USING DEEP LEARNING TO BACKCAST HYDROLOGIC RESPONSE AND INFORM LANDSLIDE EARLY WARNING SYSTEMS

by

JONATHAN SHEPPARD

A THESIS

Presented to the Department of Earth Sciences
and the Division of Graduate Studies of the
University of Oregon in partial fulfillment
of the requirements
for the degree of
Master of Science

June 2023

THESIS APPROVAL PAGE

Student: Jonathan Sheppard

Title: Using Deep Learning to Backcast Hydrologic Response and Inform Landslide Early Warning Systems

This thesis has been accepted and approved in partial fulfillment of the requirements for the Master of Science degree in the Earth Sciences by:

Joshua Roering	Chairperson
Leif Karlstrom	Member
Joseph Dufek	Member

and

Krista Chronister	Vice Provost for Graduate Studies

Original approval signatures are on file with the University of Oregon Division of Graduate Studies.

Degree awarded June 2023

THESIS ABSTRACT

Jonathan Sheppard

Master of Science

Department of Earth Sciences

June 2023

Title: Using Deep Learning to Backcast Hydrologic Response and Inform Landslide Early Warning Systems

Landslides are difficult to predict due to the influence of variable geologic and environmental factors, such as geomechanical properties, rainfall, ground saturation, topography, and earthquakes, exert on the probability of a slope failure. Deep learning (DL) models can accurately predict the site-specific hydrologic response on hillslopes using soil moisture, pore pressure, and rainfall monitoring data. Landslide early warning systems can utilize empirical thresholds from deep learning-derived soil hydrology properties to improve landslide hazard prediction accuracy. We study the possibility of improving a logistical regression-based landslide early warning system being used in Sitka, AK by incorporating pore pressure responses that correspond to past known landslide events. Because pore pressure records for past known events are nonexistent, we must backcast soil hydrology timeseries from weather records, without including antecedent soil hydrology as initial conditions. We assess the accuracy of predictions at various rainfall intensity thresholds made by a Long Short-Term Memory (LSTM) DL model trained on weather features compared to a model that includes antecedent soil hydrology conditions. We find that the average accuracy of our model decreases by up to 20% for important, high-intensity rainfall events.

DISCLAIMER

These data and analyses have not received USGS approval and as such are provisional and subject to revision. These plots are released on the condition that neither the USGS nor the U.S. Government shall be held liable for any damages resulting from its authorized or unauthorized use.

Without the motivation and support from my family, the guidance and patience from my advisor, the laughter and comraderie from my lab mates, or the winds of chance that pushed me to the University of Oregon, this work would've never come to fruition. Thank you all for the encouragement and dedication towards excelling and enlightening..

TABLE OF CONTENTS

Chapter Page

LIST OF FIGURES

Figure Page

I. INTRODUCTION

Rainfall-induced landslides are a deadly, recurring problem across the globe. In 2004-2016, more than 55,000 people lost their lives to landslides (Sim, Lee, & Wong, 2022) and economic losses were estimated to be at $20 billion annually, which is 17% of the total ($121 billion) yearly mean global disaster losses from 1980-2013 (Klose et al., 2016). In that time span, 4,862 fatal non-seismically induced landslide events were reported, with 79% being triggered by rainfall. Further, the loss of life disproportionately affects medium and developing nations, with only 5% of fatalities coming from highly developed nations (Lacasse et al., 2010; Lacasse and Nadim 2014), highlighting the importance of being able to assess the potential for landslide initiation via hydrologic response to rainfall on hillslopes in at-risk communities. This is often facilitated by hydrologic models characterized by varying degrees of complexity, such as empirical thresholds relating rainfall intensity and/or soil hydrologic measurements corresponding to past landslide events (e.g., Guzzetti et al., 2008; Wieczorek & Guzzetti, 2000; Mirus et al., 2018a), deterministic, physically based numerical models (e.g., Bellugi et al., 2015; Montgomery & Dietrich, 1994; Simunek at al. 2005), or models based on machine learning (Orland et al., 2020; Papacharalampous et al., 2019; Hewamalage et al., 2021; Lara-Benítez et al., 2021). The varying methods bring unique advantages and disadvantages: empirical relationships between soil hydrologic properties and rainfall provide an efficient means of characterizing regional landslide susceptibility, but have limited predictive capabilities (Bogaard & Greco, 2018; van Natijne et al., 2020). Conversely, physical models of variably saturated flow can be used to obtain thresholds and extrapolate beyond the observed period of record (e.g., Fusco et al., 2019; Thomas et al., 2018), but these methods require significant parametrization and computational resources.

More recently, it has been shown that machine learning models merge the benefits of both empirical thresholds and physically based models by providing low-cost, site-specific hydrologic response predictions that implicitly capture the characteristics of multi-dimensional variably saturated flow while relying on the quality and abundance of in-situ soil hydrology measurements (Orland et al., 2020; Shen et al., 2018). The Deep Learning (DL) model presented by Orland et al. (2020) provides an accurate and computationally efficient method of predicting the timing and magnitude of hydrologic response to rainfall on hillslopes in various soil conditions with as little as six months of training data. As a result, this tool may be useful for informing landslide early warning systems (LEWS) that account for antecedent conditions, such as soil moisture or pore pressure, that affect the likelihood of incoming storms to trigger landslides.

The community of Sitka, AK has implemented a LEWS based on a logistical regression statistical model using rainfall thresholds from recorded past landslide events (Patton et al., 2023). In an effort to investigate avenues of increasing model accuracy, pore pressure has been identified as a candidate additional parameter to incorporate. The current inventory of past landslides used to train the logistic regression model includes landslides from 2015 to 2020. Several soil hydrology monitoring instruments have been set up as early as 2020, resulting in current soil hydrology characteristics, but none for the corresponding landslide events prior to 2020.

Here we develop an strategy for training a DL model to backcast pore pressure values that encompass the entirety of their logistical regression landslide inventory that spans 2015 to 2020. The lack of past site-specific weather and soil hydrology data precludes the use of antecedent pore pressure data for training of a DL-based predictions of hydrologic response pre-

2020 and requires the use of a semi-local weather dataset to use for training. We begin by showing that using a semi-local weather record source is an acceptable proxy for local weather, where overlapping timeseries of precipitation are well correlated. We then explore the effect of antecedent weather conditions, number of predictions made per timestep, and number of predictions made per model epoch on model accuracy during various thresholds of high intensity rainfall events, which are of primary concern for landslide initiation. As we lack antecedent pore pressure conditions (i.e., the initial conditions) for model predictions, we further show that the overall model accuracy is slightly degraded but comparable to a model that otherwise includes antecedent pore pressure as a training input feature.

II. METHODS

2.1 Overview

To formulate and test the DL model for backcasting pore pressure responses, we first collected weather and soil hydrology timeseries from sources near Sitka, AK. The soil hydrology data is limited to recent years (2020-) and does not extend to the period covering the past landslide events. We use a longer record for weather that is collected at a nearby site (~4km away from the soil hydrology sensor location) as the model input and compared the precipitation measured at each site to ensure that the amounts were comparable, such that the rainfall at the nearby site source could be used as a proxy. We then explore model sensitivity and tune the model using three hyperparameters required by the DL algorithm. Model accuracy is then assessed by measuring the error during various rainfall intensity thresholds which have higher chances of initiating landslide events.

2.2 Study Area

We apply DL to assess hydrologic response in Sitka, a remote, steep community in southeast Alaska (Figure 1). The landscape surrounding Sitka is filled with steep hillslopes and thin volcanic soils, which are particularly susceptible to shallow-seated landslides. The climate in Sitka is characterized by high annual precipitation, mainly attributed to atmospheric rivers from September to December (Wendler et al., 2016; Sharma et al., 2015). As debris flows are commonly initiated by shallow landslides during intense precipitation, this leaves isolated communities in the region, like Sitka, exposed to the deadly hazards of debris flows.

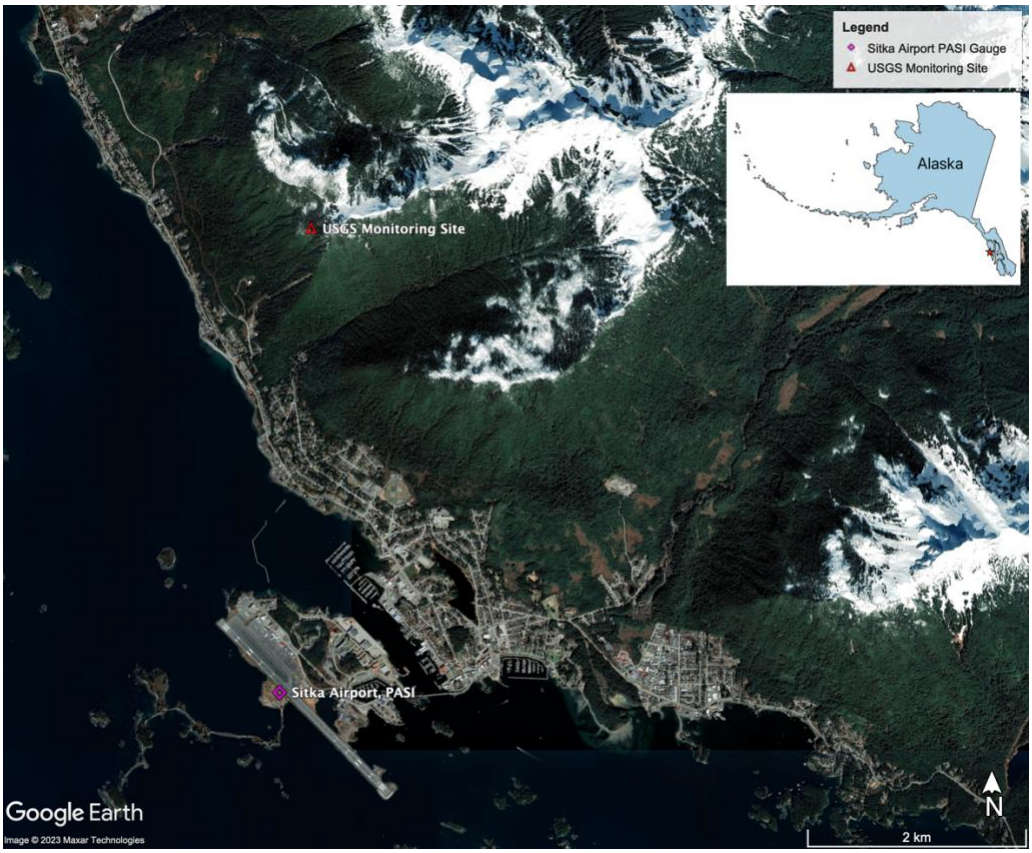

Figure 1. Study area. Google Earth image of Sitka, AK (©Landsat/Copernicus and Maxar Technologies, 2023). Shown are the PASI NWS station and the USGS Harbor Mountain monitoring station.

2.3 Data Source

We used soil hydrologic data collected from the USGS monitoring site on Harbor Mountain north of Sitka, AK as target features (Figure 1). The USGS monitoring station has sensors installed in two soil pits on the edge of a steep hollow similar to the initiation zones of past landslides. Both pits record soil moisture (VWC) and pore pressure at two depths and a single depth, respectively. The pits have different responses for soil moisture and pore pressure (Figure 2). We focus on targeting pore pressure predictions and chose to study pore pressures from soil pit 2 to build a responsive model more in line with observed landslide-initiating pore pressure responses. The instrument record includes 5-minute measurements of rain, VWC, and pore pressure acquired from 2020 to 2023.

For training features, we use weather records (primarily precipitation, but wind is also included) from the nearby Nation Weather Service (NWS) station (NWS station code PASI) operated by the Federal Aviation Administration (FAA) at Sitka Airport (NOAA NCEI, 2001) (Figure 1). Weather data has been recorded at the airport since 2002 in hourly to sub-hourly intervals.

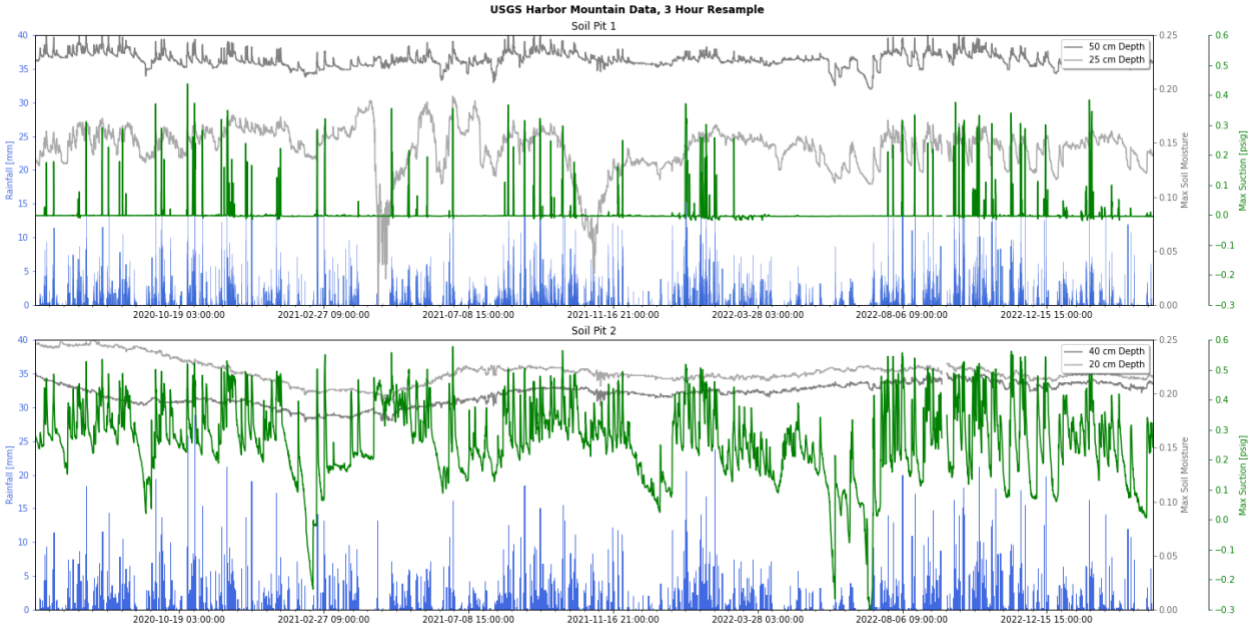

Figure 2. Compilation of USGS Harbor Mountain station records from June 2020 to March 2023 for precipitation, soil moisture, and suction (pressure) across both soil pits. Data has been down sampled from five-minute intervals to three-hour intervals. Precipitation is cumulative over the three-hour interval and soil moisture and suction are the maximum over the three-hour interval.

2.4 Source Precipitation Comparison

As depicted in Figure 1, the PASI gauge is 4.3 kilometers from the USGS monitoring station on Harbor Mountain, with a 575-meter elevation difference. As pronounced spatial heterogeneity in precipitation is typical of southeast Alaska (Patton et al., 2023), we performed a

correlation analysis on the local precipitation data of the USGS monitoring station and the semi-local PASI gauge.

2.5 Machine Learning Methods

Various machine learning methods have been applied to the field of geosciences, from Artificial Neural Networks (ANNs) (DeVries et al., 2017; Abrahart & See, 2007; Ren et al., 2019; Araya & Ghezzehei) to variations of the ANN, such as the Recurrent Neural Network (RNN) and the Long Short-Term Memory (LSTM) models. RNNs incorporate information from the previous timestep and make predictions based on a weighting between the past and present inputs. LSTM models advance this approach by incorporating an internal state which is propagated through time, allowing the model to handle longer term temporal dependencies. Furthermore, LSTM-based models are conceptually better suited than statistically-based autoregressive time series models, given a LSTM's capacity to approximate non-linear relationships between input and output variables, as opposed to assuming linear relationships between lagged endogenous or exogeneous variables in most autoregressive models (Box et al., 2008). For a more in-depth discussion of the LSTM architecture, we refer to Olah (2015), and for applications to hydrology, we refer to Kratzert et al. (2018).

We use the DL model presented by Orland (2020), which is an LSTM "encoder-decoder" model with a global Luong attention mechanism (Luong et al., 2015). The encoder-decoder architecture was built to solve sequence-to-sequence problems, such that both the input and output of a model are sequences of data. The encoder reads in and learns an input sequence, then feeds the input into a decoder which receives a fixed representation (understanding) of the input sequence. From this fixed representation, the decoder learns the proper non-linear

transformations to translate the input to an output sequence or variable length (Sutskever et al., 2014).

2.6 Data Preprocessing, Experimental Framework, and Parameter Tuning

Modeling steps include preparing input sequences into a set number of previous hours of data at three-hour resolution containing precipitation and wind data (referred to as antecedent weather), and the measured hourly maximum and cumulative precipitation for the next set number of hours of data (referred to as a forecast). These input data pair with the future pore pressure data from the current timestep to a set time in the future (referred to as the prediction span)(Figure 3). In doing so, we provide our model with both prior and anticipated weather information as inputs and seek to draw an explicit link between these inputs and the corresponding pore pressure response within a specified forecast period. Model training occurs by applying a moving window to the input provided, with a size equal to the antecedent weather and the prediction span and then finding the best set of non-linearized weights and biases that apply to current and past information which most closely match the observed pore pressure response.

We perform model tuning on three hyperparameters which show increased sensitivity to model performance and complexity. We assess model performance based on the RMSE of the model during rainfall events above a threshold amount of rainfall within three hours. The initial 2D hyperparameter space tested contained the first two hyperparameter investigated, the duration or length of antecedent weather and the prediction span of the future target pore pressure. These parameters combined to control the size of the temporal window the model is allowed to see at any given timestep. By varying these parameters, we effectively vary the amount of training data

our model has access to at any given timestep, which is a key component of any machine learning model.

The last model hyperparameter tuned is batch size. The batch size is a hyperparameter that defines the number of samples to evaluate? analyze? before updating the internal model parameters. A similar, but separate model hyperparameter is the model epochs, which is the the number of times that the learning algorithm will cycle through the entire training dataset when fitting. As model weights are set randomly upon initialization, each model run typically converges on a similar, but not necessarily identical solution. However, training for 2000-5000 epochs consistently results in a series of weights and biases that lead to comparable results across model runs.

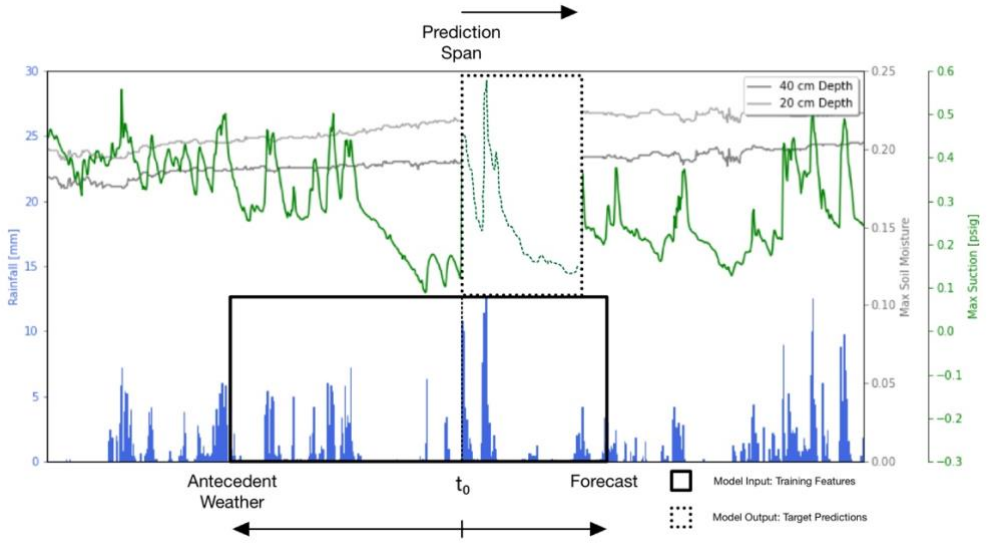

Figure 3. Conceptual diagram showing model inputs and outputs. During training, a moving window slides across input data provided to it (solid), and the model adjusts its weights to produce a sequence of pore pressure values (dashed) from the model inputs that best matches the observed pore pressure sequence for those timesteps (not pictured). This process repeats until the model converges on a set of weights and biases that produces the lowest mean squared error measured across all predicted and observed sequences.

III. RESULTS

In order to assess the correlation of rainfall amounts at the USGS Harbor Mountain station gauge and the Sitka Airport PASI gauge, we performed a pairwise comparison to compute the correlation coefficient for the two datasets. A correlation coefficient of 0.76 at a three-hour sampling interval was calculated for every sample in each dataset (Figure 4). As we are primarily interested in precipitation initiated shallow landslides, we focused on positive rainfall observations by filtering rainfall observations of 0 mm (dry conditions) out of both rainfall datasets and performed a similar analysis, which showed an increased correlation coefficient of 0.80. Both correlation coefficients are above 0.75 which indicates that the records are well correlated. Therefore, we conclude that it is reasonable to use the Sitka Airport PASI gauge record as a proxy for the local USGS station gauge as the DL model training feature input.

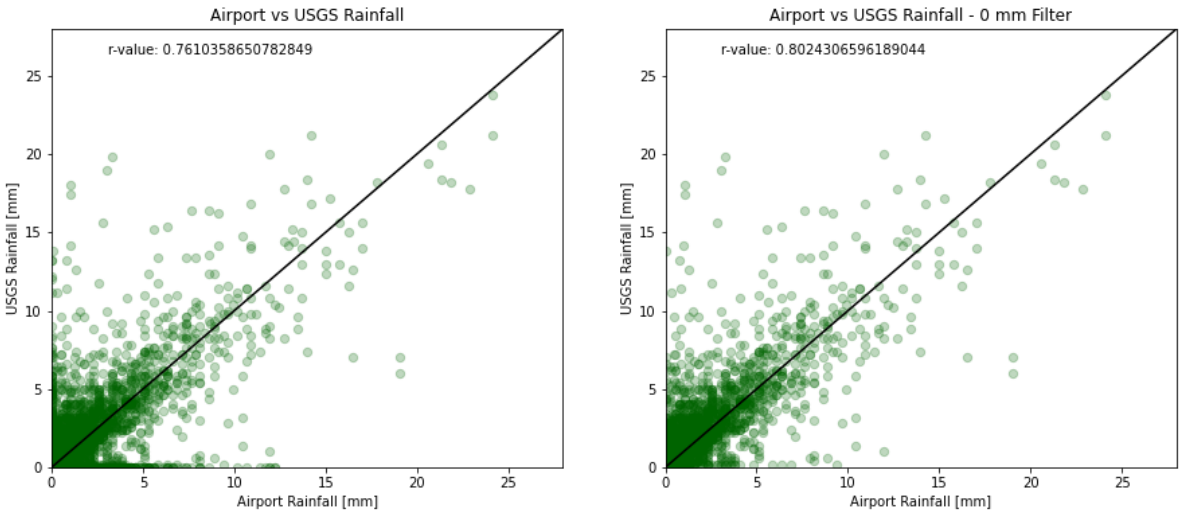

Figure 4. Pairwise comparison of precipitation data sourced from Sitka Airport PASI gauge and USGS Harbor Mountain station gauge, resampled to matching three-hour sampling intervals. Base comparison shows that there is a correlation coefficient of 0.76 between the records. When filtered such that only observations of non-zero precipitation are compared, the correlation coefficient becomes 0.80.

To prepare for model input, we first preprocessed our data. We chose to calculate a 72-hour forecast from our data, consistent with NWS forecasts of 3-hr rainfall intensity. Then we tune the model by finding the antecedent weather and prediction span that minimize the RMSE of pore pressure predictions during intense storm events above a threshold (Figure 5). We see an increase in model accuracy during intense storm events when we increase the antecedent weather length used in training and decrease the prediction span length being targeted. However, increasing antecedent weather length and decreasing the prediction span length both lead to longer model runtimes. Model weights are set randomly upon initialization, such that two models run sequentially with the same inputs should have a comparable but not necessarily identical RMSE values. We take into account this variability inherent in machine learning models by outlining the lower bounds of these parameters that would that result in a model that balances computational efficiency and accuracy during the high intensity events which are more likely to initiate a landslide than intervals with low precipitation. We determine that any antecedent weather length greater than 15 days into the past and any prediction span length less than 36 hours into the future would be sufficient to produce model accuracies comparable to the most accurate model run during parameter tuning.

To tune the batch size hyperparameter, we look for the balance of reproducibility, computational resource cost, and accuracy. An ideal batch size was found to be on the order of 100. A such, we employ a mini-batch gradient descent, which means that our batch size is greater than 1, but less than the size of our training data set. Smaller batch sizes showed increasingly diminished ability to predict the largest and smallest values of observations, while larger batch sizes show decreasing returns on accuracy in exchange for decreasing returns on computational

cost savings (Figure 6). We run our model fitting for 3000 epochs to minimize the variability of model weights and ensure comparable results between model runs.

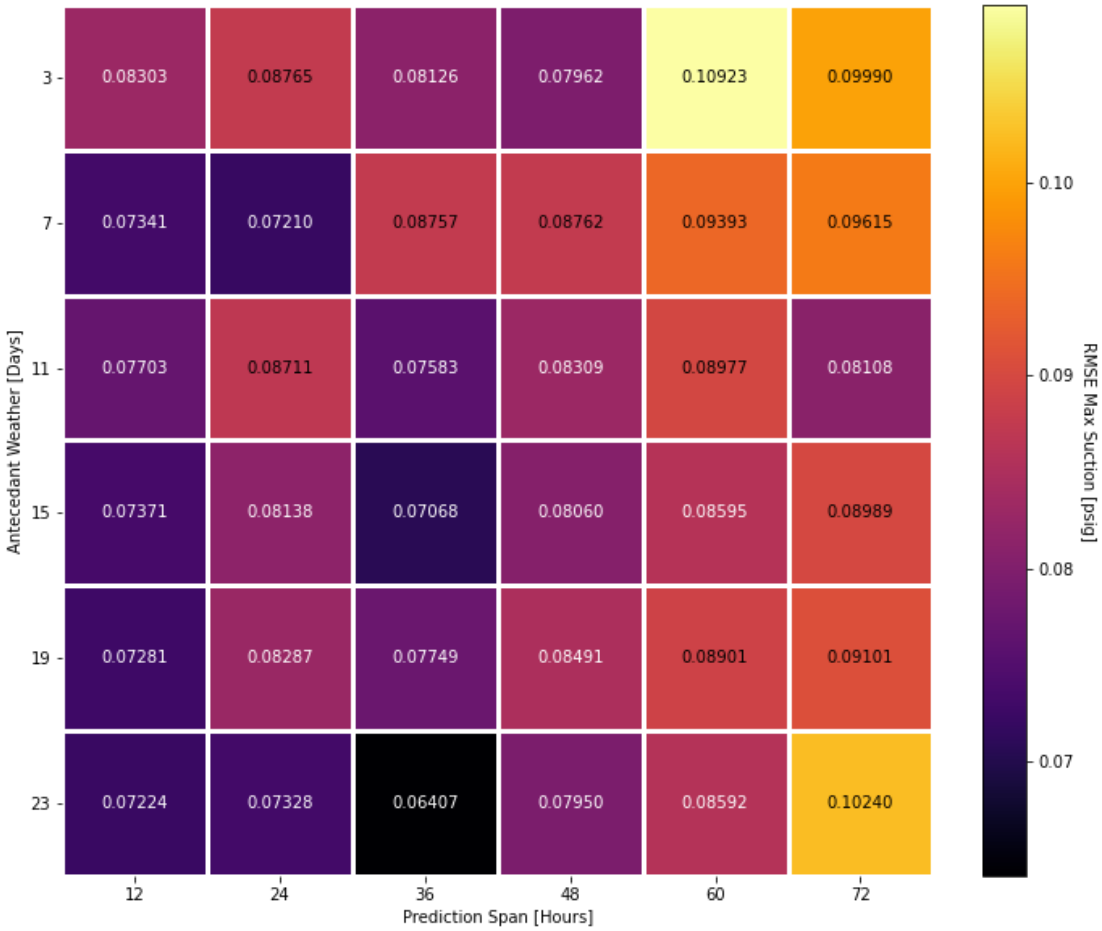

Figure 5. Parameter tuning heatmap of Antecedent Weather length and Prediction Span length. Parameter space for tuning included Antecedent Weather from three to 23 days into the past and Prediction Span from 12 to 72 hours into the future.

Notably, the ability to backcast antecedent pore pressure data is unavailable during historic landslide events used to inform the LEWS. Antecedent pore pressure conditions provide the model with an initial condition as well as additional data to use during training. If poor initial conditions are used (i.e., values are too far away or on another scale to target values), the gradient descent used in machine learning can fail to converge, as starting predictions can be too far away from target values. We mitigate this outcome by scaling our data to values between zero

and one, so that our initial conditions will be captured within the gradient descent, resulting in model convergence.

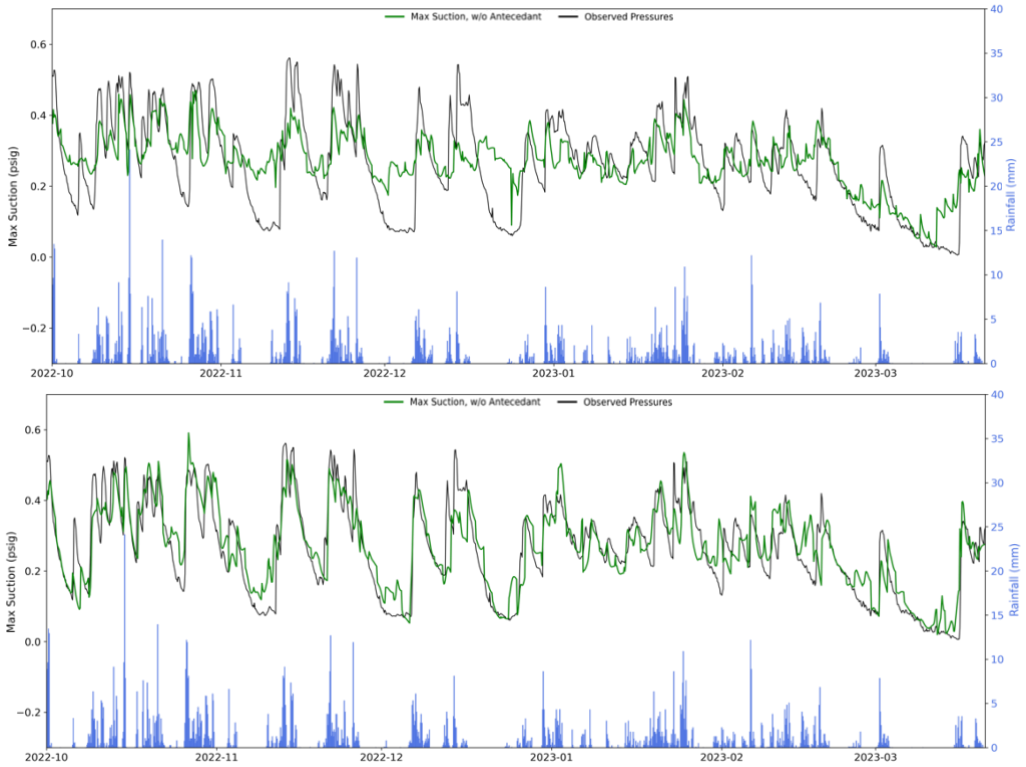

Figure 6. Comparison of the effect batch size has on model fit. Lower values of batch size (top, batch size=15) show decreased ability to correctly predict the full range of pore pressure values when compared to larger values of batch size (bottom, batch size=100).

To assess the performance of our model, we calculated the RMSE for pore pressure predictions at various rainfall intensity thresholds (Figure 7). The number of rainfall events captured with an intensity threshold of 2 mm/3hr is over 1000 events. As the rainfall intensity threshold increases, we see an nonlinear decrease of captured (or analyzed) events, specifically 26 events when for rainfall intensity of 14 mm/3hr. Because RMSE is a biased estimator, the larger number of rainfall events at lower thresholds may be a more encompassing estimate of error for rainfall events outside of the training dataset. For comparison, we perform the same

analysis for a model which includes antecedent soil hydrology conditions and observe that including antecedent soil hydrology conditions increases accuracy by up to 20%.

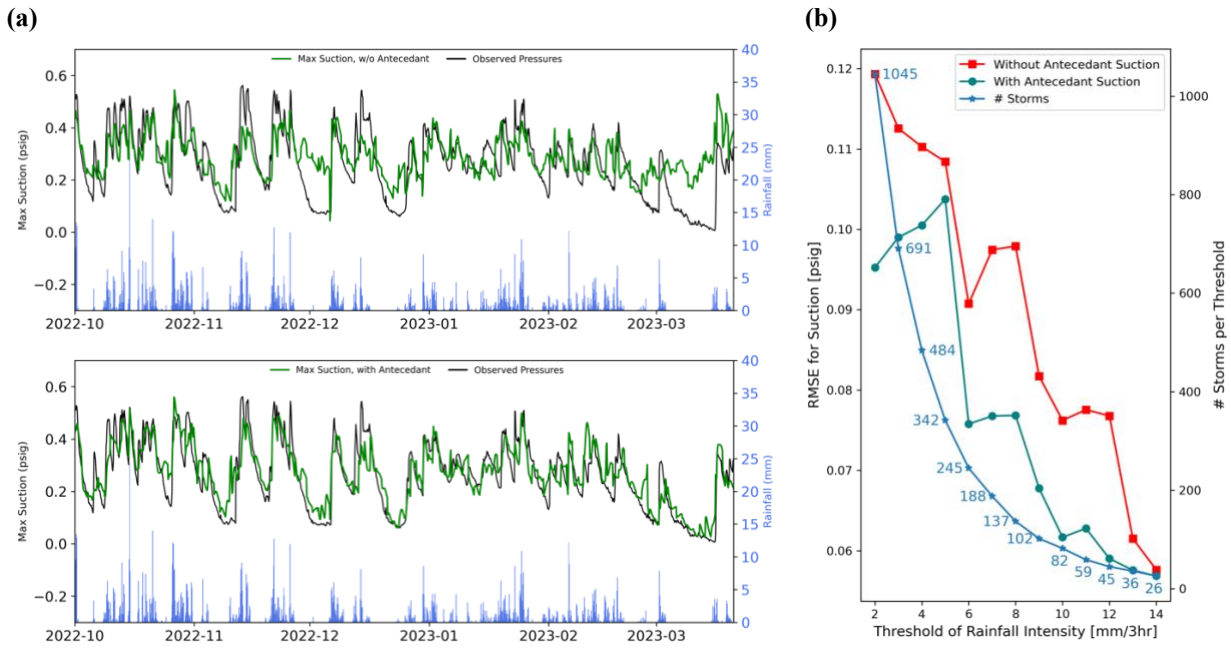

Figure 7. (a) Predicted and observed pore pressure response from October 2022 through March 2023 for a model run without including antecedent conditions during training (top) and a model run with antecedent pore pressure during training (bottom). (b) RMSE of both model type predictions during various rainfall intensity thresholds.

IV. DISCUSSION

The accuracy of a LEWS in slide-prone regions is of vital importance for the safety of the local population. The differences in local resident response to predicted landslide hazard present a challenge when implementing a LEWS and constraining that hazard prediction improves and contextualizes the amount of risk being modeled, allowing for knowledgeable decisions. As such, uncertainty in hazard prediction should be transparent and minimized. The accuracy of the logistical regression model being deployed in Sitka, AK could be improved by including an analysis of pore pressure responses for the corresponding landslides in their model inventory. Local pore pressure records do not exist for that time span but can be approximated with DL predictions.

4.1 Missing Initial Conditions for Modeling

As local pore pressure records do not exist for the period of time that spans past landslides that inform the LEWS, the DL model will not have access to them as initial conditions (antecedent conditions). Antecedent conditions are considered to be an important factor when considering hydrologic conditions and physical landslide initiation controls as most rainfall initiated shallow landslides occur during intense rainfall that follow periods of prolonged wetness (Zhao et al., 2019; Lazzari et al., 2018; Mirus et al., 2018b; Godt et al., 2006). Initial conditions within machine learning are important, as machine learning algorithms typically utilize a gradient descent approach to minimizing a loss function between predicted and observed values (Pathak et al., 2017; Wen et al., 2022) and a poor initial condition can cause the model to fail to converge. The problem is highlighted for systems, such as soil hydrology, where a small change in soil wetness and pore pressure initial conditions can differentiate whether soils have

enough shear strength to resist slope failure and landslide initiation. This gives rise to two limits in our model which has no initial conditions: (1) We limit our model accuracy by not providing initial conditions. (2) We are limited in the length of our prediction span based on the dynamic complexity of hydrology systems, which can evolve in varying ways depending on given initial conditions. Our results show that model accuracy become limited for predictions beyond 36 hours, which is comparable to similar models (Orland et al., 2020) with a slight decrease in accuracy and an increase in the required antecedent weather length.

4.2 Training Models for Intense Rainfall Event Predictions

Our model's accuracy assessment has shown that even without including antecedent soil hydrology conditions, it is still possible to predict pore pressure response during intense rainfall. We have focused on minimizing the RMSE of model predictions during these intense rainfall events. This is contrary to the loss equation used when training the model, which assesses the MSE (a less biased estimator) fit of every prediction equally. As the many landslides triggered by rainfall are caused by the buildup of water pore pressure into the ground during rainfall events after a period of prolonged wetness, we focus on the accuracy of our model predictions during intense rainfall, because accurate predictions during intense rainfall events lends itself directly to increasing accuracy for landslide hazard predictions (Mirus et al., 2018a; Bogaard et al., 2018; Fusco et al., 2019). The implications of focusing on the RMSE of select model predictions as opposed to the results of the overall model MSE fit do not, however, imply that overall model fit is low. It rather implies that every pore pressure response prediction that the model gives will not have equal importance in the context of landslide hazard prediction.

4.3 Model Parameters & Sensitivity

Throughout model training, we discovered several model hyperparameters that significantly impacted model performance. Length of antecedent weather and number of forward predictions made per timesteps combined controlled the quantity of data the model had access to during training, a crucial part of a machine learning model. Increasing the length of antecedent weather increased model performance by allowing the model access to a longer duration of antecedent conditions, which is a good proxy for the antecedent wetness (Zhao et al., 2019). Antecedent rainfall is important to hillslope conditions and landslide initiation probability, but the critical threshold for how much is important has been contended in various papers, ranging from several hours to several weeks and likely varies by study area (Segoni, 2018). We found that, in the absence of antecedent soil moisture data, increasing the amount of antecedent weather conditions from 36 hours (Orland et al., 2020) to above two weeks proved to provide comparable results for an increase in computational resources. For the prediction span (number of forward predictions made from a given timestep), decreasing the value provided more accuracy at the cost of increased computational resources. Increases this value past 36 hours generated inaccurate predictions (Figure 5). We found that predicting 24-36 hours forward based on the previous 2-3 weeks of antecedent weather provided a balance of accuracy and computation costs, with lower values of prediction span and higher values of antecedent weather gaining diminishing amounts of accuracy for increasing model runtime.

The last model hyperparameter, batch size (number of predictions made per model epoch), was found to have an optimal value of 100, which proved to balance model computational performance with model prediction accuracy. We found that predictions made

from models training with lower values tended to show a reduced capacity to capture the smallest and largest of the observed pore pressure record. Larger values quickly showed improved fit with a disparate increase in computational costs. We found values on the order of 100 to provide accurate results without long model runtimes.

V. CONCLUSIONS

We demonstrate that weather records separated by 4.3 kilometers and 575 meters in elevation in a region described by pronounced spatial heterogeneity in precipitation are well correlated and can serve as a proxy for hydrologic response on the local scale. Our model performs well even without considering important antecedent soil hydrology conditions, with a marginal decrease in accuracy. Excluding antecedent soil hydrology conditions in a LSTM model with limited training data can therefore be mitigated by expanding the duration of the antecedent weather data input to stand as a proxy for soil moisture conditions. Thus, the limited quantity of training data on the local scale serves as a first-order limit on the model's predictive capability. Further, model accuracy bias during intense storm events is related to number of storms at each intensity threshold, with more storms providing a less biased estimate of error. With previous work exemplifying the capabilities of such a LSTM DL model to learn and understand the physical hydrologic processes in landslide-prone hillslopes, we propose that this model has the capabilities to predict and backcast hydrologic response to rainfall in data-limited conditions and environments.

APPENDIX: MODEL CODE

```
%load_ext autoreload
%autoreload 2

#everything we will import to use later
import os
os.environ["TF_CPP_MIN_LOG_LEVEL"]="3"
import warnings
warnings.filterwarnings('ignore')
warnings.simplefilter('ignore')
import numpy as np
import pandas as pd
from datetime import datetime, date, timedelta
from matplotlib import pyplot as plt
from tensorflow.keras.models import model_from_json
import tensorflow_addons as tfa
from functions import *

%matplotlib inline

import tensorflow.compat.v1 as tf
from tensorflow import keras
from tensorflow.keras import backend as K
from tensorflow.keras import Model
from tensorflow.keras.layers import *
from tensorflow.keras.wrappers.scikit_learn import KerasRegressor
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import tqdm
from tqdm.keras import TqdmCallback

## OPTIONS FOR SCRIPT

# Dataset location
DATA_FILEPATH =
'./sitka_data/processed_for_training_3_hour_72_1_no_vwc_max_pp_only.csv'

# these two variables are in units of your data. Current runs have been using
3 hour data, so N_TARG is really 36 hours forward
N_TRAIN = 21*8  #how long should the antecedant window be?
N_TARG = 12  #how long should forward predictiono span be?

# standard ML value
TEST_VALIDATION_SPLIT = 0.7

# One of these must be true, at least
PP_TARGETING = True
VWC_TARGETING = False
```

```
ANTECEDANT_SOIL_HYDROLOGY = False

# if this is true, you generally wouldn't want OUTPUT to be true, as you'd
just be writing out the same data again
LOAD_POSTPROCESS_FEATURES_AND_TARGETS = False
LOAD_FEATURES_FILENAME = ''
LOAD_TARGETS_FILENAME = ''
LOAD_TARGET_INDICES_FILENAME = ''

# if this is true, you generally don't want LOAD to have been true above, as
you'd just be writing out the same data again
OUTPUT_POSTPROCESS_FEATURES_AND_TARGETS = False
OUTPUT_FEATURES_FILENAME = ''
OUTPUT_TARGETS_FILENAME = ''
OUTPUT_TARGET_INDICES_FILENAME = ''

EPOCHS = 2000
BATCH_SIZE = 150

BACKASTING = False

ocr = pd.read_csv(DATA_FILEPATH, index_col=0)

# Search data headers for strings related to soil hydrology
pp_cols = []
VWC_cols = []
if PP_TARGETING:
    pp_cols = [v for v in ocr.columns if "Pressure_" in v]  # measures number
of features for pp
if VWC_TARGETING:
    VWC_cols = [v for v in ocr.columns if "vwc_" in v]  # measures number of
features for VWC

num_targets = len(pp_cols) + len(VWC_cols)
ocr.index = pd.to_datetime(ocr.index)

ntrain = N_TRAIN  #how long should the antecedant window be?
ntarg = N_TARG  #how long should forward predictiono span be?

if not ANTECEDANT_SOIL_HYDROLOGY:
    # scale data, no antecedant pp
    data = ocr
    data_scaling_features =
preprocessing.MinMaxScaler(feature_range=(0,1)).fit(data.iloc[:,:-
num_targets])
    data_scaled_df =
pd.DataFrame(data_scaling_features.transform(data.iloc[:,:-num_targets]),
index=ocr.index)
    data_scaling_targets =
preprocessing.MinMaxScaler(feature_range=(0,1)).fit(data.iloc[:,-
num_targets:])
```

```
    data_scaled_df = pd.concat([data_scaled_df,
(pd.DataFrame(data_scaling_targets.transform(data.iloc[:,-num_targets:]),
index=ocr.index))], axis = 1)
    data_scaled_df.fillna(-1, inplace=True)

    if LOAD_POSTPROCESS_FEATURES_AND_TARGETS:
        features = np.load(LOAD_FEATURES_FILENAME)
        targets = np.load(LOAD_TARGETS_FILENAME)
        target_indices = np.load(LOAD_TARGET_INDICES_FILENAME)

    else:
        # prepare features and targets
        features, targets, target_indices =
lstm_prep_no_pp(data_scaled_df.index.values, data_scaled_df.values[:,:-
num_targets],

data_scaled_df.values[:,-num_targets:], num_targets, ntrain, ntarg)

    # create bounds for the prediction intervals
    intervals = np.zeros(len(target_indices))
    for i in range(0, len(intervals), targets.shape[1]):
        intervals[i]=1

    intervals[intervals==0] = np.nan

    binary_indices = np.copy(target_indices)

    for i in range(0, len(binary_indices), forecast_hrs):
        binary_indices[i] = np.datetime64("NaT")

    # set -1s to nan to be ignored
    targets[targets==-1] = np.nan

if ANTECEDANT_SOIL_HYDROLOGY:
    # scale data, w/ antecedant pp
    data = ocr
    data_scaling =
preprocessing.MinMaxScaler(feature_range=(0,1)).fit(data.iloc[:,:])
    data_scaled =
pd.DataFrame(data_scaling.transform(data.iloc[:,:]),index=ocr.index)
    data_scaled_df = data_scaled.fillna(-1)

    if LOAD_POSTPROCESS_FEATURES_AND_TARGETS:
        features = np.load(LOAD_FEATURES_FILENAME)
        targets = np.load(LOAD_TARGETS_FILENAME)
        target_indices = np.load(LOAD_TARGET_INDICES_FILENAME)

    else:
        # prepare features and targets
        features, targets, target_indices =
lstm_prep_w_pp(data_scaled_df.index.values, data_scaled_df.values,
num_targets, ntrain, ntarg)
```

```
    # create bounds for the prediction intervals
    intervals = np.zeros(len(target_indices))
    for i in range(0,len(intervals),targets.shape[1]):
        intervals[i]=1

    intervals[intervals==0] = np.nan

    binary_indices = np.copy(target_indices)

    for i in range(0,len(binary_indices),36):
        binary_indices[i]=np.datetime64("NaT")

    # set -1s to nan to be ignored
    targets[targets==-1] = np.nan

if OUTPUT_POSTPROCESS_FEATURES_AND_TARGETS:
        np.save(OUTPUT_FEATURES_FILENAME, features)
        np.save(OUTPUT_TARGETS_FILENAME, targets)
        np.save(OUTPUT_TARGET_INDICES_FILENAME, target_indices)

# split training/testing data, and separate train/test data from 20 years of
backcasting prediction data.
index_usgs_gap_start = ocr.index.get_loc("2020-06-09 21:00:00")
split = TEST_VALIDATION_SPLIT
pp_index = int(np.ceil(index_usgs_gap_start/ntarg))

train_split_index = int(features.shape[0]-pp_index)
test_split_index = int(train_split_index*(1-split))

train_features = features[pp_index:-test_split_index]
test_features = features[-test_split_index:]

predict_features = features[:int(pp_index+1)]  #use these predict array when
backcasting

train_targets = targets[pp_index:-test_split_index]
test_targets = targets[-test_split_index:]
test_indices = target_indices[-test_targets.shape[0]*test_targets.shape[1]:]

predict_targets = targets[:int(pp_index+1)]
predict_indices =
target_indices[:predict_targets.shape[0]*predict_targets.shape[1]]

# The below model is inspired by and adapted from:
https://github.com/LukeTonin/keras-seq-2-seq-signal-prediction
# Define an input shape. This is the tensor shape our model expects from now
on.
# This is essential for a stateful model
batch = None

# this doesn't seem to have a huge effect on the model
```

```
#n_units = 36
n_units = 72
#n_units = features.shape[2]

encoder_inputs = Input(batch_input_shape=(batch, train_features.shape[1],
train_features.shape[2]), name="encoder_input")

encoder_lstm = LSTM(n_units, return_state=True, stateful=False,
return_sequences=True, name="encoder") # define encoder
# connect encoding layer to our inputs, return all states

encoder_outputs, state_h, state_c = encoder_lstm(encoder_inputs)
encoder_states = [state_h, state_c]

# Define inputs to the decoder.
decoder_inputs = Input(batch_input_shape=(batch, None,
train_targets.shape[2]), name="decoder_input")

# Create Decoder...
decoder_lstm = LSTM(n_units, return_state=True, return_sequences=True,
stateful=False, name="decoder")

# Important step: connect Decoder to our input layers and use the hidden and
cell states
# from the encoder to instantiate this layer
#decoder_outputs, decoder_h, decoder_c = decoder_lstm(decoder_inputs,
initial_state=[state_h, state_c])
decoder_outputs, decoder_h, decoder_c = decoder_lstm(decoder_inputs,
initial_state=encoder_states)
decoder_states = [decoder_h, decoder_c]

#create attention layer
# ----------------------
attention = dot([decoder_outputs, encoder_outputs], axes=[2, 2],
name="attention_dot")
attention2 = Activation('softmax')(attention)

context = dot([attention2, encoder_outputs], axes=[2,1], name="context_dot")
decoder_combined_context = concatenate([context, decoder_outputs])

# ----------------------

decoder_dense1 = Dense(50, activation='tanh', name="decoder_dense1")
dense_context = decoder_dense1(decoder_combined_context)

dropout = Dropout(0.5)
drop = dropout(dense_context)

decoder_dense2 = Dense(train_targets.shape[2], activation='linear',
name="decoder_context")
decoder_outputs = decoder_dense2(drop)
```

```
model = Model(inputs=[encoder_inputs, decoder_inputs],
outputs=decoder_outputs)
model.compile(optimizer='adam', loss=mse_nan, metrics=mse_nan)

# initialize tqdm callback with default parameters
tqdm_callback = tfa.callbacks.TQDMProgressBar()

# if a computer can fit every epoch in one go, set epochs to 1 (or any lower
number) and encase the fit command in a for loop.
# successive calls to the fit command will increasingly fit the model
model.fit([train_features, train_features[:,-ntarg:,-num_targets:]],
train_targets, epochs=EPOCHS, batch_size=BATCH_SIZE, verbose=0, shuffle=True,
callbacks=[TqdmCallback(verbose=0)])

print("Fitting complete! ")

#define inference ('inf') model, a separate encoding model. This just outputs
our encoder states
encoder_model = Model(encoder_inputs, [encoder_outputs, encoder_states])
inf_encoder_outputs, inf_encoder_states = encoder_model(encoder_inputs)

# set state shapes, which tells our decoder to accepts inputs states of the
specificed size
decoder_states_inputs = [Input(shape=(n_units,)), Input(shape=(n_units,))]

# create our decoding layer. accepts same shape as decoder inputs and encoder
states
inf_decoder_outputs, inf_state_h, inf_state_c = decoder_lstm(decoder_inputs,
initial_state=decoder_states_inputs)

# save decoder output states. We'll use these as the input states for our
decoder for predicting each next timestep
# after the initial input of our encoder states
inf_decoder_states = [inf_state_h, inf_state_c]

inf_attention = dot([inf_decoder_outputs, inf_encoder_outputs], axes=[2, 2])
inf_attention2 = Activation('softmax')(inf_attention)

inf_context = dot([inf_attention2, inf_encoder_outputs], axes=[2,1])
inf_decoder_combined_context = concatenate([inf_context,
inf_decoder_outputs])

inf_dense_context = decoder_dense1(inf_decoder_combined_context)
inf_drop = dropout(inf_dense_context)

inf_final_outputs = decoder_dense2(inf_drop)

# finally, instantiate our decoder model. Inputs are the original sequence +
the encoder states.
# outputs: sequence prediction + the states used for the decoder
decoder_model = Model([encoder_inputs, decoder_inputs]+decoder_states_inputs,
[inf_final_outputs]+inf_decoder_states)
```

```
if not ANTECEDANT_SOIL_HYDROLOGY:
    # no antecedent pp
    predictions = predict(test_features, encoder_model, decoder_model,
num_steps_to_predict=train_targets.shape[1],
                          num_features_to_predict=train_targets.shape[2],
batch_size=None)

    tests, preds = rescale_no_pp(test_features, test_targets, predictions,
data_scaling_features, data_scaling_targets, test_indices)

else:
    # w antecedent pp
    predictions = predict(test_features, encoder_model, decoder_model,
num_steps_to_predict=train_targets.shape[1],
                          num_features_to_predict=train_targets.shape[2],
batch_size=None)

    tests, preds = rescale_w_pp(test_features, test_targets, predictions,
data_scaling, test_indices)

if BACKCASTING:
    # backcast (w no antecedent pp)
    predictions = predict(predict_features, encoder_model, decoder_model,
num_steps_to_predict=train_targets.shape[1],
                          num_features_to_predict=train_targets.shape[2],
batch_size=None)

    tests_back, preds_back = rescale_no_pp(predict_features, predict_targets,
predictions, data_scaling_features, data_scaling_targets, predict_indices)

# score, scores = evaluate_forecasts(tests.values, preds.values)

# #evaluate model performance on RMSE of just hazardous pore pressure values
# print("\nLSTM Performance:")

# Pore Pressure Threshold
# y_pos, pred_pos, rmse, diff_array = threshold_rmse_eval(tests.values,
preds.values, -1.0)

# Rainfall Threshold
#rmse = threshold_rmse_eval_rain(tests, preds,
ocr["precip_accum_one_hour_mm"], 10.0)
#print(rmse)

#creating rmse array to save and plot
#rmse_thresh = []

#thresholds = np.arange(2,15,1)

#for i in thresholds:
```

```
#    rmse_thresh.append(threshold_rmse_eval_rain(tests, preds,
ocr["precip_accum_one_hour_mm"], i))

#print(rmse_thresh)

#np.save('./sitka_data/comparison_figs/with_antecedant_threshold_analysis.npy
', rmse_thresh)

# import matplotlib.lines as mlines

# ocr.index = ocr.index.tz_localize(None)

# # start_1 = pd.to_datetime('2015-08-01')
# # end_1 = pd.to_datetime('2015-09-01')

# start_1 = pd.to_datetime('2022-05-30')
# end_1 = pd.to_datetime('2022-10-01')

# start_3 = pd.to_datetime('2022-10-01')
# end_3 = pd.to_datetime('2023-03-21')

# #messy plot of rainfall and test data/predictions. I've been messing with
the y axis range to help clean it up

# #col_list = VWC_cols + pp_cols
# col_list = pp_cols

# fig, (ax1, ax3) = plt.subplots(2, 1, figsize=(15,10), sharex=False,
constrained_layout=True)
# #fig, ax3 = plt.subplots(figsize=(10,4), sharex=False)
# #fig.suptitle("Pore Pressure Predictions without Antecedant")

# # ax1.scatter(tests.index, tests.iloc[:, 1], c='k', linewidth=1,
label='Observed Data')
# # ax1.scatter(preds.index, preds.iloc[:, 1], c='green',
label='Predictions_1')

# ax1.plot(tests.iloc[:, 0], c='k', linewidth=1, label='Observed Data')
# ax1.plot(preds.iloc[:, 0], c='green', label='Predictions_1')

# ax3.plot(tests.iloc[:, 0], c='k', linewidth=1, label='Observed Data')
# ax3.plot(preds.iloc[:, 0], c='green', label='Predictions_1')

# #ax1.scatter(target_indices, intervals, c='k', marker='|', s=50, alpha=0.5)

# ax1.set_xlim(start_1, end_1)
# ax3.set_xlim(start_3, end_3)

# ax1.set_ylabel('Max Suction (psig)')
# ax1.set_ylim(-0.3, 0.7)
# ax1.xaxis.set_visible(True)
```

```
# ax3.set_ylabel('Max Suction (psig)')
# ax3.set_ylim(-0.3, 0.7)
# ax3.xaxis.set_visible(True)

# #plt.xticks(rotation=45)

# ax1.tick_params(axis='both', labelcolor='k', labelsize=13)
# ax3.tick_params(axis='both', labelcolor='k', labelsize=13)

# ax2 = ax1.twinx()  # instantiate a second axes that shares the same x-axis
# ax4 = ax3.twinx()  # instantiate a second axes that shares the same x-axis

# ax1.patch.set_visible(False)
# ax1.set_zorder(ax2.get_zorder() + 1)

# ax3.patch.set_visible(False)
# ax3.set_zorder(ax4.get_zorder() + 1)

# ax2.set_ylabel('Rainfall (mm)', color='royalblue')  # we already handled
the x-label with ax1
# ax2.bar(preds.index, ocr.loc[preds.index, ocr.columns[0]],
color='royalblue', alpha=0.8,width=0.2)
# ax2.tick_params(axis='y', labelcolor='royalblue', labelsize=13)

# ax2.set_ylim(0,40)

# ax4.set_ylabel('Rainfall (mm)', color='royalblue')  # we already handled
the x-label with ax1
# ax4.bar(preds.index, ocr.loc[preds.index, ocr.columns[0]],
color='royalblue', alpha=0.8,width=0.2)
# ax4.tick_params(axis='y', labelcolor='royalblue', labelsize=13)

# ax4.set_ylim(0,40)

# one_line = mlines.Line2D([], [], color='green', marker='',
#                  markersize=15, label='Max Suction, without Antecedant')
# # two_line = mlines.Line2D([], [], color='blue', marker='',
# #                   markersize=15, label='2 Hour')


# black_line = mlines.Line2D([], [], color='k', marker='',
#                  markersize=15, label='Observed Pressures')

# ax1.legend(handles=[one_line, black_line], loc='upper center',
#         fontsize='large', bbox_to_anchor=(0.52,1.01), ncol=4,
#         markerscale=1.3, frameon=False)


# ax3.legend(handles=[one_line, black_line], loc='upper center',
#         fontsize='large', bbox_to_anchor=(0.52,1.01), ncol=4,
#         markerscale=1.3, frameon=False)
# # plt.legend(handles=[one_line_hr, one_line_ft, black_line], loc='upper
center',
```

```
# #         fontsize='small', bbox_to_anchor=(0.52,1.01), ncol=4,
# #         markerscale=1.3, frameon=False)

# #plt.title("Max PP, 72 Hr Forecast, 3 Input")

# # plt.setp(ax1.get_xticklabels(True, "major"), visible=True)
# # plt.setp(ax3.get_xticklabels(True, "major"), visible=True)

# plt.rc('axes', labelsize=15)
# #plt.rc('legend', fontsize=50)
# #plt.tight_layout()

#
#plt.savefig('./sitka_data/comparison_figs/redo2_without_antecedant_batch_siz
e_150_epoch_3000.png', dpi=300)
# plt.show()

# wo_ante_rmse =
np.load("./sitka_data/comparison_figs/no_antecedant_threshold_analysis.txt.np
y")
# w_ante_rmse =
np.load("./sitka_data/comparison_figs/with_antecedant_threshold_analysis.npy"
)

# n_storms_per_thresh = [1045, 691, 484, 342, 245, 188, 137, 102, 82, 59, 45,
36, 26]

# fig, ax = plt.subplots(figsize=(5,7), constrained_layout=True)

# ax.plot(thresholds, wo_ante_rmse, color='red', marker='s', label="Without
Antecedant Suction")
# ax.plot(thresholds, w_ante_rmse, color='teal', marker='o', label='With
Antecedant Suction')
# ax.set(xlabel='Threshold of Rainfall Intensity [mm/3hr]', ylabel='RMSE for
Suction [psig]')
# ax2 = ax.twinx()

# ax2.plot(thresholds, n_storms_per_thresh, marker='*', label="# Storms")
# ax2.set(ylabel='# Storms per Threshold')

# plt.rc('axes', labelsize=20)

# for index in range(len(thresholds[:3])):
#     ax2.text(thresholds[index]+.3, n_storms_per_thresh[index]-10,
n_storms_per_thresh[index], size=12, color='#1f77b4')

# for index in range(len(thresholds[3:8])):
#     ax2.text(thresholds[3+index]-1.4, n_storms_per_thresh[3+index]-10,
n_storms_per_thresh[3+index], size=12, color='#1f77b4')

# for index in range(len(thresholds[8:])):
```

```
#     ax2.text(thresholds[8+index]-.5, n_storms_per_thresh[8+index]-35,
n_storms_per_thresh[8+index], size=12, color='#1f77b4')

# fig.legend(loc="upper right", bbox_to_anchor=(1,1),
bbox_transform=ax.transAxes)
#
plt.savefig('./sitka_data/comparison_figs/RMSE_comparison_w_and_wo_antecedant
_2.png', dpi=300)

# from sklearn.metrics import r2_score
# from scipy.stats import linregress

# col_list = pp_cols
# fig, ax = plt.subplots(3, 1, figsize=(6,8), sharex=False)
# i = 1
# for col in col_list:
#     while i < (len(col_list)+1):
#         x = np.linspace(np.nanmin(tests.iloc[:,i-1]),
np.nanmax(tests.iloc[:,i-1]),100)
#         y = x
#         ax1 = ax[i-1]
#         ax1.plot(x, y, c='k')
#         ax1.scatter(tests.iloc[:,i-1], preds.iloc[:,i-1], c='slategrey',
s=15, alpha=0.4)

#         r2 = r2_score(tests.iloc[:,i-1], preds.iloc[:,i-1])
#         print(linregress(tests.iloc[:,i-1], preds.iloc[:,i-1]))
#         slope, intercept, r_value, p_value, std_err =
linregress(tests.iloc[:,i-1],
#
preds.iloc[:,i-1])
#         print(slope)
#         ax1.annotate(('R =
'+'{0:.3f}'.format((r_value))),(0.75,0.2),None,'axes fraction')
#         #r_values_full[pits_index, i-1] = r_value
#         #ax1.annotate(('Slope =
'+'{0:.3f}'.format((slope))),(0.10,0.72),None,'axes fraction')
#         #if i == 1:
#             #ax1.set_title('Soil Matric Potential - Observed vs. Predicted
Comparison')
#         ax1.set_ylabel('Predicted (kPa)')
#         ax1.set_xlabel('Observed Value (kPa)')

#         i = i+1

# #pits_index += 1

# plt.tight_layout()
# #plt.savefig('36hr_36_24_d75dr50_cor_plot_SMALLER.svg',dpi=300)
```

REFERENCES CITED

Bellugi, D., Milledge, D. G., Dietrich, W. E., Perron, J. T., & McKean, J. (2015). Predicting shallow landslide size and location across a natural landscape: Application of a spectral clustering search algorithm. Journal of Geophysical Research: Earth Surface, 120(12), 2552–2585. https://doi.org/10.1002/2015JF003520

Bogaard, T., & Greco, R. (2018). Invited perspectives: Hydrological perspectives on precipitation intensity-duration thresholds for landslide initiation: Proposing hydro-meteorological thresholds. *Natural Hazards and Earth System Sciences*, *18*(1), 31–39. https://doi.org/10.5194/nhess-18-31-2018

Fusco, F., De Vita, P., Mirus, B. B., Baum, R. L., Allocca, V., Tufano, R., Di Clemente, E., & Calcaterra, D. (2019). Physically based estimation of rainfall thresholds triggering shallow landslides in volcanic slopes of Southern Italy. *Water*, *11*(9), 1915. https://doi.org/10.3390/w11091915

Godt, J. W., Baum, R. L., & Chleborad, A. F. (2006). Rainfall characteristics for shallow landsliding in Seattle, Washington, USA. *Earth Surface Processes and Landforms*, *31*(1), 97–110. https://doi.org/10.1002/esp.1237

Guzzetti, F., Peruccacci, S., Rossi, M., & Stark, C. P. (2007). The rainfall intensity–duration control of shallow landslides and debris flows: An update. *Landslides*, *5*(1), 3–17. https://doi.org/10.1007/s10346-007-0112-1

Hewamalage, H., Bergmeir, C., & Bandara, K. (2021). Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, *37*(1), 388–427. https://doi.org/10.1016/j.ijforecast.2020.06.008

Klose, M., Maurischat, P., & Damm, B. (2015). Landslide impacts in Germany: A historical and socioeconomic perspective. *Landslides*, *13*(1), 183–199. https://doi.org/10.1007/s10346-015-0643-9

Lacasse, S., & Nadim, F., Chen, W.-F., Duan, L. (2014). Landslide risk assessment and mitigation strategy. In *Superstructure design: Bridge engineering handbook* (pp. 45–61). essay, CRC Press.

Lacasse S, Nadim F, Kalsnes B (2010) Living with landslide risk. *Geotech Eng J SEAGS AGSSEA* 41:1–13

Lara-Benítez, P., Carranza-García, M., & Riquelme, J. C. (2021). An experimental review on Deep Learning Architectures for time series forecasting. *International Journal of Neural Systems*, *31*(03), 2130001. https://doi.org/10.1142/s0129065721300011

Lazzari, M., Piccarreta, M., & Manfreda, S. (2018). The role of antecedent soil moisture conditions on rainfall-triggeredshallow landslides. *Natural Hazards and Earth System Sciences Discussions*. https://doi.org/10.5194/nhess-2018-371

Mirus, B., Smith, J., Godt, J., Baum, R., & Coe, J. (2016). Simulated effect of topography and soil properties on hydrologic response and landslide potential under variable rainfall conditions in the Oregon Coast Range, USA. *Landslides and Engineered Slopes. Experience, Theory and Practice*, 1431–1439. https://doi.org/10.1201/b21520-176

Mirus, B.B, Morphew, M., & Smith, J. (2018a). Developing Hydro-Meteorological Thresholds for Shallow Landslide Initiation and Early Warning. *Water*, 10(9), 1274. https://doi.org/10.3390/w10091274

Mirus, B. B., Becker, R. E., Baum, R. L., & Smith, J. B. (2018b). Integrating real-time subsurface hydrologic monitoring with empirical rainfall thresholds to improve landslide early warning. *Landslides*, *15*(10), 1909–1919. https://doi.org/10.1007/s10346-018-0995-z

Montgomery, D. R., & Dietrich, W. E. (1994). A physically based model for the topographic control on shallow landsliding. *Water Resources Research*, 30(4), 1153–1171. https://doi.org/10.1029/93WR02979

National Oceanic and Atmospheric Administration (NOAA) National Centers for Environmental Information (NCEI): Global Surface Hourly Precipitation for Sitka Airport station PASI. NOAA National Centers for Environmental Information, Retrieved from https://www.ncei.noaa.gov/access/search/data-search/global-hourly, 2001.

Orland, E., Roering, J. J., Thomas, M. A., & Mirus, B. B. (2020). Deep learning as a tool to forecast hydrologic response for landslide-prone hillslopes. *Geophysical Research Letters*, *47*(16). https://doi.org/10.1029/2020gl088731

Papacharalampous, G., Tyralis, H., & Koutsoyiannis, D. (2019). Comparison of stochastic and machine learning methods for multi-step ahead forecasting of Hydrological Processes. *Stochastic Environmental Research and Risk Assessment*, *33*(2), 481–514. https://doi.org/10.1007/s00477-018-1638-6

Pathak, J., Lu, Z., Hunt, B. R., Girvan, M., & Ott, E. (2017). Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *27*(12), 121102. https://doi.org/10.1063/1.5010300

Patton, A. I., Luna, L. V., Roering, J. J., Jacobs, A., Korup, O., & Mirus, B. B. (2023). Landslide initiation thresholds in data sparse regions: Application to landslide early warning criteria in Sitka, Alaska, USA. *EGUSphere*. https://doi.org/10.5194/egusphere-2023-25

Segoni, S., Piciullo, L., & Gariano, S. L. (2018). A review of the recent literature on rainfall thresholds for landslide occurrence. *Landslides*, *15*(8), 1483–1501. https://doi.org/10.1007/s10346-018-0966-4

Sharma, A. R., & Déry, S. J. (2020). Contribution of atmospheric rivers to annual, seasonal, and extreme precipitation across British Columbia and southeastern Alaska. *Journal of Geophysical Research: Atmospheres*, *125*(9). https://doi.org/10.1029/2019jd031823

Sim, K. B., Lee, M. L., & Wong, S. Y. (2022). A review of landslide acceptable risk and tolerable risk. *Geoenvironmental Disasters*, *9*(1). https://doi.org/10.1186/s40677-022-00205-6

Shen, C., Laloy, E., Elshorbagy, A., Albert, A., Bales, J., Chang, F.-J., Ganguly, S., Hsu, K.-L., Kifer, D., Fang, Z., Fang, K., Li, D., Li, X., & Tsai, W.-P. (2018). Hess opinions: Incubating deep-learning-powered hydrologic science advances as a community. *Hydrology and Earth System Sciences*, *22*(11), 5639–5656. https://doi.org/10.5194/hess-22-5639-2018

Simunek, J., Genuchten, M. Van, & Sejna, M. (2005). The HYDRUS-1D software package for simulating the one-dimensional movement of water, heat, and multiple solutes in variably-saturated media. HYDRUS Software Ser.

Thomas, M. A., Mirus, B. B., & Collins, B. D. (2018). Identifying Physics-Based Thresholds for Rainfall-Induced Landsliding. *Geophysical Research Letters*, 45(18), 9651–9661. https://doi.org/10.1029/2018GL079662

van Natijne, A. L., Lindenbergh, R. C., & Bogaard, T. A. (2020). Machine Learning: New Potential for Local and Regional Deep-Seated Landslide Nowcasting. *Sensors*, 20(5), 1425. https://doi.org/10.3390/s20051425

Wen Y, Chaolu T, Wang X (2022) Solving the initial value problem of ordinary differential equations by Lie group based neural network method. *PLoS ONE* 17(4): e0265992. https://doi.org/10.1371/journal.pone.0265992

Wendler, G., Galloway, K., & Stuefer, M. (2015). On the climate and climate change of Sitka, Southeast Alaska. *Theoretical and Applied Climatology*, *126*(1–2), 27–34. https://doi.org/10.1007/s00704-015-1542-7

Wieczorek, G., & Guzzetti, F. (2000). A review of rainfall thresholds for triggering landslides. Mediterranean Storms, Proceedings of the EGS Plinius Conference '99, 8(January), 404–414. Retrieved from http://www.idrologia.polito.it/~claps/pliniusonline/pdf_proceedings/Plinius/Wieczorek/WIECZOREK.pdf

Zhao, B., Dai, Q., Han, D., Dai, H., Mao, J., & Zhuo, L. (2019). Antecedent wetness and rainfall information in landslide threshold definition. *Hydrology and Earth System Sciences Discussion*. https://doi.org/10.5194/hess-2019-150