Datenverwendung:

Pretraining: Alleinige Verwendung von ERA5-Daten. Dafür werden synthetische Gaps, die der zeitlichen Dynamik der Satellitendaten entsprechen eingefügt.

Finetuning: Verwendung von ERA5 (keine SM), Satelliten-SM und in-situ-SM

Trainingsdatenauswahl

Das LSTM arbeitet immer nur auf einem bestimmten Kontext-Window mit fix vorgegebener Länge (z.B. 180 Tage). Die Länge ist ein Hyperparameter, den man tunen kann, auf jeden Fall wärs aber sinnvoll, wenn die deutlich länger ist als der längste Gap. Entsprechend dieser vorgegebenen Länge alle Trainings/Testdaten aufzubereiten.

Ein einzelnes ERA5-Pixel, das z.B. 10 Jahre umfasst, kann also genutzt werden, um eine Menge an Trainingsbeispielen zu erzeugen, z.B. Tag 1-181, Tag 2-182, … (den Offset zwischen den Tagen nennt man „Stride“, in dem Fall also 1). Denke für den Anfang wäre eine Wert von mindestens 10 sinnvoll. Wenn wir also sagen, dass wir (fürs Erste) 100 Pixel verwenden und für jedes Pixel z.B. 150 Zeitreihen, hätten wir 15.000 Beispiele. Die müssten dann in Trainings- und Testdatensatz gesplitted werden (z.B: 70%/30). Gut wäre ein spatial Split (bestimmte Pixel nur für Training, andere nur für Test). Modell darf Testdaten nicht schon vom Training kennen.

LSTM Parameter

Batchsize: Ist ein Hyperparameter, der festlegt, wie viele Beispiele (Zeitreihen) verwendet werden, bevor ein Update der Modellparameter (Gewichte/Biase) durchgeführt wird. In jedem Durchlauf eines einzelnen Beispiels wird in jedem LSTM-Modul ein Output für den jeweiligen Zeitpunkt berechnet. Kleine Batchsize führt zu rauschigem Gradienten, was helfen kann lokalen Minima zu entkommen, dauert aber länger. Wahrscheinlich guter Startwert: 32

LSTM Units (hidden_size): Ist die Dimension des Hidden States und Cell States. Gute erste Wahl: 64 (führt dann zu einem Bi-LSTM Output der Länge 128). Zu grosse Werte können schnell zu Overfitting führen.

Lernrate: Schrittweite. Gute erste Wahl: 0.001 (Fürs Finetuning evtl. mit kleinerer Lernrate starten. z.B. 0.0001, damit das im Pretraining gelernte physikalische Grundverständnis nicht sofort durch rauschende Satellitendaten überschrieben wird).

Optimizer: Gute Wahl: Adam. Passt die Lernrate für jedes Gewicht individuell an.

Gewichts Initialisierung: Gute Wahl: Glorot Uniform

Sequence Length: Ist die LSTM Kontext-Breite. Schon oben besprochen.

Epochs: Gibt an wie oft das Modell während des Trainings alle Beispiele aus dem Trainingsdatensatz sieht. Evtl. nicht auf fixer Epochenanzahl trainieren sondern EarlyStopping verwenden (Training wird, sobald die Validation Loss für eine gewissen Anzahl an Epochen nicht mehr sinkt, abgebrochen)

Loss Function:

Verwendung einer gewichteten Loss Function, in der Gaps ein Gewicht (wahrscheinlich im Bereich so um 10 sinnvoll) zugewiesen wird, ausserhalb davon ein Gewicht von 1. Das sorgt dafür, dass das Modell primär lernt die Lücken zu füllen, gleichzeitig aber auch bestraft wird, wenn die Ränder der Lücken nicht sauber an die echten Daten andocken (d.h. die verfügbaren umgebenden SM Daten z.B: als Ausgangspunkt dienen). Die besprochene, harte Masked-Loss sollte aber auch ok sein.

$$Loss = \frac{1}{N} \sum_{t=1}^{T} w_t \cdot (SM_{pred,t} - SM_{true,t})^2$$

Grundsätzliche Architektur

- **Input Layer** → Modell erhält Input in shape (LSTM Kontext Window Size, Anzahl Features).
- **Bi-LSTM Layer 1** → Um kurzzeitigen Zusammenhang zwischen den Features abzubilden. Durch die bidirektionale Komponente werden im Modell für jeden Zeitpunkt Informationen aus der Vergangenheit (vorwärts gerichteter Strang) und Zukunft (rückwärts gerichteter Strang) kombiniert. Zu beachten ist, dass dadurch die Anzahl der Units im Output sich verdoppelt (also wenn hidden_size auf 64 gesetz wurde, ist der Output des Bi-LSTMS 128- dimensional)
- **Dropout** → Zwingt das Modell Redundanz aufzubauen, damit das Modell nicht nur massgeblich von einzelnen Verbindungen abhängt. Nach jedem Update, also sobald ein Batch durchlaufen wurde, werden wieder zufällig neue Verbindungen ausgeschalten. Dropout ist nur aktiv, während dem Training, bei der Prädiktion werden wieder alle Verbindungen verwendet.
    - ***Standard Dropout*** (kappt zufällig Verbindungen zwischen den beiden LSTM Layern) (dropout=0.2 → 20% der Verbindungen werden gekappt). Üblich Werte zwischen 0.1 und 0.5.
    - ***Recurrent Dropout*** (kappt zufällig Verbindungen zwischen den Hidden States innerhalb eines LSTM Layers). Werte zwischen 0.1 und 0.3 üblich,

oft aber nicht verwendet (auf 0 gesetzt), da es das Training sehr verlangsamen kann. Also eher nicht verwenden.

- **Bi-LSTM Layer 2** → Inputs dieses layers sind die berechneten hidden states aus Bi-lSTM Layer 1. Durch Verwendung eines zweiten Layers, können zugrundeliegende Langzeittrends besser abgedeckt werden.

- **Dense Layer** → Erhält die Hidden States von Bi-LSTM Layer 2 als Input und gibt dann den Soil Moisture Wert aus. Transformiert daher für jeden Zeitpunkt den hochdimensionalen Hidden-State (bei Bi-LSTM (2*hidden_size)-dimensional) in einen skalaren Soil Moisture Wert. Die Gewichte sind hier für alle Zeitpunkte die gleichen (Time-Distributed Dense Layer). Sigmoid Aktivierungsfunktion macht im Output Sinn (Wertebereich 0-1). Um auf Soil-Moisture Skalierungsbereich [0.1,1] zu kommen (wird im nächsten Punkt erklärt) sollte der output so berechnet werden: y = 0.1 + 0.9 * sigmoid(out).

Features

All die aufgelisteten Features gehen als Input in das Modell. Evtl. könnten wir später noch mehr dazunehmen (z.B. Landcover), denke fürs erste reichen die aber. Damit das Modell nicht manche Features überpriorisiert und effektiv lernt, sollten alle auf eine ähnliche Range gebracht werden. Wichtig: die Min-Max-Skalierung sollte auf allen Daten des Trainingsdatensatzes beruhen. Testdaten sollten nicht miteinbezogen werden.

- **Soil Moisture**: Min-Max Scaling von SM in den Bereich [0.1,1]. An Tagen, an denen keine Soil Moisture Daten verfügbar sind, wird SM auf 0 gesetzt. Der Grund warum die invaliden SM Daten nicht klarer abgegrenzt sind ist, weil dadurch das Gradientenverhalten stabiler ist. Dadurch dass Sigmoid und Tanh als Aktivierungsfunktionen verwendet werden, ist die Ableitung am steilsten im Bereich Nahe 0. Je höher der Gradient, desto stärker ist auch die Anpassung der Gewichte. Wenn invalide Werte jetzt z.B. -999 wären, würde der Gradient sehr klein werden und es dadurch deutlich länger dauern bis das Modell versteht, warum SM so stark abfällt.

- **Precipitation**: Sollte log transformiert werden: $P' = \ln(1+P)$. Modell ist dadurch besser in der Lage keinen Regen von leichten Regen zu unterscheiden. Danach min-max skalieren auf [0,1].

- **Potential Evapotranspiration**: PET sollte standardisiert werden $PET' = (PET - mean(PET))/std(PET)$. PET folgt normalerweise einer Normalverteilung, würde Min-Max Scaling verwendet werden, würden Ausreisser zu suboptimalen Bereich führen.

- **Temperatur:** Verwenden von Skin Temperature. Sollte standardisiert werden (wie bei PET). Lufttemperatur evtl. später dazunehmen.

- **Day Of Year**: Tag innerhalb eines Jahres. Würde verwendet werden um Saisonalität besser abbilden zu können. Sollte encoded werden, sodass Modell nicht denkt, dass Tag 1 und Tag 365 in der Charakteristik sehr verschieden voneinander sind. Lösung: Aufteilen in zwei separate Features.

  DoY_sin = sin(2*PI * day / 365.25)

  DoY_cos = cos(2*PI * day / 365.25)

- **Maske**: 0 (SM nicht vorhanden), 1 (SM vorhanden) → Durch Verwendung einer Maske als Feature sollte das Modell mit der Zeit lernen, dass immer dann, wenn der Wert von dem Feature 0 ist, also ein Gap vorherrscht, der Soil-Moisture-Wert, der als Input reinkommt (immer 0 bei Gaps) zu ignorieren ist und stattdessen die anderen Features zur Aktualisierung des Hidden-States verwendet werden sollen.

- **Time delta**: Tage seit der letzten validen SM Daten. Proxy für Unsicherheit. Wenn wir dieses Feature dazunehmen, sollte das Modell lernen, dass wenn schon einige Tage vergangen sind, seit den letzten validen SM Daten, der Hidden-State des vorhergehenden Zeitpunkts weniger relevant ist. Sollte nach oben hin geclipped werden, da ab gewissen Punkt der letzte Soil-Moisture Wert vollkommen irrelevant wird (vllt. so 60 Tage). Ohne Clipping würde bei einem langen Ausreisser, die Auflösung für die häufigen und wichtigeren kurzen Lücken verloren gehen. Nach Clipping min-Max skalieren auf [0,1].

Evaluierung

(folgt)