hands-on Session 2.)

Think-Through: Why Not Simpler Methods?

Before diving deeper, consider this question: *Why do we need LSTMs at all for EO time series?* Couldn't we just use an ARIMA (a classic statistical model) or simple regression on time features? Discuss what advantages an LSTM offers (hint: ability to automatically learn **non-linear** and **long-term** dependencies). Also consider the downsides – LSTMs need more data and computational power. By thinking this through, you'll appreciate when a deep learning approach is warranted versus when a simpler time series model might suffice.

Mini-Challenge: Identify the Pattern

Below is a short NDVI sequence (just conceptually, not actual numbers) for a rice paddy: `[0.3, 0.4, 0.6, 0.8, 0.7, 0.5, 0.3]` over several months. What do you think is happening in this sequence? Can you identify a plausible phenological stage (e.g., green-up then harvest)? How might an LSTM help in detecting such patterns automatically? This mini-challenge encourages you to link the numbers to real events and see how a model could learn those associations.

Session 2: *Hands-on* – Drought Monitoring with Sentinel-2 NDVI & LSTM (2.5 hours)

This session is an overview of a hands-on exercise (without diving into actual code here). We outline how an LSTM can be applied, step-by-step, to a real drought-monitoring problem in the Philippines.

Case Study Overview: Mindanao Drought Monitoring (CCA Focus)

We focus on a case study in **Mindanao**, a region with extensive agriculture (e.g., corn, rice, pineapple) that is periodically affected by drought. Specifically, consider provinces like **Bukidnon** or **South Cotabato**, which are important crop-growing areas and have experienced droughts in past El Niño years. This case ties directly into Climate Change Adaptation (CCA) – helping farmers and planners cope with climate variability [9] [10].

Objective: Use Sentinel-2 NDVI time series to monitor drought stress on vegetation, possibly predicting a drought severity index. For instance, we might try to predict the Standardized Precipitation

Evapotranspiration Index (SPEI) or a simpler drought category (normal, moderate drought, severe drought) for each time step, using NDVI sequences as input. Alternatively, the task could be to forecast NDVI itself a few months ahead to serve as an early-warning of impending drought impacts on crops.

Data: We will use pre-prepared time series data [11]. The dataset could include: - **NDVI sequences** (monthly or bi-monthly averages) for several representative pixels or agricultural zones in Mindanao, spanning multiple years (so we have enough history, e.g., 2015–2021). - **Drought indices or rainfall** as target outputs. If available, PAGASA's drought reports or indices like SPEI over the same time can be the value the LSTM tries to predict from the NDVI sequence. If not, one could use CHIRPS satellite rainfall anomalies as a proxy target. The idea is to correlate vegetation health with actual drought conditions [12].

Why not use raw imagery? Because for time series modeling, we want a *clean, consistent* series. Sentinel-2 images are frequent (every 5 days) but cloudy weather is a major issue – especially in the humid tropics of the Philippines. So we aggregate and pre-process: - Compute NDVI from each image. - Mask out clouds (using Sentinel-2 cloud masks or QA bands) to avoid bad data. - Aggregate the cloud-free pixels over a month to get a representative NDVI value (this could be an average or median over the area of interest). - Fill or interpolate missing values if an entire month is missing due to clouds (e.g., use linear interpolation or carry forward the last value cautiously).

This ensures the NDVI time series we feed to the model reflects real vegetation signals and not noise from clouds or other artifacts [13].

Data Prep Pitfalls: Emphasize to participants that a huge amount of effort can go into making an **analysis-ready time series**. Cloud masking must be accurate – any undetected cloud could drastically lower NDVI spuriously. If multiple images per month are available, using medians can help cancel out cloud effects. Interpolation should not create false data – it's better to mark gaps and perhaps let the model handle them (some advanced sequence models can handle missing time steps). For training, we provide a cleaned dataset so attendees can focus on modeling, but we note these steps because in real projects, *data quality is king* [14].

LSTM Modeling Workflow for Drought

We conceptually walk through the workflow of training an LSTM on this data [15] [16]:

1. **Load and Inspect Data:** Imagine we load our NDVI CSVs for a couple of locations. We plot them to see baseline patterns – e.g., a strong seasonal cycle, and notably, 2015-2016 shows a dip corresponding to a drought. Understanding the data visually is an important first step.
2. **Normalize Features:** NDVI values range from -1 to +1, but typically 0 to 0.8 for vegetated areas. We scale the inputs to a 0–1 range or standardize them. This helps the neural network train faster and avoid getting stuck (since unnormalized sequences might cause issues).
3. **Create Input Sequences and Labels:** This is crucial. We decide on a **window size** – say 12 months. Using a sliding window, we take NDVI values from month t-11 to t (12 points) as one input sequence, and the **target** could be e.g. the drought index at month t (sequence-to-value prediction) or NDVI at month t+1 (if forecasting NDVI itself). For each time step in our series (except the first 11 which don't have 12 prior points), we get one training sample. This way we turn one long time series into many training examples. *(This approach assumes fairly stationary patterns; we might also include multiple locations as separate sequences to have more training samples.)*

4. **Define the LSTM Model:** We choose an architecture – for example, one LSTM layer with 50 units, followed by a Dense layer that outputs a single value (the predicted drought index). The input shape is (timesteps=12, features=1) for univariate NDVI input. If we included multiple features (say NDVI plus rainfall), then features would be >1.
5. **Compile the Model:** Select a loss function appropriate for the task. If predicting a continuous drought index, use a regression loss like Mean Squared Error. If classifying drought severity (e.g., normal vs drought categories), use a classification loss like cross-entropy. Use an optimizer (Adam is a good default for LSTMs).
6. **Train the Model:** Feed in our sequences and targets, iterating for a certain number of epochs. We have to be careful with sequences – typically we shuffle the sequences during training but not within a sequence (the temporal order inside each training sample must be preserved).
7. **Evaluate Performance:** If it's regression, look at RMSE between predicted and actual drought index [17]. If classification, look at accuracy or better, precision/recall for the drought class. We also hold out some data for validation (e.g., data from a later year or another province) to ensure the model generalizes.
8. **Visualize Predictions:** A great way to understand the model is to plot the actual vs predicted time series [18]. For example, if we predicted NDVI itself, plot the real NDVI curve and the one-step-ahead predictions to see if the model captures the dips during droughts. If predicting drought index, maybe plot predicted vs actual index over time to see timing.

Throughout this workflow, we reinforce *conceptual understanding*. For instance, **why 12 months window?** Perhaps because crops have annual cycles – but what if the target has longer memory? Discuss how one might choose or tune the sequence length. Also, highlight that LSTMs *learn to "remember"* past information internally, but they have limits – if a drought's effect lasts 2 years, maybe we need a longer window or a different approach (or multiple LSTM layers).

We'll also mention practical tips: use early stopping to avoid overfitting, visualize training loss curves, etc. While code is not shown here, in the actual session participants would implement these steps in a Colab notebook, solidifying the process.

Challenges and Considerations

Dealing with Cloud Gaps: If some sequences have missing months due to clouds, how do we handle that? One approach: use interpolation to fill, but mark an additional binary feature "is_real_data" vs "interpolated" so the LSTM knows to trust some points less. Another approach: use a special value (like -1) for missing and modify the model or loss to skip those. These are advanced considerations; for the training, we likely avoid missing data by careful prep. But we want participants aware that EO time series can be messy.

Conceptual Hurdle: EO practitioners might find it abstract how the LSTM "remembers". We will visually explain that inside the LSTM, the cell state carries information from prior time steps. For example, if NDVI has been steadily decreasing for 3 months, the LSTM can carry that trend and thus predict a continued decrease – mimicking how we would reason about a developing drought. A simple diagram of sequence-to-value flow can be shown: Input NDVI[t-3...t] -> [LSTM memory] -> predict NDVI[t+1]. Understanding this flow is key [19]. We encourage questions like "*If the rainy season failed, does the model 'know' it?*" – Yes, because the lack of an NDVI rise would be encoded in the cell state.

Think-Through: Designing Your Own Time Series Model

Suppose you wanted to monitor **urban heat island** effect with time series of land surface temperature (LST) in Metro Manila. How would you set up a similar LSTM? Think about what the input sequence would be (perhaps monthly average LST), and what target you'd predict (maybe the next month's LST or an extreme heat event occurrence). What challenges might arise (e.g., holiday anomalies, sensor calibration changes)? This exercise connects the dots from our drought case to other EO time series problems, prompting you to outline an approach.

Mini-Challenge: Cloud Clearing Strategy

You have NDVI data for 24 months but 4 months are missing due to clouds during a long monsoon spell. Propose a strategy to still use LSTM on this data. Options could include: (a) fill with the last known NDVI, (b) fill with a climatological average for that month, or (c) treat them as special "masked" inputs the model learns to skip. Discuss pros and cons of each. This mini-challenge pushes you to consider data preprocessing choices critically.

Session 3: Emerging AI Trends in EO – Foundation Models, SSL, and XAI (2 hours)

In this session, we step back from specific models and look at **cutting-edge trends** in AI for EO. These are more research-oriented but increasingly practical topics: large **Foundation Models** for geospatial data, **Self-Supervised Learning (SSL)** techniques, and **Explainable AI (XAI)** methods to interpret models. We will cover three modules:

Module 3.1: Geospatial Foundation Models (GeoFMs)

Foundation Models are large AI models trained on broad data at scale, which can be adapted to many tasks [20]. In EO, this means models that ingest a huge variety of satellite imagery (possibly from different sensors, times, locations) without a specific task, and learn general representations of Earth data. Later, we **fine-tune** these models on a specific task (with a relatively small labeled dataset) and achieve good performance because the model already learned a lot of "Earth intelligence" during pre-training [21].

Think of a foundation model as a *base model* that knows, for example, how typical vegetation looks in multispectral data, or the difference in texture between urban and forest in radar, etc., because it saw millions of examples. You can then quickly adapt it to, say, classify land cover in a new Philippine province with minimal training data – the heavy lifting was done before.

Several **Geospatial Foundation Models (GeoFMs)** have emerged recently [22]:

- **Prithvi** – a foundation model by IBM and NASA. *Prithvi-EO* (Earth Observation) is a Vision Transformer-based model pre-trained on a terabyte of satellite data (Harmonized Landsat-Sentinel imagery over the U.S.) [23] [24]. It was one of the largest GeoFMs (the first version had 100 million parameters, later versions up to 600M). Prithvi can be fine-tuned for tasks like flood mapping or land classification and often gives a boost in accuracy because of its rich pre-learned features [25] [26].

- **Clay** – an open-source foundation model developed by a coalition (including Radiant Earth, Development Seed). Clay is a Vision Transformer adapted for EO that takes in imagery plus metadata like location and time [27]. It's trained with a Masked Autoencoder approach (more on that in SSL) on global satellite data. The goal is to have a model readily available to the community. Clay outputs **embeddings** (vector representations) for any given satellite image patch, which can then be used for various purposes (classification, clustering, change detection) [28] [29]. It's like a *general-purpose feature extractor* for Earth.
- **SatMAE** – stands for Satellite Masked AutoEncoder. It is a framework from research that specifically pre-trains a Transformer by having it reconstruct missing pieces of multi-spectral and temporal satellite images [30] [31]. By doing so on lots of data, the model learns powerful features. SatMAE introduced clever strategies for handling time series of images (like a sequence of images over a year) and multispectral information. It's a type of foundation model focused on self-supervised learning.
- **DOFA** – stands for *Dynamic One-For-All*. This is a new kind of foundation model that is **multimodal** [32]. It's designed to handle multiple types of EO data (like optical imagery, SAR imagery, elevation data, etc.) within one unified model [33]. Inspired by **neural plasticity**, DOFA can adjust to different input channels and modalities without needing a separate model for each [34]. For example, DOFA could take a Sentinel-1 radar image or a Sentinel-2 optical image and still produce a useful embedding – it's flexible like a brain adapting to new senses [35] [36]. This is especially appealing for places like the Philippines where integrating data (optical + SAR for cloudy areas) is valuable.

Why it matters: Foundation models could *revolutionize* EO analysis [37]. Instead of training a new model from scratch for each task (land cover, flood, drought, etc.), we start with a robust GeoFM and just fine-tune. This saves time (less training data needed for each task) and often improves performance (the model's prior knowledge fills in where data are scarce). For Philippine applications, this is a big deal: we often lack large labeled datasets for specific local tasks. A GeoFM that's seen *the whole world* can generalize to Philippine geography with just a few examples. Imagine a model that has learned how rice paddies look in Vietnam and Thailand – it could identify rice in the Philippines with minimal additional training.

Illustration of a Masked Autoencoder approach for satellite imagery (SatMAE example). The model is given an input image (or sequence of images) where parts of the data (patches of the image, across spectral bands and time steps) are masked out (shown as blank areas). The encoder learns to produce a compact representation of the visible parts, and the decoder then reconstructs the missing parts. By trying to in-paint the missing patches, the model learns rich features of satellite imagery without any human labels [38] [39]. After such pre-training on a large dataset, the encoder can be reused for various EO tasks.

Think of it like solving millions of jigsaw puzzles of satellite images – eventually, the model figures out what pieces typically look like and how they fit together. This example highlights **self-supervised learning** (SSL), which we discuss next.

Module 3.2: Self-Supervised Learning (SSL) in EO

Self-Supervised Learning is the key technique behind many foundation models. In SSL, the model creates its **own training task** (a *pretext task*) using unlabeled data. We just saw one example: masked autoencoding – hide parts of the image and make the model predict them. Another example is **contrastive learning**: e.g., take two different views of the same scene (like one Sentinel-2 image and a slightly later one, or an image and its augmented version) and train the model so that it produces similar embeddings for these related inputs and dissimilar embeddings for unrelated inputs. The model thus learns meaningful features (e.g., invariant to lighting or season) without any human telling it what is “forest” or “urban” explicitly [40].

Why is SSL **particularly useful for EO?** Because we have **tons of data but few labels**. Satellites generate terabytes of imagery, but labeling images (for land cover, for damage assessment, etc.) is expensive and slow. SSL leverages the unannotated portion. For instance, you could take all Landsat images of the Philippines and train a model with a simple objective: each image, randomly crop two patches from it, and ask the model to recognize that those two patches come from the same larger scene (positive pair) versus patches from different scenes (negative pair). This is a form of **contrastive learning**. The model, in trying to solve this, must learn that patches from the same location likely share certain features (same vegetation type, etc.), whereas random patches might not. Over time, the model clusters similar land types in its feature space – without any explicit labels.

Real-world example: A recent approach, **SimCLR or MoCo** adapted to EO, could be used on Sentinel-2: take an image, create augmentations (one with slight color jitter, one with a small rotation), feed through a CNN, and train to maximize agreement between augmentations of the same image versus different images. Models like SatMAE we mentioned specifically tailor the **masked autoencoder** idea to multi-spectral imagery [31] – for example, randomly mask out entire spectral bands or entire time steps and let the model guess them, which forces it to learn how different bands relate (e.g., how NIR can predict Red, etc.) and how images change over time.

The result of SSL pre-training is a model that has learned **useful representations** of EO data. We can then fine-tune it on a small labeled dataset and get much better performance than training from scratch, as the model already *speaks the language of EO*. This addresses the **data scarcity** problem in regions like the Philippines [41]. Even if we only have 500 labeled samples of mangroves vs. rice fields, an SSL-pretrained model might already “know” the spectral-temporal signature of mangroves vs rice from unlabeled data, thus it only needs slight nudging to perform the classification.

Module 3.3: Explainable AI (XAI) for EO

As we deploy complex AI models (random forests were sort of interpretable, but CNNs, U-Nets, LSTMs are *black boxes*), it’s crucial to **explain and trust** their decisions [42]. **Explainable AI (XAI)** is a set of techniques to make the model’s workings more transparent.

We will introduce three popular XAI methods and how they apply to EO:

- **SHAP (SHapley Additive exPlanations):** Think of SHAP as asking, "*Which features contributed most to this prediction?*". For tabular data, features could be indices or bands; for an image, “features” could mean super-pixel regions or some summary of input. SHAP assigns each input feature a *Shapley value* indicating how much it pushed the model’s prediction up or down [43]. For example, if a random forest predicts "forest" for a pixel, SHAP might tell us Band 8 (NIR) with high value was the biggest reason (since forests have high NIR reflectance), while Band 11 (SWIR) contributed less. In EO context, SHAP has been used to rank the importance of different satellite bands or indices in classification tasks, or even non-image features like elevation or climate data in yield models. We can show a simple **bar chart** of feature importances from SHAP: e.g., *NDVI and Near-Infrared have the highest positive influence on predicting dense vegetation, whereas the presence of built-up index has a negative contribution for the same.* This helps practitioners validate that the model is “looking at” sensible features (if SHAP said the *blue band* is the top predictor for vegetation, we'd suspect something's off, since NIR or SWIR should be more important for vegetation health).

- **LIME (Local Interpretable Model-Agnostic Explanations):** LIME is like doing *"what-if" analysis* around a single prediction [44]. For an image classification, LIME will take the image, break it into segments, and try turning segments off to see how the prediction changes. In EO, imagine a satellite image where a model predicts "urban". LIME might turn off (mask) the roads, then the buildings, etc., to see what made the model say "urban". If masking out the high-reflectance rooftops drops the confidence drastically, it tells us the model was relying on those features. This is useful for *local explanation* – why *this particular* image was labeled a certain way [45]. We might show an example: a patch classified as "water" – LIME highlights the dark blue pixels (water region) as the reason. If it highlighted something odd (like clouds), we'd have a problem.
- **Grad-CAM (Gradient-weighted Class Activation Mapping):** This is specific to CNNs (works for imagery). Grad-CAM produces a **heat map** over the input image, showing which regions were most influential for a given class decision [46]. For example, we feed a satellite image into a scene classification CNN that says "airport". Grad-CAM might highlight the runway shape in the image as the key feature the CNN focused on (perhaps a bright elongated region). In EO disaster contexts, if we had a CNN detecting flooded areas, a Grad-CAM could show *which parts of the image were lit up for "flood"*. If the water pixels are highlighted, great – it means the CNN truly learned to detect water; if some unrelated region glows, the model might be picking up on a proxy (like shadows or certain vegetation) incorrectly. We can present a simple visual: original image vs. Grad-CAM heatmap overlay. For instance, for a *"building" detection CNN*, the heatmap might glow on the building rooftops, confirming the model's focus is correct.

These XAI tools are vital for **building trust**. Philippine agencies (and any stakeholders) will ask, "*Why did the model predict a drought here?*" or "*Why does it say this area is deforested?*". With XAI, we can answer in human terms: *"Because the vegetation index dropped sharply and rainfall was low – see these features – thus the model is confident it's drought."* or *"The CNN highlighted this texture that corresponds to a flooded river channel when making its decision."* This not only helps in convincing decision-makers but also helps us **debug models**. If an explanation reveals the model is using a flawed cue (say, it flags clouds as "deforestation"), we know the model might fail in real scenarios and needs improvement.

Finally, we stress **transparency**: For applications in disaster risk reduction (DRR) or climate adaptation, lives and resources could be at stake. Black-box predictions are hard to act on. If a model predicts a severe flood, authorities need to know if it's seeing actual floodwater or some noise. XAI provides that window into the model's "brain", which is crucial for adoption and integration of AI in operational workflows [47].

Philippine Relevance & Examples

In the Philippines, we have some unique drivers for these emerging methods: - **Data Scarcity:** Many local environmental phenomena lack large labeled datasets (e.g., coral reef health, certain crop pests). Foundation models + SSL can help by transferring learning from global data to local needs. For instance, a foundation model pre-trained on global coastline images could be fine-tuned to map Philippine mangroves with few labels, aiding NRM (Natural Resource Management) efforts. - **Black-box Mistrust:** There can be skepticism towards AI models in government or communities, especially if a model's recommendation contradicts local knowledge. XAI can bridge this gap by showing *why* – for example, an AI system identifies a landslide risk zone; explaining that it's due to slope, deforestation, and recent rainfall patterns (factors that domain experts recognize) will make the AI's output more credible and actionable. - **DRR/CCA Needs:** In disasters, time is critical but so is justification. AI might quickly flag 100 buildings as collapsed after an earthquake (great, speed-wise), but officials will prioritize sending help if they trust those flags. With XAI, we can show satellite image chips with red highlights on the collapsed structures (via Grad-CAM), giving confidence that the model truly saw damage, not just a shadow.

Think-Through: Would You Deploy It?

Consider a scenario: you have a fancy foundation model that can map informal settlements from satellite images with 90% accuracy, but it's completely black-box. On the other hand, a simpler model (say, a decision tree) has 80% accuracy but is fully transparent (it uses a few spectral thresholds that you can explain easily). **Which would you choose for a government program on urban planning?** This question doesn't have a single answer – it asks you to weigh accuracy vs. explainability. In some cases, the stakes may demand the transparency of the simpler model. In others, the higher accuracy might justify using the complex model with some XAI techniques to support it. Think about the context (e.g., is this life-and-death or a low-risk mapping exercise?), and what **"explainable enough"** means for your stakeholders.

Mini-Challenge: Propose a Pretext Task

Imagine you want to pre-train a model on all Philippine satellite imagery (say from Diwata microsatellite or Sentinel) without labels. Come up with a **self-supervised task** for this data. Hints: You could use the fact that the Philippines has distinct dry/wet seasons – maybe have the model predict which month of the year an image patch belongs to (so it must learn seasonal appearances). Or use multi-modal data – e.g., train a model to align radar images with optical images for the same location (forcing it to understand both modalities). Be creative. The goal is to realize there are many ways to "trick" a model into learning useful features without human labels.

Session 4: Synthesis, Q&A, and Pathways to Continued Learning (2 hours)

Recap of Models and Their Uses (Philippine Context)

Over the past days, we covered a suite of AI/ML techniques. Let's **recap the key models**, linking each to real-world Philippine applications in Disaster Risk Reduction (DRR), Climate Change Adaptation (CCA), and Natural Resource Management (NRM):

- **Random Forest (RF):** A machine learning ensemble based on decision trees. It shines with tabular or structured data and was our introduction to AI for EO. In Day 1, we used RF for land cover classification on Sentinel-2 imagery. RF was intuitive (if NDVI > X and SWIR < Y, classify as forest, etc.). *Philippine use case:* Rapid land cover mapping for NRM – e.g., mapping forest cover in Palawan – RF can do this if provided good spectral features [48] [49]. It's fast and gives reasonable accuracy, though it might struggle with very complex patterns (enter deep learning next).
- **Convolutional Neural Networks (CNNs):** Deep learning models for grid data (like images). We learned how CNNs automatically extract features through convolutional layers. On Day 2, a basic CNN classified EO image patches (e.g., distinguishing 'water', 'urban', 'forest' in EuroSAT or local data). *Philippine use case:* identifying **urban vs rural** areas for planning – CNNs can process high-res images and detect built-up textures. Also, CNN backbones are part of more complex models below.

- **U-Net (CNN for Semantic Segmentation):** This encoder-decoder network performs pixel-wise classification. We applied U-Net for flood mapping (a DRR task) on Sentinel-1 SAR data in Day 3. U-Net's skip connections allowed precise segmentation of flooded pixels. *Use case:* **Flood extent mapping** after typhoons (like Typhoon Ulysses in Luzon) – U-Net can delineate floodwaters vs. land [50] [51]. Also applicable to mapping burn scars after wildfires, or delineating coral reefs vs. sand in coastal maps for NRM.
- **Object Detection Models (e.g., YOLO):** While not covered in detail in our write-up above, object detectors were mentioned. These models (like YOLO – You Only Look Once) can put bounding boxes around specific objects in images. *Philippine use case:* **Urban monitoring** – detect buildings or houses from aerial images to update maps in rapidly developing areas. Or count fishing boats in ports (for fisheries management), etc. We touched on how these differ from classification/ segmentation (they locate individual instances). The *urban change* example ties to both DRR (identifying high-risk informal settlements in floodplains) and NRM/urban planning [52].
- **LSTMs (Recurrent Neural Networks):** Today's focus in Session 1 & 2. LSTMs analyze temporal sequences. *Use cases:* **Drought prediction** (CCA) as we did, **phenology trend analysis** (e.g., monitoring how rice cropping cycles shift – relevant to food security), or even **air quality forecasting** (using time series of satellite-derived aerosol data – important for health). The key advantage is capturing temporal patterns and making forecasts, which is crucial for *early warning systems* in DRR (like predicting a dry spell's impact on water supply).
- **GeoSpatial Foundation Models & SSL:** These aren't separate "applications" but rather *game-changing approaches* that can improve all the above tasks. We include them in recap because moving forward, they will likely become part of the toolkit for Philippine EO analytics – making models more accurate even with limited local training data, and enabling *fast development* of AI solutions for new problems.

In summary, there's no one-model-fits-all. Each has a niche: - For **small/medium tabular datasets and interpretability**: Random Forest is great. - For **image classification or when spatial detail is needed**: CNNs/U-Net are the go-to. - For **counting or locating objects**: YOLO (object detection). - For **time-dependent trends**: LSTM. - And the new paradigm: **pre-train (GeoFM) + fine-tune** can augment any of these tasks with more knowledge.

Best Practices for AI/ML in EO Projects

Now that we know *what* models to use, let's talk about *how to use them well*. Some best practices to emphasize [53] [54]:

- **Data Quality over Quantity:** It's tempting to feed as much data as possible, but *garbage in, garbage out*. Ensure imagery is well-calibrated, cloud-free, and representative of the target conditions. In our flood mapping example, we insisted on radiometrically corrected SAR images and accurate flood masks. For drought, NDVI had to be gap-filled carefully. Investing time in preprocessing and understanding your data will pay off more than any fancy model tweaks.
- **Robust Validation:** Don't just chase training accuracy. We stress using proper validation – spatial and temporal splits when appropriate. For instance, test your land cover model on a different region (spatial split) to check generalization. Or for time series, train on earlier years, validate on later year – does it still predict well? Use metrics beyond one number: **confusion matrices** for classification (to see which classes get confused), **IoU (Intersection over Union)** for segmentation quality, **RMSE and MAE** for regression, etc. Sometimes a model with slightly lower overall accuracy might be preferable if it performs more consistently across all classes or locations.

- **Avoid Overfitting:** Especially with small datasets, deep models can memorize noise. Techniques like cross-validation, early stopping, data augmentation (for images, e.g. random flips, rotations to increase sample variety) are important. We discussed this in earlier days: e.g., augmenting our Palawan land cover patches to make the CNN more robust. Always keep a hold-out test set that you don't touch until final evaluation.
- **Deployment Considerations:** The end goal is often to deploy the model in an operational setting (e.g., a drought early warning system or a web tool for land cover). This means considering things like inference speed (can the model run fast enough on available hardware?), scalability (can it handle new satellite scenes daily?), and integration (does it plug into existing GIS systems?). Also, maintenance: models may need periodic retraining as new data comes in (especially if sensor characteristics change or landscapes change – e.g., Taal volcano erupts and creates new land cover types, will your model handle that?). We advise to plan for an **operational pipeline** – from data ingest, preprocessing, running the model, to delivering outputs in a user-friendly format (maps, alerts, etc.).
- **Ethical and Responsible AI:** Not exactly a focus of this technical training, but worth a mention. Ensure the models are used for the benefit of communities and consider any unintended consequences. For example, if an AI model identifies households at "high flood risk", ensure that information is used to help (like prioritize assistance) and not to, say, deny insurance without human review. Also, be mindful of biases – if your training data is all from Luzon, the model might not work well in Mindanao due to differences in environment; that's a bias in geographic representativeness you should acknowledge.

By following these practices, participants will be equipped not just to build models, but to build **reliable, real-world solutions**.

CopPhil Digital Space Campus and Continued Learning

Learning doesn't stop when this workshop ends! We are making all materials available on the **CopPhil Digital Space Campus**, an online portal for EO and AI learning. There you will find: - Slides and lecture notes for all sessions. - Colab notebooks with code for the hands-on parts (you can run them anytime, tweak parameters, and even use them as templates for your own projects). - Datasets or links to where to download the example datasets used (e.g., the flood mapping SAR patches, the NDVI time series CSVs). - Additional **guides and tutorials**, possibly curated from external resources, on related topics we couldn't cover in depth.

The idea is to enable *self-paced learning* after the workshop [55]. Maybe you got intrigued by U-Net but want to try it on a different dataset – the Campus resources will help you do that. We'll also post **bonus content** like how to set up your environment for GPU training, how to label your own data (if you want to create a custom dataset), etc.

Additionally, you'll have access to discussion forums or a chat group (if provided) where you can ask questions even after the training. The instructors and perhaps a broader community will be there. This platform essentially extends our four-day training into a continuous learning journey.

For example, if in 2 months you attempt a project to map **urban heat islands** in Cebu using AI, and you encounter issues, you could refer back to Campus notebooks or ask in the forum, "Hey, I have LST time series, how to handle missing data?" – Perhaps a fellow participant or instructor will guide you (maybe referencing the LSTM session solutions).

We strongly encourage everyone to take advantage of these resources and not be shy about continuing to experiment. AI/ML is a fast-moving field – new techniques emerge, and the Campus will also update with significant new developments (for instance, if a new foundation model for EO from PhilSA or abroad comes out, we might add a note or tutorial about it).

Fostering a Community of Practice in the Philippines

One of the most important outcomes of this training is hopefully the network you form with each other. We want to **foster a community** of EO + AI practitioners in the Philippines [56] . Why? Because local challenges are best solved collaboratively. You now all share a common skillset baseline; by keeping in touch, you can share data, co-develop models, avoid duplicate efforts, and present a united voice on what tools/methods are needed for the country.

National Initiatives to Connect With: There are several programs and institutions actively building AI/EO capacity: - **SkAI-Pinas:** The "Sky AI" program, led by DOST and partners, aims to democratize AI for Filipino users. It has sub-initiatives like **DIMER** (Decentralized Intelligent Model Exchange Repository) and **AIPI** that were mentioned [56] . DIMER is essentially a model hub where people can share and access pre-trained AI models tailored to local needs (imagine a library of models – you can grab a pre-trained model for crop classification in Central Luzon, for example) [57] . AIPI likely refers to an AI Philippines Initiative (possibly focusing on making AI accessible daily – unfortunately the acronym wasn't fully spelled out in our materials, but it's part of SkAI-Pinas efforts). SkAI-Pinas also involves building infrastructure (maybe an AI compute platform) and holding events (like an annual congress) to gather practitioners [58] . By engaging with SkAI-Pinas, you can stay at the forefront of AI in the country, contribute models to DIMER, or use models others provide. - **PhilSA (Philippine Space Agency) Programs:** PhilSA, being in charge of national space and remote sensing, has platforms like the Space Data Dashboard and leads projects on satellite development (e.g., Diwata, MULA satellites). They also have research initiatives in applying AI to satellite data for DRR, climate, etc. PhilSA often collaborates with DOST on challenges such as flood monitoring and agriculture mapping. Being aware of PhilSA's projects (like if they start an AI for satellite data working group) could provide avenues to apply what you learned on national scales [59] . - **DOST Programs:** Aside from SkAI-Pinas, DOST-ASTI manages a lot of relevant infrastructure (such as computing facilities like the **COARE** supercomputer mentioned in news, or data hubs). They also run the **Philippine Research, Education, and Government Information Network (PREGINET)** which might host data and tools. DOST is very keen on AI as seen by investment in projects and establishing guidelines (they have an AI roadmap 2019–2029, etc. as per recent news). Getting involved in DOST-led pilot projects (they often have calls for proposals or pilot site implementations) can amplify your work. - **Academe and NGOs:** Universities like UP, Ateneo, DLSU have research labs on AI and environment. There are also NGOs using geospatial AI for conservation (e.g., monitoring forest cover in Sierra Madre). By linking with them, you can tackle real problems with mentorship and possibly funding. We encourage joining local AI or geospatial meetups (e.g., via PSSEA – Philippine Society of Surveying and Geomatics, or the Computing Society conferences). - **International collaborations:** The community you form can also collectively interface with international programs (like GEO, or AI for Earth initiatives). As a community, you can propose projects, share data internationally, and attract collaborations that benefit the country.

The training's final part is an **open Q&A** – we expect you might have questions that arose during your hands-on work or as you plan to use these tools back in your job. We'll address any unresolved issues (e.g., "I tried to run the flood model on my province's data and it fails, what could be wrong?"). This is also the time to discuss *specific challenges you anticipate* in your agencies or projects [60]. For example, maybe *data sharing* is a challenge (one agency holds data needed by another to train a model). In the community spirit, we might brainstorm how to facilitate that (like using the Digital Campus as a neutral repository or drafting data-sharing MOUs).

Finally, we'll gather feedback to improve future trainings [61]. But more importantly, we want you to leave empowered with concrete next steps: perhaps you will start a pilot project (like *"AI for Mangrove Mapping in Mindanao"*), or you'll champion setting up an AI unit in your bureau. We want to ensure you know what support is available – e.g., maybe CopPhil or DOST can provide some advisory support, or you now know whom to call (one of your fellow trainees!) when you need help labeling data or troubleshooting code.

The journey continues – today you are equipped with knowledge and initial practice; tomorrow you apply it and keep learning. The Philippine EO and AI community is growing, and by being part of it, you contribute to making our country more resilient, sustainable, and technologically advanced.

Think-Through: Which Model for Which Problem?

As a concluding exercise, imagine a table of environmental problems and AI models. For each problem, which model (or combination) would you pick and why? For example: - *Coral Reef Mapping (segmenting coral vs sand vs seagrass)* – likely a U-Net (segmentation) using multispectral water-penetrating bands. - *Volcano Eruption Early Warning (time series of seismic signals + satellite thermal images)* – maybe an LSTM (for seismic temporal patterns) combined with a CNN (for thermal imagery), or even a hybrid. Discuss your reasoning. - *Automatic Counting of Vehicles in Drone Imagery for traffic analysis* – an object detection model (like YOLO) is apt. - *Predicting landslide occurrence* – possibly an ensemble where an RF uses factors like slope, rainfall (structured data) alongside an CNN that analyzes terrain imagery. Or simpler, an RF if all inputs are already features. This thought experiment makes you solidify what you've learned by mapping problem requirements to model capabilities.

Think-Through: What Does "Explainable Enough" Mean?

Reflect on the organizations you belong to. Some might be very conservative, preferring methods that engineers can manually check (e.g., if it's a flood model, they might prefer a rule-based approach because they can understand it fully). Others might be more open to black-box models if they've been validated. Ask yourself: *For my organization to adopt an AI model, what level of explainability is required?* Would showing SHAP plots of feature importance suffice? Do we need a full narrative explanation for each prediction? Or is accuracy king and they'll accept a black box if it demonstrably works? By anticipating this, you can choose tools accordingly – maybe you'll lean more on XAI techniques when deploying in a highly accountability-focused setting (like disaster response), whereas for a quick-and-dirty forest cover map, you might not need to justify every pixel decision.

Answering these questions will help you tailor your future AI projects to the real-world constraints and expectations of your stakeholders.

End of Day 4. Congratulations on completing the training! Remember, this is a springboard – keep practicing, keep learning, and support each other. With these AI/ML tools in your arsenal, you are now better equipped to tackle pressing EO challenges in the Philippines, from climate adaptation to disaster resilience and beyond. Go forth and innovate!

Sources: The content above has drawn on the CopPhil training agenda and various references for technical explanations. Key references include the training document for Day 4 [62] [4] [21], Wikipedia descriptions of LSTM architecture [6], IBM and Radiant Earth announcements on geospatial foundation models [24] [27], and Philippine government news on AI initiatives [57], among others, to ensure accuracy and relevance. All images and diagrams are used for educational purposes to illustrate concepts (e.g., LSTM cell structure [6] and SatMAE masked autoencoder approach [31]).

---

[1] Featured Image: Kubina, Karachaevo-Cherkessia, Russia | by PHL-Microsat | PHL-Microsat
https://blog.phl-microsat.upd.edu.ph/featured-image-kubin-karachaevo-cherkessia-russia-1a247aa7d350?gi=20a875a64bfa

[2] [3] [4] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [21] [22] [37] [40] [41] [42] [43] [45] [46] [47] [48] [49] [50] [51] [52] [53] [54] [55] [56] [59] [60] [61] [62] CopPhil EO AI_ML Training Agenda - Final - 040725.docx
file://file_000000004c30620aa965644877a59b86

[5] [6] [7] Long short-term memory - Wikipedia
https://en.wikipedia.org/wiki/Long_short-term_memory

[20] Reflections on Foundation Models | Stanford HAI
https://hai.stanford.edu/news/reflections-foundation-models

[23] Foundation Models for Geospatial Analytics (Collaboration with ...
https://www.clarku.edu/centers/geospatial-analytics/projects/prithvi-foundation-model/

[24] [25] [26] ibm-nasa-geospatial (IBM-NASA Prithvi Models Family)
https://huggingface.co/ibm-nasa-geospatial

[27] [28] [29] Clay Foundation Model — Clay Foundation Model
https://clay-foundation.github.io/model/index.html

[30] [31] [38] [39] SatMAE: Pre-training Transformers for Temporal and Multi-Spectral Satellite Imagery
https://sustainlab-group.github.io/SatMAE/

[32] zhu-xlab/DOFA: Code for Neural Plasticity-Inspired ... - GitHub
https://github.com/zhu-xlab/DOFA

[33] [34] [35] [36] earthflow/DOFA · Hugging Face
https://huggingface.co/earthflow/DOFA

[44] Machine Explainability: A Guide to LIME, SHAP, and Gradcam
https://suryansh-raghuvanshi.medium.com/machine-explainability-a-guide-to-lime-shap-and-gradcam-60f6265f365f

[57] DEPDev: PH harnessing AI for innovation, digital transformation | Philippine News Agency
https://www.pna.gov.ph/articles/1258998

[58] Democratizing AI for Every Filipino: SkAI-Pinas 3rd Congress ...
https://asti.dost.gov.ph/news-articles/skai-pinas-3rd-congress-champions/