Deep Learning Approaches for Efficiently Emulating Noah Land Surface Model Soil Hydrology

Master's Thesis Proposal
Mitchell Dodson

1 Introduction and Hypothesis

Accurate characterization of land surface states including profiles of soil moisture and temperature are fundamental to numerical weather prediction, crop health and hydrologic monitoring, and for situational awareness during drought and flood events. To that end, the Noah Land Surface Model (Noah-LSM) has become a fixture in the numerical modeling and operational forecasting communities, serving as the land component of the Weather Research and Forecasting (WRF) community model, NOAA's Global Forecast System (GFS) and Climate Forecast System (CFS) models, and as a near real-time product utilized by the National Weather Service and US Drought Monitor [1][2][3].

High spatial resolution and the ability to generate land surface states from ensembles of atmospheric forcings are high priorities for forecasting and reanalysis frameworks including the next-generation North American Land Data Assimilation System (NLDAS) [4][5]. Like many dynamical models, evaluating Noah-LSM over large spatial or temporal domains is computationally costly since it relies on recurrent numerical integration techniques which can be difficult to parallelize. Deep learning techniques provide the means to approximate a highly nonlinear and heavily parameterized system like Noah-LSM by inferring the relationship between input and output values in a set of training samples, and encoding their dynamics as a sequence of composed affine matrix operations followed by nonlinearities, which are computationally efficient on modern GPU hardware [6][7]. As such, this proposal hypothesizes that a deep learning model trained on atmospheric forcings and model outputs from NLDAS-2 [1] can simulate the soil hydrodynamics of the Noah model with low computational cost and fidelity sufficient enough to provide useful estimates.

2 Background

2.1 The History of Noah-LSM

The theoretical framework underpinning Noah-LSM was initially formulated in the 1980s as part of the OSU model, which characterizes boundary layer moisture and energy fluxes as a 2-layer soil model subject to atmospheric forcings. The model expresses the infiltration and movement of water between the soil layers with the diffusive form of the Richards equation [8], direct evaporation using an analytic approximation of the Penman-Montieth relation in terms of atmospheric stability [9], and basic plant transpiration in terms of vegetation density and soil water content [10]. These features form an interdependent system of differential equations that are numerically integrated using a combination of the Crank-Nicholson method and finite-differencing [11], which introduces the need for short time steps of 15 or 30 minutes in order for the system to remain numerically stable [12][8].

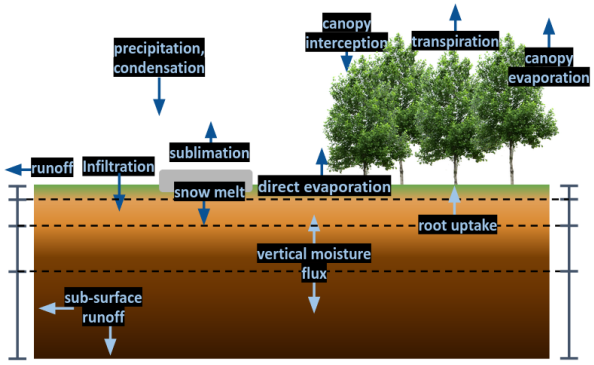

Figure 1: Schematic diagram of the feedbacks contributing to the evolution of volumetric soil moisture in Noah-LSM

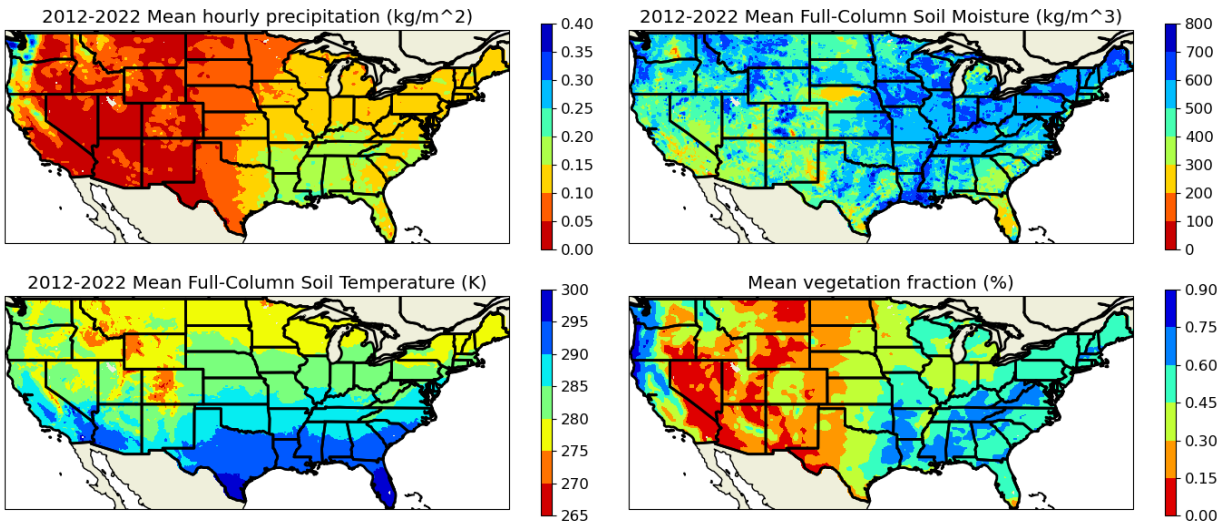

Figure 2: NLDAS-2 mean values over study period. Clockwise: hourly precipitation (either phase), full-column (2m) soil moisture, vegetation fraction, and full-column soil temperature.

The OSU model was later significantly improved, renamed to the first generation of Noah-LSM, and coupled with the NCEP Eta forecast model. Noah-LSM expanded the domain to four soil layers of increasing thicknesses 10cm, 30cm, 60cm, and 100cm, improved runoff dynamics by implementing Philip's equation for infiltration capacity [13], and represented influence of soil texture on moisture transport by introducing bounds on bare-soil potential evaporation that are determined by the soil composition [14][15]. The model also features a significantly enhanced representation of vegetation including a more thorough treatment of canopy resistance via a "Jarvis-type" model of leaf stomatal control [16][17], which accounts for the dependence of transpiration on insolation, air temperature and dewpoint, soil moisture capacity, and vegetation density. The vegetation effects are scaled by a monthly climatology of normalized difference vegetation index (NDVI) values observed by the NOAA-AVHRR satellite radiometer, which serve as a proxy for green vegetation fraction (GVF) [18][19], and the depth of root water uptake associated with plant transpiration is determined by a pixel's vegetation class as specified by the Simple Biosphere Model [20]. Finally, the model's utility was greatly expanded with the addition of a frozen soil and snow pack parameterization incorporating the thermal and hydraulic properties of fractionally-frozen soil layers, the effects of state changes [19][21], radiative feedbacks from partial snowpack coverage, and a snow density scheme [22].

2.2 NLDAS and Noah-LSM

Soon after the turn of the millennium, the first generation of NLDAS was under development as part of a multi-institution collaborative effort sponsored by the Global Energy and Water Cycle Experiment (GEWEX) Continental-scale International Projects (GCIP) team. The goal of the project was to incorporate long-term observations of land surface temperature, snow pack depth, and meteorological forcings from multiple sources (in-situ, satellite, radar) into a common framework used to independently evaluate land surface states and energy fluxes with four land surface models including Noah-LSM [23]. On a domain including the full conterminous United States (CONUS) at 0.125° resolution, the models were allowed to spin up over the course of a year, and soil states were recurrently used to initialize subsequent time steps rather than being "nudged" to correct for drift. Land cover and soil texture classification over the domain was derived by coarsening the University of Maryland and STATSGO datasets, respectively, from their native 1km resolutions [24], surface geometry and elevation is provided by the GTOPO30 dataset [25], and the parameter values for soil hydraulic properties were adapted from observations taken at the University of Virginia [26].

Forcing	Unit	Source	Δt	Δx
Temperature	K	NCEP fta/EDAS	3h	40km
Specific Humidity	kg kg^{-1}	NCEP Eta/EDAS	3h	40km
Wind Velocity	m s^{-1}	NCEP Eta/EDAS	3h	40km
Downward Longwave Flux	W m^{-2}	NCEP Eta/EDAS	3h	40km
Downward Shortwave Flux	W m^{-2}	UMD GOES-based insolation	1h	55km
Precipitation	kg m^{-2}	Gauge observations	24h	14km
		WSR-88D radar retrievals	1h	4km

Table 1: Atmospheric forcings provided by NLDAS at a 1-hourly resolution on the 0.125° CONUS grid. Data are resampled using spatial bilinear interpolation, then temporal disaggregation [23]. NLDAS forcing files also include values for CAPE, the ratio of convective precipitation, and potential evaporation (calculated as in [9]), but these three values won't be used as inputs to the models.

Attention remained on Noah-LSM in the following years as it continued to support NLDAS and other data assimilation and forecasting systems, which led to a series of improvements introduced alongside the next phase of the NLDAS project. A seasonal effect was added to vegetation by scaling the leaf area index (LAI) by GVF within bounds determined by the plant type, and transpiration was scaled by a root uptake efficiency factor determined by the proximity of soil temperature to an optimum growth temperature (298 K).

Several parameters were adjusted including the influence of vapor pressure deficit on transpiration rate, the minimum stomatal resistance for several plant species, and hydraulic parameters for some soil textures. The aerodynamic conductance coefficient – an important factor in the strength of moisture and energy fluxes from the surface – was increased during daylight hours, and a basic anisotropy model was introduced by modifying the albedo of some surfaces in terms of the solar zenith angle [27]. Snowpack physics were also modified to improve surface exchange coefficients, and to gradually diminish the snow albedo over the time since the last snowfall [28][29]. These changes introduce new feedbacks and involve sensitive parameters like LAI which have a strong influence on the model's dynamics [30].

The retrospective NLDAS-2 data record generated after applying these modifications extends back to 1979, and continues to be updated in a near real-time capacity [1]. Its forcings listed in Table 2 serve as the inputs to the neural networks, which are trained to predict the associated Noah land surface model states minimally including soil moisture and snow water equivalent.

2.3 Noah as a Dynamical Model

At its core, Noah-LSM is a collection of differential equations that express the time rate of change of land surface states $\frac{\partial \vec{s}}{\partial t}$ in terms of the current state $\vec{s}$, forcing $\vec{\psi}$, and parameters $\vec{\phi}$. Here, $\vec{s}$ consists of the model's dynamic state variables (snow pack depth, soil moisture and temperature, canopy storage), $\vec{\psi}$ encodes forcings from Table 2, and $\vec{\phi}$ includes static coefficients of the governing equations (ex. vegetation type/fraction, soil texture, elevation).

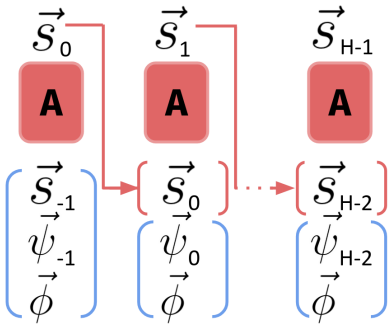

Figure 3: Diagram of a self-cycling discrete-time dynamical system with no hidden state. At each time, nonlinear operator A maps an initial state $\vec{s}_k$, exogenous forcing $\vec{\psi}_k$, and time-invariant parameters $\vec{\phi}$ to a new state $\vec{s}_{k+1}$, used to initialize the subsequent time step.

To generate a time series, Noah-LSM numerically integrates the system of equations using Euler and Crank-Nicholson techniques, which explicitly evaluate the differential equations at several time intervals in order to estimate the residual change in state while conserving the appropriate quantities. It is crucial that the time step remains small (15min for NLDAS-Noah) to mitigate truncation error from the assumption of local linearity [23][12].

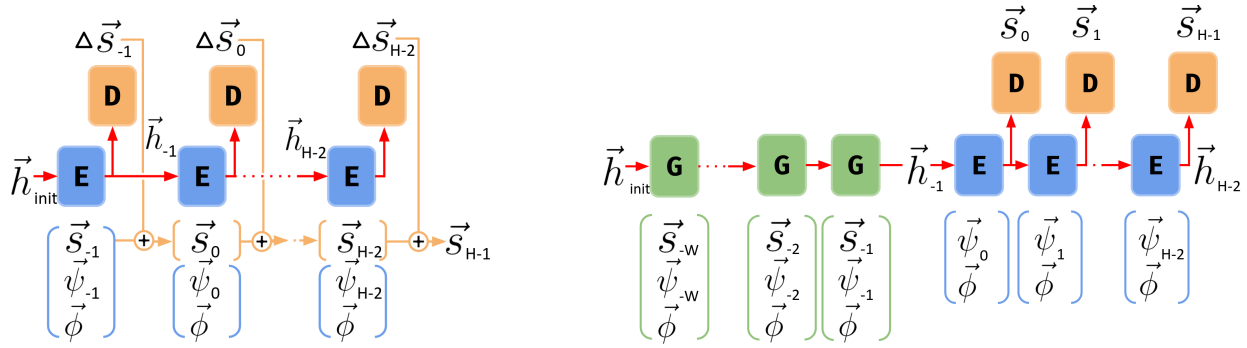

Figure 4: Information flow diagrams for a basic self-cycling RNN (left) and a sequence-to-sequence RNN (right). Both models have an encoder **E** that transforms current inputs and the previous hidden state into a new hidden state, then use a decoder **D** to generate outputs for H discrete time steps in the "forecast horizon." The sequence-to-sequence RNN features a separate encoder **G** that uses a series of prior values to initialize the hidden state used in prediction steps.

Figure 3 provides a general schematic that applies to Noah's evaluation process. Although no information is propagated between time steps by the model (except the output state), hysteretic effects emerge from the dynamics of **A** when it is numerically integrated. For example, the characteristic curves of unsaturated soil moisture diffusion depend on the timing of periods of wetting and dry-down since drainage and infiltration rates have a nonlinear relationship to the moisture content in the Richards equation [31].

2.4 Deep Learning Approach

As discussed in the previous section, the goal of numerical integration is to approximate the continuous evolution of a dynamical system by applying an explicit and highly nonlinear equation over small discrete time steps. Neural networks have effectively the opposite approach, in that they use many sample time series (generated by the model) to learn the governing equations implicit to the samples' dynamics.

Deep neural networks are a reasonable strategy for creating a reduced-order simulation of Noah-LSM's dynamics because are universal function approximators, which means that given a sufficient number of learnable weights, a feedforward network with at least one hidden layer (two composed linear matrix operations each followed by an element-wise nonlinear function) can learn an arbitrary decision boundary or multivariate function [32]. This high level of expressivity enables the network to learn complex relationships and generalizations among high-dimensional parameters given repeated exposure during training.

Although a simple feed-forward neural network (FNN) is theoretically capable of simulating Noah-LSM in the manner of Figure 3, inductive biases are commonly introduced in model architectures in order to promote efficiency, explainability, stability, and parsimony. For example, most neural networks used for sequence modeling like the recurrent neural networks (RNNs) in Figure 4 maintain a "hidden" latent parameter $\vec{h}$ with an arbitrary number of dimensions. This vector is modified and passed along by each subsequent iteration, giving the network the ability to make generalizations and propagate information between time steps. Although $\vec{h}$ is typically difficult to interpret directly, the gradient descent process incentivizes the network to preserve and consolidate information that is needed to accurately generate the full sequence of predictions. Each of the following network architectures ultimately aim to improve the information quality of the latent vector by implementing algebraic structure, introducing statistical uncertainty, and encouraging sparsity.

$$\begin{aligned} \vec{h}_k &= \mathbf{E}(\vec{h}_{k-1}, \vec{s}_k, \vec{\psi}_k, \vec{\phi}) \\ \vec{s}_{k+1} &= \vec{s}_k + \mathbf{D}(h_k) \end{aligned} \tag{1}$$

The RNNs discussed here will follow the structure described by Equation 1, so that the decoder output is a prediction of the residual change in the state rather than the absolute

magnitude of the next state. Previous experiments with this project suggested that this improves model stability, and enables the model loss to balance local error in the residual with global error from the new state (which is vulnerable to accumulation) [33]. The most basic RNNs implement their encoder and decoder modules each as a FNN such that the latent state $\vec{h}_k$ is output directly by the final layer of the encoder, and then decoded to the output.

The basic RNN architecture is vulnerable to the so-called vanishing or exploding gradient problem, which arises from the fact that $\vec{h}_k$ is the product of a learned matrix operation. Since the encoder and decoder weights are shared between sequence steps, the back-propagation through time algorithm may update a parameter many times per sample during training, which can cause weights to diverge and cease learning [34]. Furthermore, since the hidden state undergoes a nonlinear transformation at each step, it is difficult for the network to sustain information over a long context of past observations.

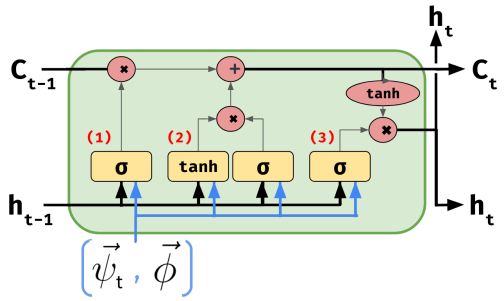

Figure 5: Long Short-Term Memory module. The previous latent vector $\vec{h}_{t-1}$ and inputs $\vec{\psi}_t$, $\vec{\phi}$ are used to select coefficients for linearly scaling the context vector C_{t-1}, then the inputs and updated C_t are used to generate the new latent vector h_t.

The Long Short-Term Memory (LSTM) architecture addresses these shortcomings by maintaining a separate hidden state $\vec{C}_t$ called the context vector. Rather than being generated by a matrix operation, the context vector is only modified by the output of a series of three "gates." These gates (numbered in Figure 5) include (1) the "forget gate", which uses a FNN to select a vector of values in the range (0,1). The vector is multiplied element-wise by $\vec{C}_{t-1}$ in order to selectively emphasize or diminish its activation. The "update gate" (2) transforms the inputs into a new coefficient vector in the range (-1,1), which is added to the context vector in order to retain information from the current time step. Finally, the "output gate" (3) generates a vector of multiplicative coefficients in the range (0,1) used to scale the new context vector $\vec{C}_t$ to the output latent state $\vec{h}_t$ [35].

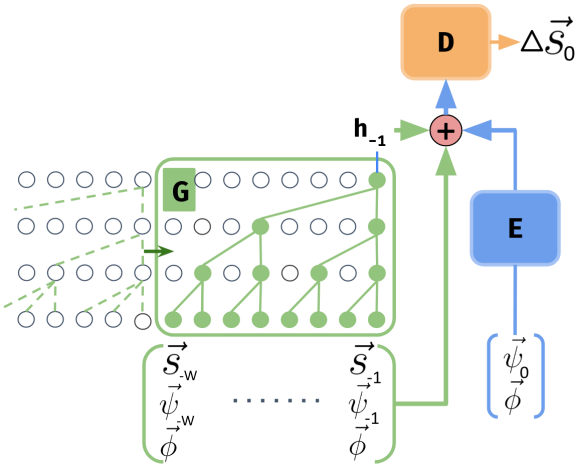

Figure 6: Deep Temporal Convolutional Network module, featuring three layers of dilated 1D convolutions for encoding previous time steps, a FNN-based residual encoder **E** for incorporating information from forcing variables, and a FNN decoder **D** that generates the residual state change.

Rather than maintaining a persistent latent vector between sequence steps, an alternative approach is to expose each timestep to the full sequence of past states. This is difficult to implement using traditional FNN modules since model complexity would scale quadratically with sequence length. The Deep Temporal Convolutional Network (DeepTCN) addresses this with dilated convolutions, which expand the receptive field of the encoder exponentially with depth. Skipped connections are used to more directly preserve information from the input sequence, and exogenous forcings associated with the prediction steps are independently encoded and added to the latent vector before it is decoded to a prediction [36]. As the dilated convolutional filters slide across the input sequence, each element in the latent sequence they generate is only exposed to past information, and has an implicit causal ordering from the kernel's structure. These properties grant the model an inherent temporal-

ity that may help it infer high-level patterns [37]. In this problem context, the DeepTCN could serve as an encoder that recurrently cycles its previous state predictions, or as a decoder that generates predictions from a latent sequence. Previous testing suggested that this architecture is the most promising for stably generating short-term dynamics at an hourly resolution.

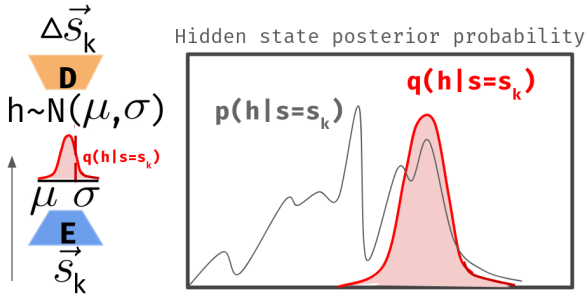

Figure 7: (left) Schematic overview of the variational encoder-decoder procedure. The encoder **E** produces vectors of coefficients parameterizing analytic distributions (ie mean and variance describing a standard normal distribution N). An independent random variable is used to sample a latent vector $\vec{h}$ from the distribution in accordance with its parameters. Finally, the chosen $\vec{h}$ is provided to the decoder **D**, which converts it back to the original state space to generate the residual. (right) The encoder's approximate posterior distribution $q(\vec{h}|s = s_k)$ over the latent vector space is selected by the encoder based on the input state $\vec{s}_k$. The encoder aims to use the previous-state information $\vec{s}_k$ to restrict the domain of valid latent activations to a simple distribution, which is a subspace of the intractable true posterior distribution $p(\vec{h}|\vec{s} = \vec{s}_k)$.

The final model type considered here are variational encoder-decoders, which enable models to explicitly characterize their uncertainty by generating parameters for analytic surrogate distributions $q(\vec{h}|\vec{s} = \vec{s}_k)$ over the latent space. The parameters are optimized to approximate the true posterior distribution of valid latent space activations given the input state s_k. During training, the model fits the surrogate distributions by seeking to minimize the Kullback-Leibler divergence between the two distributions, thereby minimizing their relative entropy. Backpropagation through the distribution is made possible by the so-called "reparameterization trick," which stochastically samples a latent vector according to the predicted distribution using an independent random variable. This simulates the posterior in terms of a Monte-Carlo estimator given many training samples [38][39].

There are several benefits to the variational approach. First, since local uncertainty in the latent space is explicitly encoded during training, variational models tend to be more spatially coherent, and tend to find more parsimonious representations of their dynamics [40][41]. This often means that a family of valid outputs can be generated by smoothly interpolating between two latent vectors. The approximate posterior distribution produced by the encoder can also be easily used to generate an ensemble of outputs according to the model's intrinsic uncertainty, which can enhance explainability. Finally, hidden states parameterized by variational distributions can be applied to any of the previous architectures as a means of improving latent space dynamics. For instance, an LSTM encoder could generate parameters to a distribution for each of its sequence outputs, which are individually sampled during inference, or a decoder could output a variational distribution that makes explicit the uncertainty in its residual.

The Transformer architecture has dominated many sequence modeling tasks in recent literature thanks to the key innovation of multi-head self-attention (MSA), which enables the architecture to learn complicated relationships between individual members of the input sequence regardless of their relative position. Additionally, transformers are efficient to train in parallel, unlike RNN-style architectures which rely on a chain of sequential operations, which makes them straightforward to train on a massive scale [42]. The results for natural language processing (NLP) [43] and image classification [44] tasks are impressive, however there are several key drawbacks that make them less appealing for a time series generation task like this one. First, the memory cost of a transformer scales quadratically with sequence length since MSA learns parameters relating every possible combination of input steps. This is compounded with the fact that the full input sequence needs to be re-initialized with every step during inference.

These properties are a direct trade-off with the Transformer's ability to train separate sequence steps in parallel. Furthermore, unlike RNNs and CNNs, Transformers don't have an inherent notion of order. In problems like NLP where sequence position conveys some information, a simple form of locality is introduced by adding a positional embedding vector directly to the inputs. Nonetheless, prior literature shows that transformers equipped with positional embedding still perform no better on basic time series forecasting tasks than a simple 2-layer FNN [45].

3 Procedure

3.1 Dataset Curation

The data used for this project is retrieved from the NLDAS-2 climatology available on NASA Goddard's GES DISC archive, where it is stored as a collection of GRIB-1 files for each hourly time step on a 225x464 spatial grid. So far, the 10 years between 2012 and 2022 have been acquired for model training and testing, but this can be expanded to a larger interval for further testing. Each time step has separate files for NLDAS-2 forcing and the subsequent Noah-LSM model run, both of which contain several superfluous data values.

It isn't feasible to train models directly from data read from the GRIB files since generating a set of multi-day single-pixel sequences by sparsely sampling files from a 10-year record would require hundreds of files to be opened and preprocessed in quick succession during training, which is computationally prohibitive. An intermediate format for storing and retrieving data is needed, which should be able to quickly index data across time.

The HDF5 format provides the ability to store a tree of multidimensional arrays in a compressed format, and to memory-map user-specified chunks for efficient cached retrieval from storage. At this stage, it's helpful to store the dynamic data (forcings, model outputs), static data (parameters and masks), and spatial/temporal coordinates together with unique labels assigned to each feature. This structure makes it easy to retrieve perform operations on arbitrary subsets of data given coordinate bounds and an ordered set of labels. The requisite metadata for interacting with multidimensional arrays in this way can be JSON-serialized and reloaded from the HDF5s using a Python object interface I call the FeatureGrid.

In practice, each individual HDF5 represents a 3-month interval of data from a region covering 1/6 of the CONUS domain. The files contain a (N, P, Q, F_D) array of dynamic features F_D on a PxQ pixel subset of the domain, measured at N consecutive time steps, a (P, Q, F_S) array of F_S static features on the same grid, and a (N, F_T) array of temporal but spatially-invariant features F_T (ie timestamp, solar zenith angle). The arrays will be organized in memory so that each chunk contains 8 features observed at 256 hours ($\sim$11 days) over a 32x32 pixel region. The size of each chunk is about 10.5MB, and the full size of each regional HDF5 file is about 3GB. This chunking strategy is useful because it promotes quick extraction of long time series needed for training without loading the full file into memory, and without compromising access to the spatial structure needed for analysis and visualization.

Before the data can be used to train and evaluate the models, it still needs to be sampled, preprocessed, and reshaped to the format expected by the model. This procedure is implemented using Python generators, which are able to produce the curated data on-demand by memory-mapping many HDF5 files in parallel and generating samples according to user settings. Since the generators are initialized at training time, model parameters like input/output variables, normalization bounds, and sequence length can be specified last-minute by the training configuration. During model development, it is also advantageous to be able to constrain the data population that training samples are drawn from. For example, models may be trained on a simplified problem by only sampling sandy-soil pixels in a spatially bounded region during the summer, or a model could be evaluated with respect to each vegetation type independently. The genera-

tors will open the HDF5 data as FeatureGrid objects, which provide the ability to restrict the data to a subset based on conditional constraints like these, then the generator will extract a shuffled batch of samples from the remaining domain.

After being normalized (typically by linearly scaling the data to a standard gaussian), the batch is yielded by the sample in the format expected by the models. The model input is a 3-tuple of arrays consisting of a (B, W, F_W) array of B samples of F_W dynamic features in the spin-up 'window' sequence of length W ($\vec{\psi}_k$ $\vec{s}_k$ for $k \in [-W, 0)$), a (B, H, F_H) array of F_H horizon covariate features (forcings $\vec{\psi}_k$ for $k \in [0, H)$) forming a length L sequence, and a (B, F_P) array of static parameter inputs $\vec{\phi}$. The output is a (B, H, F_Y) array of labels F_Y at each of the H horizon steps, corresponding to the true model states ($\vec{s}_k$ for $k \in [0, H)$).

3.2 Model Training

Variations and combinations of each of the model types described in Section 2.4 will adhere to the following general training strategy:

1. Train relatively small versions of the model on several highly restricted problem cases, randomly selecting combinations of hyperparameters like activation function, regularization method, and learning rate.
2. Using the most promising configurations, train small and large models on very broad range of samples to test generalizability.
3. Again selecting the most promising configurations, modify architectural aspects like window sequence length, output interval, latent vector size, and loss function.

This approach is facilitated by a model training framework I developed named tracktrain (`https://github.com/Mitchell-D/tracktrain`), which is driven by a custom JSON-based configuration system sufficient to initialize a model of any architecture, prepare a parameterized data generator, select an optimization method, and set training conditions. After a ModelDir creates a directory where settings, training progress, and intermittent model weights are stored according to a standardized naming system, the object automatically dispatches training. This allows for many model variants to be queued and trained concurrently. Once several models have been trained, a `ModelSet` object can be used to load model weights and compare results from many `ModelDir`s simultaneously.

3.3 Results Evaluation

The main goal of this project is to efficiently make predictions of soil moisture dynamics out to 14 days given a spin-up sequence of land surface states and a covariate sequence of atmospheric forcings. An ideal model will be numerically stable, computationally efficient, consistent with physics, and as general as possible. In order to determine whether models meet these objectives, they will be evaluated on a test dataset consisting of three nonconsecutive years of unseen data, and scored according to the following metrics and tests:

1. Average training and inference speed and memory cost.
2. Error accumulated when self-cycling over a period longer than 14 days.
3. Error rates with respect to individual vegetation and soil types.
4. Regional distribution of total error on the CONUS grid.
5. Time series of average diurnal and seasonal error rates per region.
6. Comparison between error in residual and error in magnitude.

An additional direction of inquiry is to test the performance of model variants trained to make predictions at a coarser time resolution than Noah-LSM, which has the potential to dramatically reduce the computational cost of inference. As discussed, Noah-LSM is originally evaluated at a 15 or 30 minute time interval since the dynamics are estimated by discretely evaluating a nonlinear algorithm. Neural networks aren't subject to the same constraint, and may be able to learn to generalize over several time steps.

References

[1] Y. Xia, K. Mitchell, M. Ek, J. Sheffield, B. Cosgrove, E. Wood, L. Luo, C. Alonge, H. Wei, J. Meng, B. Livneh, D. Lettenmaier, V. Koren, Q. Duan, K. Mo, Y. Fan, and D. Mocko, "Continental-scale water and energy flux analysis and validation for North American Land Data Assimilation System project phase 2 (NLDAS-2): 1. Intercomparison and application of model products," *Journal of Geophysical Research: Atmospheres*, vol. 117, no. D3, 2012. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2011JD016048.

[2] S. Sorooshian, "NCEP IMPLEMENTS MAJOR UPGRADE TO ITS MEDIUM-RANGE GLOBAL FORECAST SYSTEM, INCLUDING LAND-SURFACE COMPONENT," *GEWEX News*, vol. 15, p. 8, Nov. 2005.

[3] S. McCartney, K. White, R. Heim, C. Davis, and B. Smith, "Application of NASA SPoRT-Land Information System (SPoRT-LIS) Soil Moisture Data for Drought," May 2023.

[4] M. B. Ek, C. D. Peters-Lidard, Y. Xia, J. Meng, S. V. Kumar, H. Wei, J. Dong, A. Getirana, and S. Wang, "Next Phase of the NCEP Unified Land Data Assimilation System (NULDAS): Vision, Requirements, and Implementation," *NOAA-NCEP White Paper*, Aug. 2017.

[5] D. Mocko, "Preparing the Next Phase of NLDAS," Nov. 2023.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015. Number: 7553 Publisher: Nature Publishing Group.

[7] S. J. Russell and P. Norvig, *Artificial Intelligence: a Modern Approach*. London, 2010.

[8] L. Mahrt and H. Pan, "A two-layer model of soil hydrology," *Boundary-Layer Meteorology*, vol. 29, pp. 1–20, May 1984.

[9] L. Mahrt and M. Ek, "The Influence of Atmospheric Stability on Potential Evaporation," *Journal of Climate and Applied Meteorology*, vol. 23, pp. 222–234, Feb. 1984.

[10] H.-L. Pan and L. Mahrt, "Interaction between soil hydrology and boundary-layer development," *Boundary-Layer Meteorology*, vol. 38, pp. 185–202, Jan. 1987.

[11] F. Chen, Z. Janjic, and K. Mitchell, "Impact of Atmospheric Surface-layer Parameterizations in the new Land-surface Scheme of the NCEP Mesoscale Eta Model," *Boundary Layer Meteorology*, vol. 85, pp. 391–421, Dec. 1997. Num Pages: 391-421 Place: Dordrecht, Netherlands Publisher: Springer Nature B.V.

[12] J. H. E. Cartwright and O. Piro, "THE DYNAMICS OF RUNGE–KUTTA METHODS," *International Journal of Bifurcation and Chaos*, vol. 02, pp. 427–449, Sept. 1992.

[13] J. C. Schaake, V. I. Koren, Q.-Y. Duan, K. Mitchell, and F. Chen, "Simple water balance model for estimating runoff at different spatial and temporal scales," *Journal of Geophysical Research: Atmospheres*, vol. 101, no. D3, pp. 7461–7475, 1996. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/95JD02892.

[14] A. K. Betts, F. Chen, K. E. Mitchell, and Z. I. Janjić, "Assessment of the Land Surface and Boundary Layer Models in Two Operational Versions of the NCEP Eta Model Using FIFE Data," *Monthly Weather Review*, vol. 125, pp. 2896–2916, Nov. 1997. Publisher: American Meteorological Society Section: Monthly Weather Review.

[15] J. F. Mahfouf and J. Noilhan, "Comparative Study of Various Formulations of Evaporations from Bare Soil Using In Situ Data," *Journal of Applied Meteorology and Climatology*, vol. 30, pp. 1354–1365, Sept. 1991. Publisher: American Meteorological Society Section: Journal of Applied Meteorology and Climatology.

[16] P. G. Jarvis, J. L. Monteith, and P. E. Weatherley, "The interpretation of the variations in leaf water potential and stomatal conductance found in canopies in the field," *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, vol. 273, no. 927, pp. 593–610, 1976. Publisher: Royal Society.

[17] B. Jacquemin and J. Noilhan, "Sensitivity study and validation of a land surface parameterization using the HAPEX-MOBILHY data set," *Boundary-Layer Meteorology*, vol. 52, pp. 93–134, July 1990.

[18] G. Gutman and A. Ignatov, "The derivation of the green vegetation fraction from NOAA/AVHRR data for use in numerical weather prediction models," *International Journal of Remote Sensing*, vol. 19, pp. 1533–1543, Jan. 1998. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/014311698215333.

[19] F. Chen, K. Mitchell, J. Schaake, Y. Xue, H.-L. Pan, V. Koren, Q. Y. Duan, M. Ek, and A. Betts, "Modeling of land surface evaporation by four schemes and comparison with FIFE observations," *Journal of Geophysical Research: Atmospheres*, vol. 101, no. D3, pp. 7251–7268, 1996. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/95JD02165.

[20] J. L. Dorman and P. J. Sellers, "A Global Climatology of Albedo, Roughness Length and Stomatal Resistance for Atmospheric General Circulation Models as Represented by the Simple Biosphere Model (SiB)," *Journal of Applied Meteorology and Climatology*, vol. 28, pp. 833–855, Sept. 1989. Publisher: American Meteorological Society Section: Journal of Applied Meteorology and Climatology.

[21] V. Koren, J. Schaake, K. Mitchell, Q.-Y. Duan, F. Chen, and J. M. Baker, "A parameterization of snowpack and frozen ground intended for NCEP weather and climate models," *Journal of Geophysical Research: Atmospheres*, vol. 104, no. D16, pp. 19569–19585, 1999. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/1999JD900232.

[22] M. B. Ek, K. E. Mitchell, Y. Lin, E. Rogers, P. Grunmann, V. Koren, G. Gayno, and J. D. Tarpley, "Implementation of Noah land surface model advances in the National Centers for Environmental Prediction operational mesoscale Eta model," *Journal of Geophysical Research: Atmospheres*, vol. 108, no. D22, 2003. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2002JD003296.

[23] K. E. Mitchell, D. Lohmann, P. R. Houser, E. F. Wood, J. C. Schaake, A. Robock, B. A. Cosgrove, J. Sheffield, Q. Duan, L. Luo, R. W. Higgins, R. T. Pinker, J. D. Tarpley, D. P. Lettenmaier, C. H. Marshall, J. K. Entin, M. Pan, W. Shi, V. Koren, J. Meng, B. H. Ramsay, and A. A. Bailey, "The multi-institution North American Land Data Assimilation System (NLDAS): Utilizing multiple GCIP products and partners in a continental distributed hydrological modeling system," *Journal of Geophysical Research: Atmospheres*, vol. 109, no. D7, 2004. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2003JD003823.

[24] M. C. Hansen, R. S. Defries, J. R. G. Townshend, and R. Sohlberg, "Global land cover classification at 1 km spatial resolution using a classification tree approach," *International Journal of Remote Sensing*, vol. 21, pp. 1331–1364, Jan. 2000. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/014311600210209.

[25] E. R. O. a. S. C. S. G. S. S. D. of the Interior, "USGS 30 ARC-second Global Elevation Data, GTOPO30," July 1997.

[26] B. J. Cosby, G. M. Hornberger, R. B. Clapp, and T. R. Ginn, "A Statistical Exploration of the Relationships of Soil Moisture Characteristics to the Physical Properties of Soils," *Water Resources Research*, vol. 20, no. 6, pp. 682–690, 1984. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/WR020i006p00682.

[27] H. Wei, Y. Xia, K. E. Mitchell, and M. B. Ek, "Improvement of the Noah land surface model for warm season processes: evaluation of water and energy flux simulation," *Hydrological Processes*, vol. 27, no. 2, pp. 297–303, 2011. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/hyp.9214.

[28] B. Livneh, Y. Xia, K. E. Mitchell, M. B. Ek, and D. P. Lettenmaier, "Noah LSM Snow Model Diagnostics and Enhancements," *Journal of Hydrometeorology*, vol. 11, pp. 721–738, June 2010. Publisher: American Meteorological Society Section: Journal of Hydrometeorology.

[29] X. Liang, D. P. Lettenmaier, E. F. Wood, and S. J. Burges, "A simple hydrologically based model of land surface water and energy fluxes for general circulation models," *Journal of Geophysical Research: Atmospheres*, vol. 99, no. D7, pp. 14415–14428, 1994. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/94JD00483.

[30] E. Rosero, Z.-L. Yang, T. Wagener, L. E. Gulden, S. Yatheendradas, and G.-Y. Niu, "Quantifying parameter sensitivity, interaction, and transferability in hydrologically enhanced versions of the Noah land surface model over transition zones during the warm season," *Journal of Geophysical Research: Atmospheres*, vol. 115, no. D3, 2010. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2009JD012035.

[31] D. B. Jaynes, "Comparison of soil-water hysteresis models," *Journal of Hydrology*, vol. 75, pp. 287–299, Dec. 1984.

[32] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, Jan. 1989.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," Dec. 2015. arXiv:1512.03385 [cs].

[34] M. C. Mozer, "A focused backpropagation algorithm for temporal pattern recognition," in *Backpropagation: theory, architectures, and applications*, pp. 137–169, USA: L. Erlbaum Associates Inc., Jan. 1995.

[35] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, pp. 1735–1780, Nov. 1997. Conference Name: Neural Computation.

[36] Y. Chen, Y. Kang, Y. Chen, and Z. Wang, "Probabilistic Forecasting with Temporal Convolutional Neural Network," Mar. 2020. arXiv:1906.04397 [cs, stat].

[37] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," Sept. 2016. arXiv:1609.03499 [cs].

[38] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," May 2014. arXiv:1312.6114 [cs, stat].

[39] A. Ganguly, S. Jain, and U. Watchareeruetai, "Amortized Variational Inference: A Systematic Review," *Journal of Artificial Intelligence Research*, vol. 78, pp. 167–215, Oct. 2023. arXiv:2209.10888 [cs, math, stat].

[40] R. Lopez and P. J. Atzberger, "GD-VAEs: Geometric Dynamic Variational Autoencoders for Learning Nonlinear Dynamics and Dimension Reductions," Dec. 2022. arXiv:2206.05183 [physics, stat].

[41] J. N. Kutz and S. L. Brunton, "Parsimony as the ultimate regularizer for physics-informed machine learning," *Nonlinear Dynamics*, vol. 107, pp. 1801–1817, Feb. 2022.

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," Dec. 2017. arXiv:1706.03762 [cs].

[43] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," May 2019. arXiv:1810.04805 [cs].

[44] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," June 2021. arXiv:2010.11929 [cs].

[45] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are Transformers Effective for Time Series Forecasting?," Aug. 2022. arXiv:2205.13504 [cs].