ChatGPT

Day 4 Session 1: LSTMs for Earth Observation Time Series

Module 1: Introduction to Time Series in EO

Earth Observation (EO) **time series** are sequences of satellite-derived measurements tracked over time (e.g. monthly values over several years). A common example is a time series of **Normalized Difference Vegetation Index (NDVI)**, which measures vegetation greenness. Unlike a single image, a time series reveals **temporal patterns** – how a value changes with seasons or years. For instance, NDVI time series can capture **phenology** (seasonal plant life-cycle patterns), detect **droughts** (unusual drops in greenness), or monitor **land cover change** (gradual trends when forests are cleared or urban areas expand). Radar data (like **SAR backscatter**) also produce time series useful for tracking surface moisture or disturbances over time [1]. In short, time series add the critical dimension of time, helping us see changes and trends that single snapshots would miss.

- **Phenology:** NDVI time series show seasonal cycles – for example, greenness rising in the wet season and falling in the dry season. By examining these cycles, one can identify the **start of the growing season, peak growth, and senescence** of vegetation. In croplands, distinct peaks might correspond to planting and harvest times.
- **Drought monitoring:** Vegetation indices like NDVI tend to drop below normal levels during drought stress [2]. Tracking NDVI over time can thus signal drought onset and severity. (In the Philippines, **PAGASA** defines drought often by consecutive dry months; NDVI time series can corroborate these periods by showing vegetation browning.)
- **Land change:** Long-term EO time series can reveal gradual changes such as deforestation (a steady NDVI decline) or urbanization. For example, a forest converted to farms would show a different temporal signature than before.

NDVI time series over 3 years for an agricultural region, with a drought event in the second year (highlighted in orange). During the drought year, NDVI values are noticeably lower than in the non-drought years – the peak greenness in the rainy season is reduced and overall vegetation health dips. This reflects how a prolonged lack of rain (often associated with El Niño) causes vegetation stress, captured as a sharp drop in the NDVI curve. In the Philippines, the 2019 El Niño-related drought caused significant agricultural losses [3], and a similar NDVI dip would be observed during that period. Time series like this help analysts visually pinpoint when an ecosystem deviated from its normal vegetative cycle due to drought.

Think-Through: *Why do we need a time series for this analysis? Consider a single NDVI image of Mindanao vs. a 3-year NDVI time series. The single image might show where vegetation is green or dry at that moment, but the time series shows when and how those areas changed. How could observing NDVI over an entire year help distinguish seasonal dry spells from an abnormal drought? Think about what patterns (timing, duration of low NDVI) would indicate a drought versus a normal dry season.*

Mini-Challenge: *Examine the NDVI plot above. Identify which year experienced the drought and list two visual clues that support your conclusion. Next, suppose you have NDVI time series data for 5 different provinces – how might you determine which province had the most severe drought impact? Outline a simple approach (e.g., comparing how far below the normal baseline each province's NDVI dropped).*

Module 2: RNNs and the Vanishing/Exploding Gradient Problem

Traditional neural networks (like feed-forward networks) assume inputs are independent, but many EO problems are sequential – yesterday's observation influences today's. **Recurrent Neural Networks (RNNs)** address this by introducing a **hidden state** that carries information from one time step to the next. At each time step of a sequence, an RNN takes the current input (e.g. NDVI at time *t*) and the previous hidden state (which contains info from time *t-1*) to produce a new hidden state and an output. In essence, an RNN has a *memory*: it "remembers" past data through the hidden state, enabling it to model sequences (like time series). Formally, one can imagine an update rule such as *h<sub>t</sub> = f(h<sub>t-1</sub>, x<sub>t</sub>)*, where *h<sub>t-1</sub>* is the past state and *x<sub>t</sub>* is the current input. This feedback loop allows the RNN to **influence the current output based on prior inputs** [4]. For example, an RNN could predict today's flood level by considering not just today's rainfall but also the accumulated effect of previous days' rain (stored in its state) [5].

However, training standard RNNs across long sequences is challenging due to the **vanishing and exploding gradient problem** [6]. When we train an RNN (or any deep network) via back-propagation through time, the error gradients are propagated backwards through many time steps. If the sequence (or the network) is long, gradients can **vanish** – they shrink exponentially as they go back in time, effectively **fading out** by the time they reach the early timesteps. This is like trying to recall the start of a very long lecture: by the end, the early points are so faint in memory that they hardly influence your understanding (the network "forgets" what happened long ago). Conversely, gradients can also **explode** – they grow exponentially and become astronomically large, causing wildly large weight updates that destabilize learning (the network's parameters effectively overflow, leading to numerical issues). In our lecture analogy, this would be like information overload: each point builds up so much that your "memory" of earlier content becomes erratic or overwhelmed, making learning impossible.

Why do these problems occur? A key reason is the repeated multiplication of gradients through each time step. If the effective weight is less than 1, multiplying many such terms drives the gradient towards zero (vanishing); if greater than 1, the gradient blows up. RNNs using activation functions like sigmoid or tanh are especially prone to vanishing gradients, since those activations squash values between -1 and 1. In practice, this means a vanilla RNN has difficulty learning long-term dependencies – e.g., relating an NDVI anomaly one year to a crop yield drop the next year – because the signal linking those far-apart events diminishes during training [6]. Researchers observed that basic RNNs perform well for short sequences but struggle as sequence length grows due to this limitation.

Think-Through: *Imagine you're tracking a simple pattern: a spike in NDVI should cause a drop in some index 10 steps later. A basic RNN might know the last few NDVI values well, but why would it have trouble learning an effect 10 steps back? Think about what happens to the error signal that needs to travel back those 10 steps during training. If each step even slightly diminishes the signal (say multiplies by 0.9), by the time it reaches step 10 it's barely 0.9^10 of its original strength. Reflect on how this "fading signal" is analogous to forgetting the early part of a lecture by the time it ends.*

Mini-Challenge: *Let’s do a quick mental experiment. Start with the number 1.0 as a “gradient.” Multiply it by 0.5 fifty times (0.5^50). What do you get (roughly)? Essentially zero! This illustrates vanishing gradients. Now take 1.0 and multiply by 1.5 fifty times (1.5^50). The number grows astronomically – an example of exploding gradients. For a mini-challenge, calculate (roughly or using a calculator) 0.5^10 and 1.5^10. How do these smaller sequences compare to the 50-step case? What does this tell you about the ease of learning dependencies 10 steps long vs. 50 steps long in an RNN?*

Module 3: LSTM Architecture and Intuition

To overcome the limitations of standard RNNs, researchers developed **Long Short-Term Memory (LSTM)** networks. An LSTM is a special kind of RNN that introduces an internal **memory cell** and a set of gating mechanisms that regulate information flow. The core idea is to let the network **learn what to keep, what to throw away, and what to output** at each time step [7]. This architecture preserves long-term information much better, hence the name “long short-term” memory – it retains long-term dependencies while still focusing on short-term inputs [8].

At the heart of the LSTM is the **cell state**, often visualized as a horizontal line running through time, which is like a conveyor belt carrying information forward largely unchanged. The network can add information to this cell or remove information from it using structures called **gates**: - **Forget gate:** decides what old information to erase from the cell. It looks at the previous hidden state and current input and outputs a number between 0 and 1 for each piece of the cell state (1 means “keep this completely”; 0 means “drop this”) [9]. For example, if a certain signal is no longer relevant (e.g., a note that the last rainy season was wet, when now a new rainy season is starting), the forget gate can remove it. - **Input gate:** decides what new information to write to the cell. It uses a similar 0-1 scaling to determine which parts of the new candidate information are important enough to store [10]. Continuing the example, if the current month’s NDVI indicates vegetation is flourishing, the input gate might allow this to be added to the memory cell as an indicator of good conditions. - **Output gate:** decides what information from the cell to output (to the hidden state) at the current time step [11]. This gate controls the exposure of the memory – e.g., even if the cell has remembered a long-term drought trend, the network might not need to **output** that info at every step, only when relevant to the prediction.

Diagram of an LSTM cell. The cell state (long horizontal bar) carries accumulated information. Sigmoid-based gates (σ) regulate the flows: the forget gate (typically noted as f) controls what to drop from the cell state, the input gate (i) controls what new content to add (in combination with a candidate value, often via a tanh layer), and the output gate (o) controls what final hidden state (h) to output from the cell. By adjusting these gates at each time step, an LSTM can decide to retain long-term information (e.g., a seasonal trend), forget irrelevant details, and output only the pertinent information for making a prediction [7]. This gating mechanism allows gradients to flow back through time more effectively, mitigating the vanishing gradient problem – important information can be kept in the cell state for dozens of steps without being constantly overwritten.

An **analogy** for an LSTM is a **smart notebook** that a student uses during a lecture. Think of the cell state as the notebook’s running notes. At any point: - You decide to **forget** certain info: maybe you cross out or ignore older notes that are no longer relevant – akin to the forget gate. - You decide what to **write down** from the current discussion – akin to the input gate filtering in new notes (only the key points, not the trivial chatter). - When asked a question, you decide what part of your notes to **read out or use** – akin to the output gate deciding what from the cell state should influence the response.

Using this notebook, you can retain important points from the beginning of the lecture until the end (long-term memory), while not cluttering your mind with every detail. In the same way, LSTMs can maintain information over long sequences without losing it or blowing up, by **learning** to balance remembering and forgetting.

Think-Through: *If an LSTM were monitoring a yearly NDVI cycle, how might it use its gates? Imagine last year had an extreme drought. The LSTM might input that event strongly into its cell state (so it "remembers" that a drought occurred). As time moves on, if conditions return to normal, the forget gate might gradually lessen the influence of that old drought memory (since after a few years it might be less relevant). When making a prediction (say, forecasting crop yield or next month's drought risk), the output gate will decide if that stored drought information is needed now. Why is this selective memory useful? Think about how some events have long-term effects (a drought can impact multi-year water resources), while others are short-lived. The LSTM can learn to carry important long-term signals and drop others.*

Mini-Challenge: *To test your understanding, consider two extreme scenarios: (a) the forget gate is always 1 (it never forgets anything in the cell); (b) the forget gate is always 0 (it wipes the cell state every time step). Discuss what each scenario means. In case (a), what might happen to the cell state over time? (Hint: it might accumulate a lot of outdated information – good or bad?) In case (b), what does the LSTM reduce to? (Hint: if you reset memory at every step, can it learn long-term dependencies at all?) By analyzing these extremes, we appreciate why a balance (learning when to forget vs. retain) is crucial.*

Module 4: LSTM Use Cases in EO

LSTMs have opened up many applications in Earth Observation where temporal dependence is key. Here are a few **use cases** relevant to EO and especially to the Philippines context:

- **Drought forecasting:** Perhaps the most pertinent example – using LSTMs to predict drought indicators from time series data. For instance, feeding an LSTM with the past 12 months of NDVI (and even rainfall data) to predict next month's drought status (e.g., whether a drought will occur, or the value of a drought index). Studies show that LSTMs are among the most effective models for drought prediction, often incorporating indices like NDVI, rainfall-based indices (SPI, SPEI), etc., as inputs [12]. In fact, NDVI is one of the most widely used satellite indices for drought monitoring [13] given its link to vegetation health. An LSTM can learn patterns such as "if vegetation has been steadily declining and rainfall is below average for several months, a drought is likely to intensify next month." This can improve early warning systems for droughts.

- **Crop yield prediction:** Agricultural yields often depend on the weather and vegetation conditions throughout the growing season. By inputting a time series of variables (NDVI, rainfall, temperature) leading up to harvest, an LSTM can predict final crop yield. For example, the rice yield in **Mindanao** could be predicted by looking at NDVI patterns during planting and growth stages. If the NDVI time series shows a truncated growing season or stress at critical times, the LSTM might learn to correlate that with lower yield. Compared to simpler regression, the LSTM can capture the timing of events (e.g., a dry spell during flowering vs. during planting might have different yield impacts).

- **Phenology and land cover change:** LSTMs can classify or cluster time series to detect what kind of land cover a pixel is (e.g., distinguishing forest, rice paddies, grassland by their seasonal NDVI signatures) or to detect changes. A forest that suddenly loses greenness outside of the normal seasonal variation might indicate deforestation or fire. A well-trained LSTM could potentially flag such anomalies by learning the normal sequence pattern and detecting deviations. Similarly, phenological metrics (start of season, end of season) can be derived or even predicted for ecosystems using LSTMs, which could help in climate change studies.

- **Multi-sensor data fusion:** In EO we often have multiple data streams (satellite imagery, climate data, soil moisture sensors). LSTM networks can ingest multiple synchronous time series – for instance, NDVI and rainfall together – by having multiple features at each time step. This way, they learn the interactions (e.g., vegetation typically grows after rain; if rain falls but NDVI doesn't increase, that might signal something unusual). This is particularly useful for drought and flood forecasting.

Let's consider a **Philippine example** in detail: **Drought prediction in Bukidnon (Mindanao).** Suppose we have a dataset of monthly NDVI values for farmland in Bukidnon over several years, and we also have a drought index like SPEI (Standardized Precipitation-Evapotranspiration Index from PAGASA) or monthly rainfall totals (e.g., from CHIRPS satellite data) for the same periods [14]. We want to predict next month's drought index level from the past year of data. We can set up an LSTM where each input sequence is the NDVI (and possibly rainfall) for the past 12 months, and the output is the drought index for the next month. By training on historical data, the LSTM will adjust its weights to connect patterns in the input sequence with the subsequent drought outcome. For example, a pattern of steadily declining NDVI combined with low rainfall might precede a drought classification of "severe" in the target output.

Conceptual flow of an LSTM model for drought forecasting. The model takes a sequence of NDVI values from the past 12 time steps (e.g., the last 12 months) as input, processes them through its LSTM layers, and outputs a prediction of the drought index for the next time step (t+1). In practice, multiple features can be used – for instance, NDVI and rainfall together at each time step – to improve predictions. This sequence-to-value approach allows the model to learn how the trajectory of vegetation health over time relates to future drought conditions. By training on past data (where both NDVI sequences and actual drought outcomes are known), the LSTM learns to forecast, say, a high drought index when it sees the telltale precursor signs in the NDVI trend.

Such an LSTM-based system could give **one-month lead time** drought warnings. This has huge implications for agriculture: farmers and authorities could prepare if the model forecasts a drought next month based on current trends. Notably, LSTMs have been successfully used in many countries for drought prediction; they excel at this task compared to older statistical models [12] because they can capture non-linear patterns and memory over long sequences.

Think-Through: *Why use an LSTM for these EO problems? Consider drought forecasting: a single month's NDVI might not tell you much (maybe it's low because it's normally a dry month). But an LSTM can consider context – the sequence of values. It might recognize "the last 6 months NDVI have been consistently below normal" as a stronger drought signal than any single month in isolation. Think of another example: crop yield depends on a combination of events (adequate rain early on, no drought during flowering, etc.). A time series model can encode these time-dependent effects. How is this fundamentally different from doing a simple regression on the average NDVI or last month's NDVI alone? (Hint: sequence models preserve the order and timing of events, which often matter greatly in EO contexts.)*

Mini-Challenge: *Brainstorm another EO scenario that could benefit from an LSTM. For instance, flood risk forecasting: could we use a time series of soil moisture or river levels to predict floods? Or urban growth:*

analyzing night-time light intensity changes over time to predict new urban areas. Pick one scenario and outline what the input sequence and output prediction would be. Why do you suspect an LSTM would be helpful here? (Think in terms of needing to capture trends or periodicity or long-term dependencies in the data.) Finally, for the drought case we discussed, list the types of data you would feed into the LSTM (NDVI, rainfall, etc.) and what output you'd expect. This will help solidify how to set up an LSTM for a real-world EO application.

---

[1] [14] CopPhil EO AI_ML Training Agenda - Final - 040725.docx
file://file_000000004c30620aa965644877a59b86

[2] The time series of NDVI presented in the same format as Figure 6. | Download Scientific Diagram
https://www.researchgate.net/figure/The-time-series-of-NDVI-presented-in-the-same-format-as-Figure-6_fig7_363062282

[3] philsa.gov.ph
https://philsa.gov.ph/wp-content/uploads/2023/10/3D-02-Ang-et-al.pdf

[4] [5] [6] What is a Recurrent Neural Network (RNN)? | IBM
https://www.ibm.com/think/topics/recurrent-neural-networks

[7] [8] [9] [10] [11] Long short-term memory - Wikipedia
https://en.wikipedia.org/wiki/Long_short-term_memory

[12] [13] Characterizing drought prediction with deep learning: A literature review - PubMed
https://pubmed.ncbi.nlm.nih.gov/38989261/