# DRAFT Deep Learning for Spatial Time Series

*Geoffrey Fox, Indiana University 17 November 2020*

**Abstract**

We show that one can study several sets of sequences or time-series in terms of an underlying evolution operator which can be learned with a deep learning network. We use the language of geospatial time series as this is a common application type but the series can be any sequence and the sequences can be in any collection (bag) - not just Euclidean space-time -- as we just need sequences labeled in some way and having properties consequent of this label (position in abstract space). This problem has been successfully tackled by deep learning in many ways and in many fields. The most advanced work is probably in Natural Language processing and transportation (ride-hailing). The second case with traffic and the number of people needing rides is a geospatial problem with significant constraints from spatial locality. As in many problems, the data here is typically space-time-stamped events but these can be converted into spatial time series by binning in space and time.

Comparing deep learning for such time series with coupled ordinary differential equations used to describe multi-particle systems, motivates the introduction of an evolution operator that describes the time dependence of complex systems. With an appropriate training process, we interpret deep learning applied to spatial time series as a particular approach to finding the time evolution operator for the complex system giving rise to the spatial time series. Whimsically we view this training process as determining hidden variables that represent the theory (as in Newton's laws) of the complex system.

We formulate this problem in general and present an open-source package FFFFWNPF as a Jupyter notebook for training and inference using either recurrent neural networks or a variant of the transformer (multi-headed attention) approach. This assumes an outside data engineering step that can prepare data for ingest into FFFFWNPF.

We present the approach and a comparison of transformer and LSTM networks for time series of COVID infection and fatality data from 314 cities as well as hydrology from 671 locations

The paper concludes with a discussion of future applications including earthquake science, logistics (job scheduling), and epidemiology as well as other important neural networks -- graphs, convolutional, convLSTM. We expect to use this technology with MLPerf datasets. We intend to understand how complex systems of different types (different membership linkages) are described by different types of deep learning operators. Geometric structure in space and multi-scale behavior in both time and space will be important. We anticipate that the current forecasting formulation is easily extended to sequence to sequence problems.

## 1. Introduction

There is increasing recognition of the importance of deep learning in data-driven discovery across a broad range of applications. In this paper, we study time series where the MLPerf [1], [2] working group led by Cisco technical leader Xinyuan Huang has recently highlighted many areas and available datasets [3]. Logistics, network intelligence, manufacturing, smart city, and ride-hailing [4] (transportation) are major Industry areas having important time series while medical data is often of this form. We note that similar technical approaches (recurrent neural nets and Transformers) are often used for both time series and "sequence to sequence mapping" as seen in the major voice and translation areas separately studied at MLPerf. We focus here on the analysis of time-dependent data for forecasting where approach can be illustrated by the examples below. There is a class presentation covering this material [5], [6], which extends an earlier technical report [7].

We compare deep learning for such time series with coupled ordinary differential equations used to describe multi-particle systems, and this motivates the introduction of an evolution operator that describes the time dependence of any complex systems. With an appropriate training process, we interpret deep learning applied to spatial time series as a particular approach to finding the time evolution operator for the complex system giving rise to the spatial time series. Whimsically we view this training process as determining hidden variables that represent the theory (as in Newton's laws) of the complex system. Below we analyze three examples focussing on what we term a spatial bag. These are problems consisting of a collection of points (thought of members in a space) which have the same dynamics but with different parameters and different initial conditions. The points in the bag have time series and static data which can be considered as specifying parameters allowing different evolution operators for each point. In the two methods presented here the static and dynamic data are treated in identical fashion with static data viewed as a time series with time-independent values. This is a common approach in the literature but I did not see it articulated definitively anywhere. We use it as it fits our view of parameterized operators which is not obtained from alternate methods that treat static data outside the LSTM or Transformer.

We formulate this spatial bag problem in general and present an open-source package FFFFWNPF as a Jupyter notebook for training and inference using either recurrent neural networks or modified transformers using multi-headed attention at decoder stage and an LSTM for the encoder. This assumes an outside data engineering step [8], [9] that can prepare data for ingest into FFFFWNPF. The software is open source and available [10] but it needs more attention to clean API's, further testing and documentation. The notebook is modified from that prepared by Google [11] for the original transformer [12], so you can easily see the nature of changes made.

We motivate our approach in section 2 considering a study of Newton's laws for molecular dynamics and COVID-19  infection and fatality data from 314 cities (space points) using a well established LSTM approach [13]. In section 3, we describe the spatial bag and the Transformer model for it. Section 4 looks at LSTM and Transformer models for COVID-19 and a large hydrology dataset [14]–[18]. In this study the work is motivated by realistic applications but only uses them to study the technology. In later works we will apply these ideas to answer science questions. The final section has conclusions

## 2. Motivating Examples

### 2.1 Deep Learning as a Particle Dynamics Integrator

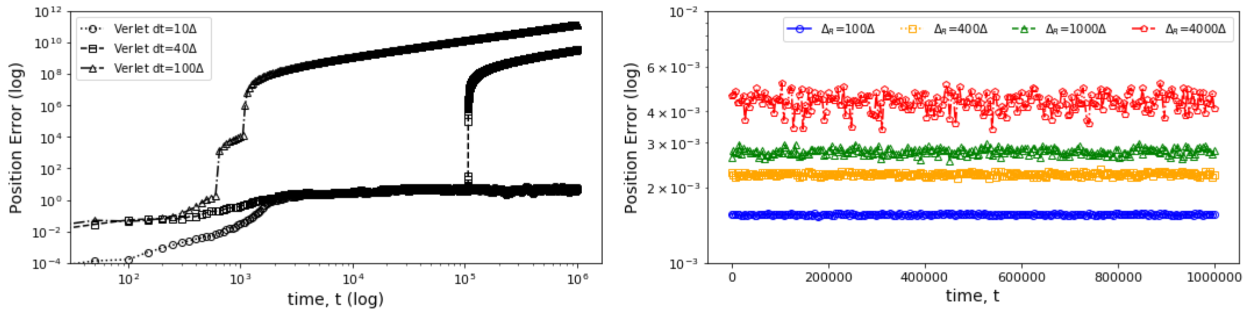

*Fig. 1. The average error in position updates for 16 particles interacting with an LJ potential, The left figure is standard MD with error increasing for $\Delta t$ as 10, 40, or 100 times robust choice (0.001). On the right is the LSTM network with modest error up to $t = 10^6$ even for $\Delta t$ = 4000 times the robust MD choice.*

Molecular dynamics simulations rely on numerical integrators to solve Newton's equations of motion. Using a sufficiently small time step to avoid discretization errors, these integrators generate a trajectory of particle positions as solutions to the equations of motions. In [19]–[21], the IU team introduces an integrator based on recurrent neural networks that is trained on trajectories generated using a traditional Verlet integrator and learns to propagate the dynamics of particles with timestep up to 4000 times larger compared to the Verlet timestep. As shown in fig. 1 (right) the error does not increase as one evolves the system for the surrogate while the standard integration in fig. 1 (left) has unacceptable errors even for time steps of just 10 times that used in an accurate simulation. The surrogate demonstrates a significant net speedup over Verlet of up to 32000 for few-particle (1 - 16) 3D systems and over a variety of force fields including the Lennard-Jones (LJ) potential.

We often think of the laws of physics described by operators that evolve the system given sufficient initial conditions and in this language, we have shown how to represent Newton’s law operator by a recurrent network. We expect that the time dependence of many complex systems: Covid pandemics, Southern California earthquakes, traffic flow, security events can be described by deep learning operators that both capture the dynamics and allow predictions. In the covid example below for example one can learn an operator that depends on the demographics and social distancing approach for a given region.

### 2.2 Deep Learning to describe Covid Daily Data

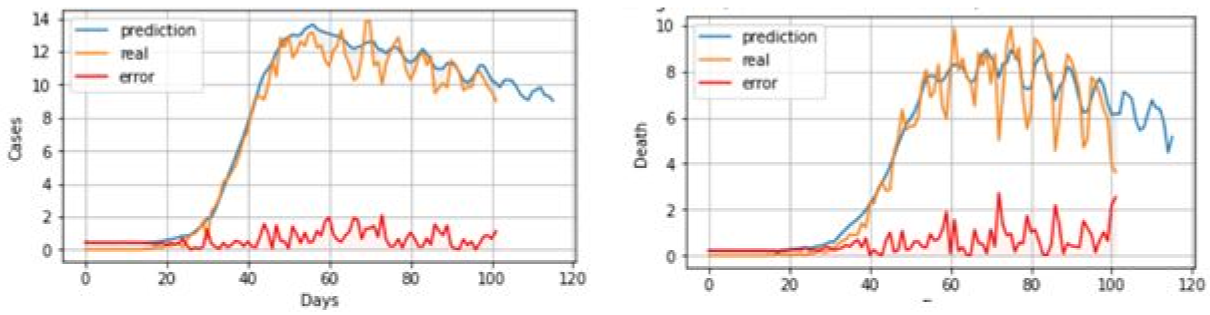

*Fig 2: Deep Learning fits to Covid case and death data from Feb. 1 to May 25, 2020, with predictions 2 weeks out and showing a weekly structure. The data is the square-root of counts for individual counties normalized between 0 and 1.*

There are extensive collections of daily data for the number of Covid reported cases and deaths. These can be described by epidemiological models plus empirical fits [22] but as proposed above and illustrated in fig. 2, we developed a deep learning model [23] that learned a Covid daily evolution operator from (initially) 110 separate time series of curated (by the University of Pittsburgh) data for different US cities. The time series were 100 days long and the model was a 2 layer LSTM recurrent network similar to that used to describe the evolution of molecular dynamics above. Additional features were learning from the demographics (fixed data for each city) as well as time-dependent data and by predicting ahead for two weeks with each series as shown in the figure. This capability is important in any application with multiple time scales. For example, in earthquake forecasting multiscale in time effects are critical and one might want to combine a general forecast for the next time step (days to months) with the probability of the big one happening in the next 10 years. For 37 of the 110 cities reliable empirical (not deep learning) fits are available to the case and death data up to April 15, 2020 [22]. A single deep learning time evolution operator can describe these 37  separate datasets and smooth fitted data leads to very accurate deep learning descriptions shown in fig. 3. For both figs. 2 and 3, the data is divided into windows of size 5, 9, or 13, and cases and deaths were simultaneously trained together with demographic data. The later work in section 4 increases the number of locations to 314 and links with time-dependent mobility and social distancing data[24].

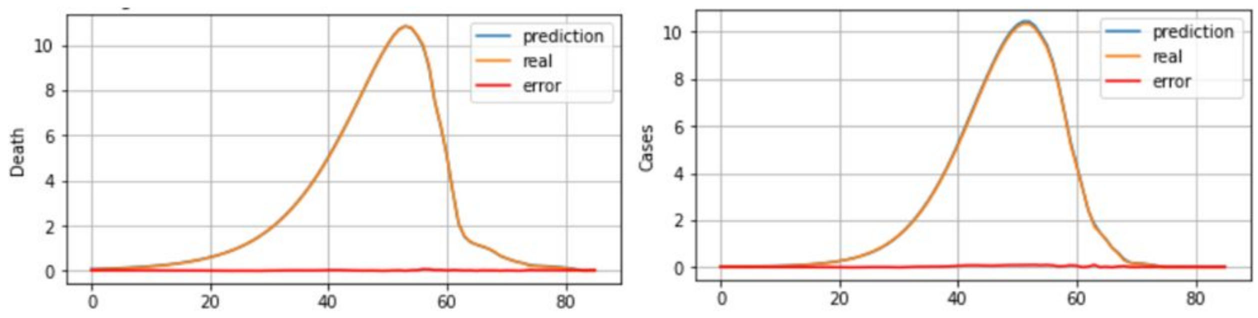

*Fig 3: Deep Learning Fits empirical Covid data descriptions with 37 separate results shown as summed over cities. The cases and death were learned together in time series for different locations*

## 3. General Formulation of Deep Learning for Time Series

### 3.1 Spatial Bag Problem

We now generalize the above to a spatial bag of time series shown in fig. 4, where we have a set of time series where the spatial distances (e.g. locality) between points is not important; rather they are differentiated by values of properties which can either be static (such as percentages of population with high blood pressure for Covid example above or minimum annual temperature for hydrology catchment studies) or dynamic (such as a local social distancing measure). In later papers we will discuss problems which combine the features of spatial bags and distance locality where convolutional networks (especially convLSTM) are clearly useful.

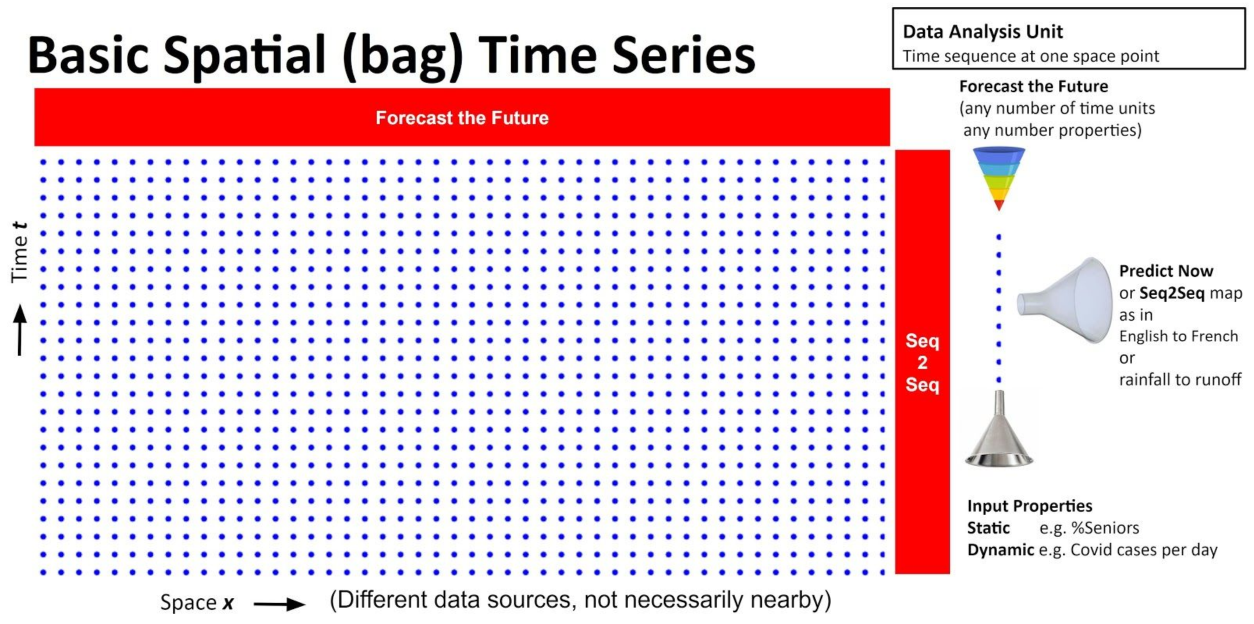

*Fig 4: Illustration of a spatial bag with associated time sequences. t labels time and x location, The Seq2Seq and forecast modes are illustrated and the structure of input data and predictions.*

## 3.2 Using Transformers for Spatial Bags

The deep learning methods use approaches that were largely originally developed for natural language processing.

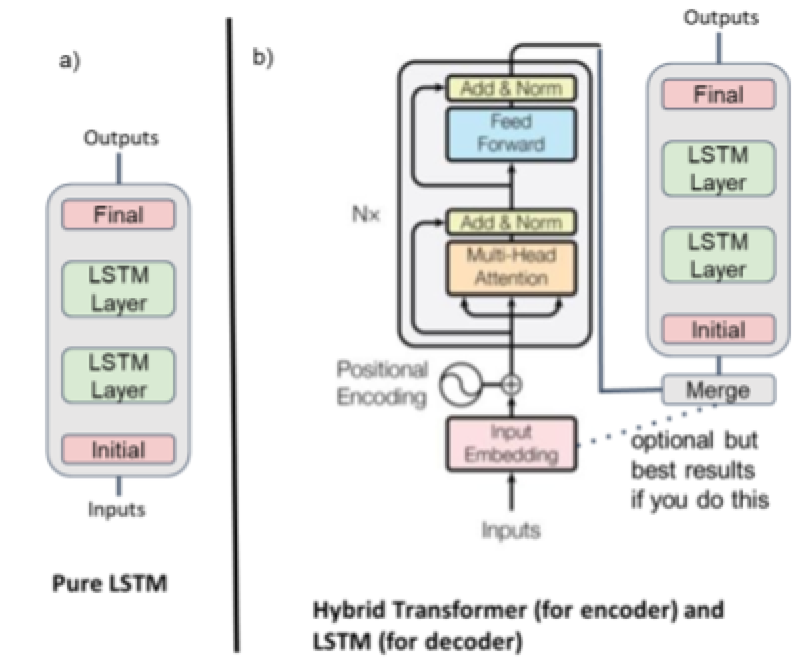

*Fig. 5: a) Pure LSTM and b) Hybrid Transformer model discussed in this paper. The latter has an attention-based decoder and an LSTM-based final encoder*

Recurrent Neural Networks RNN explicitly focus on sequences and pass them through a common network with pretty subtle features. RNN are designed to gather a history which allows the time dependence to be remembered Attention-based methods [25] are more modern and perhaps somewhat easier to understand as attention is a simple idea.

NLP is basically a classification problem (look up tokens in a context sensitive dictionary) whereas (science) tends to be numerical and so it is not immediately obvious how to use attention in technical time series and we describe one possible approach in this paper. There have been a few studies of transformer architectures for numerical time series such as [26]–[32] but there is not a large literature.

Attention [12], [33] means that you “learn” structure from other related data and look for patterns using a simple “dot-product” mechanism discussed later matching structure of different sequences; there are other approaches to match patterns which is a good topic for future work. Here we use a simple attention mechanism in an initial decoder but use a recurrent net LSTM for the encoder as shown in fig. 5b). Such mixtures have been investigated and compared [34], [35]. We compare the two architectures shown in a) and b) of fig. 5; a pure LSTM used in sec. 2 and a hybrid transformer

**Scaled Dot-Product Attention and the Vectors Q K V**

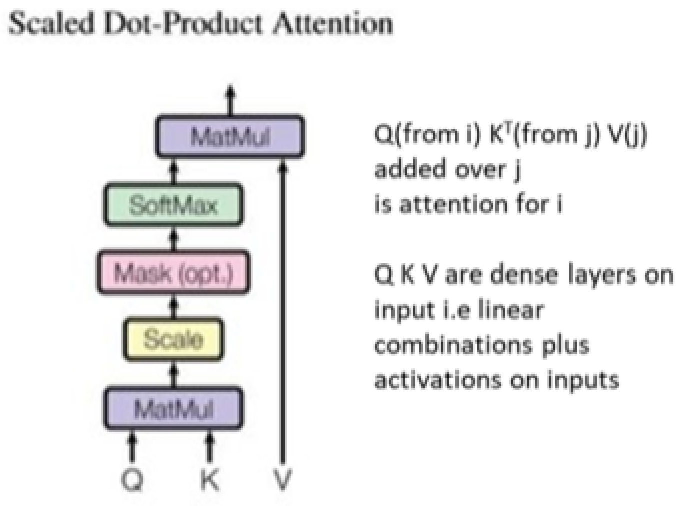

*Fig. 6: The Q K V layers of the Attention Mechanism.*

The basic item for LSTM and Transformer is the same; a space point with a time sequence with each time in the sequence having a set of static and dynamic values. In an LSTM the sequence is “hidden” and you have to unroll the recurrent network to see it. However in transformer the different time values in a sequence are treated directly and so each item contains W terms (W is size of time sequence), Each term is embedded in an input layer and then mapped by 3 different layers into vectors Q (query) K (key) V . As shown in fig. 6, one matches terms i and j by calculating $Q(i)K^T(j)$ and ranking with a soft-max step. This multiplies the characteristic vector V(j) of this pattern and the total attention A(i) for item i, is calculated as a weighted sum over values V(j). There are several different attention “heads” (networks generating Q K V) in each step and the whole process is repeated in multiple encoder layers. The result of the encoder step is considered separately for each item (each time in a time sequence at a given location) and the embedded input of this layer is combined with the attention as input to the LSTM decode step.

**Choosing group of items over which Attention Calculated**
In natural language processing, you look for patterns among neighbouring sentences but for science time series you can have larger regions as spatial bags have no locality. This leads to many choices as to the space over which attention is calculated as one can’t realistically consider all items simultaneously. Suppose we have $N_{loc}$ locations; each with $N_{seq}$ sequences of

length W. Then the space to be searched has size $N_{loc}$. $N_{seq}$ . W which is too large. In COVID-19 example $N_{loc}$= 314,  $N_{seq}$ ~ 200 and W up to 13. In the hydrology example in sec. 5, $N_{loc}$= 671, $N_{seq}$ ~ 7000 and W up to 270. The next subsection describes the 3 search strategies we have looked at in FFFFWNPF: and depicted in fig. 7. There is

- Temporal search:  points in sequence for  fixed location
- Spatial search: locations for a fixed position in sequence
- Full search: complete location-sequence space

One will need to sample items randomly as only a small fraction of space is looked at in one attention step whatever method used. Note that in all cases we used a batch size of 1 as the attention space was effectively the batch. Actually in the LSTM stage, the different locations in the attention space were considered separately and the attention search space became the batch. In the work reported here the attention space (and batch size) was set to $N_{loc}$ but this is not required and would not work in some cases with large values of $N_{loc}$. Even in examples considered here, the search space can get so large that one needs to address memory size issues and we describe this in sec. 3.3. Note the encode step of the transformer is many matrix multiplications and gets excellent GPU performance and typically LSTM decoder would be a significant part of the compute time and so the addition of attention is not a major performance hurdle.

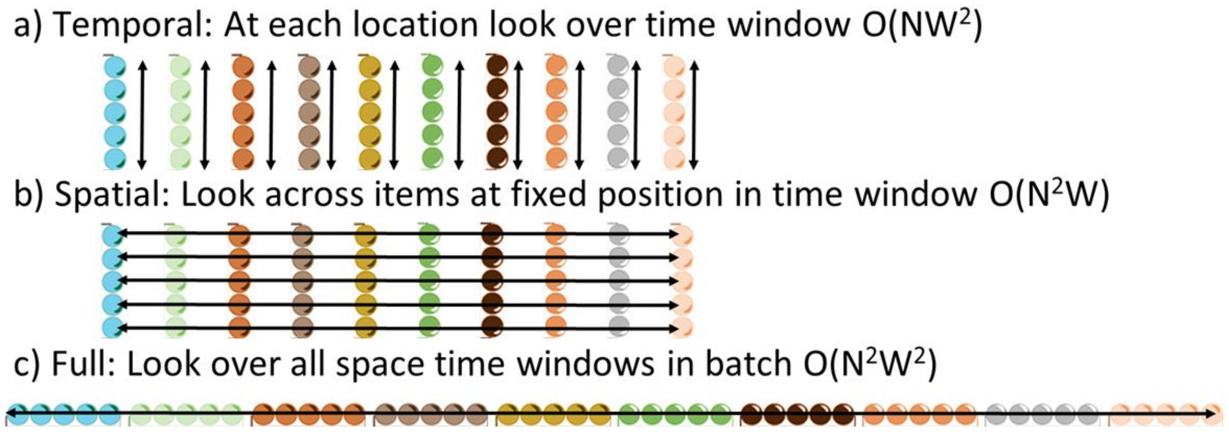
a) Temporal: At each location look over time window O(NW²)
b) Spatial: Look across items at fixed position in time window O(N²W)
c) Full: Look over all space time windows in batch O(N²W²)

*Fig. 7: Three search strategies discussed in paper. $N=N_A$ is the number of items considered in attention search- each labelled by a location and starting time - and each a window of length W. W=5 in diagram. $N_A$ = $N_B$ =* $N_{loc}$ here.

An epoch contains $N_{loc}$.$N_{seq}$ sequences each of length W and consisting of static and dynamic variables characteristic of location. For an LSTM, you would have a batch which consists of $N_B$ ~ $N_{loc}$ sequences. The number of batches in an epoch is approximately $N_{seq}$. For a transformer the batch is currently unwrapped as discussed above and used as the attention space. Memory use or compute issues could require a different strategy separately considering batch and attention space.

Note the attention space choice implies different initial shuffling to form batches and also different prediction stage approaches. For spatial and temporal searches one can keep all locations at a particular time value together in both forming batches and in calculating prediction. For the full search the complete set of $N_{loc}.N_{seq}$ sequences must be shuffled. In practice we combined the spatial and temporal search and accumulated the results of these two attention searches.

**3.3 Comments on FFFFWNPF Implementation**

**Sequence Memory Use:** Suppose that one has $N_{loc}$ locations, $N_{seq}$ sequences and time windows of length W. This requires memory of order $N_{loc}N_{seq}W$ which for hydrology example exceeds CPU memory for W □ 10. However the sequences are only needed at the time the sequence is being used in a batch and so we use “Virtual Windows” where the system is set up with just time values with predictions calculated for sequences that end at each time value. Then one forms the sequences inside the deep learning loop dynamically for each batch. This requires care to be efficient and needs custom Tensorflow training but works well with little overhead. Probably this approach should always be used for time series.

**Attention Search Memory Use:** The full search requires matrices of size $(N_AW)^2$ and this can be too large to fit in the GPU memory. As W is chosen for a given scenario, if this limit is seen, one essentially needs to reduce the number of locations ($N_A$) and this is done (hidden from user) by breaking matrix into blocks inside the function calculating the scaled dot product attention. These blocks require tensor slicing which is a little tricky as sliced tensors cannot be assigned in current tensorflow but appending blocks to a list and using concat achieves this goal. This approach was used for larger values of W for models that used the mechanism of fig. 7c). In future work, we can of course look at parallelism to address both memory use and performance. For W=120 for example, each epoch takes over an hour to calculate on Colab.

**Deep Learning Approach:** All these results used Tensorflow and Jupyter notebooks but they can surely use PyTorch or other frameworks. The simplest LSTM can be done with the basic sequential Keras model with a succession of layers. The Transformer has a more complex structure that needs Tensorflow’s custom training. We designed a custom monitor that checkpointed (using Tensorflow’s standard mechanism) weights and monitored the variation of loss with increasing epoch number. Large increases in loss were rolled back during the training and at the end of the training, the best checkpointed result was used. The number of rollbacks increased at end of run when improvement was anyway very small

**Real-Time Issues**: Often time-series are observed in real-time and need to be responded to with low latency requiring inference to execute at the edge. This was not relevant for examples considered here with daily data. However these methods need to be reviewed for real-time inference scenarios when the large search space that can be used by the transformer can significantly increase prediction time latency.

**Structure of workflow:** We can divide data processing for the type of problem considered here into five stages: pre-notebook; specialized notebook input; generic data pre-processing; training; visualization. We give details of five steps below

1) The pre-notebook data engineering performs actions specific to a particular domain as, for example, in forming binned time series from recorded earthquake events.
2) This stage prepares data for the notebook which reads the data (typically csv files) into a set of numpy arrays. This includes dynamic and static properties as well as metadata such property and location names.
3) In the important third stage, the notebook performs generic preparation tasks common to all datasets. This includes normalizing static and dynamic data by taking powers (square root, cube root, log to reduce standard deviation/mean) and linear transformations so values lie between 0 and 1. Also one must generate sequences (or prepare the virtual sequences described above under sequence memory use) and generate predictions associated with each sequence final time value. The predictions include futures which for COVID-19 use cases was for two weeks ahead. These cannot be found for sequences whose endpoint is within two weeks of the final data point; those points are set to NaN. In general the method accommodates any missing data for predictions but not for the input sequences where all data must be present. The generic input processing includes adding of positional space and time encoding for both LSTM and Transformer. We use simple linear indices for both space and time supplemented by periodic time encoding which is weekly for the COVID-19 data but annual for hydrology. This periodic structure corresponds to series such as ($\cos\theta$, $\sin\theta$) where $\theta$ runs from 0 to $2\pi$ over the period length. Note this seems extravagant but it is not and performance of the network is not significantly impacted by adding either these encoding or by the extra future variables.
4) The next stage is a custom Tensorflow training where we must of course design the network from a collection of class instances. It is also set up with a monitor that controls the backup and restores the weights if the optimizer sends the fit into left field and the loss is significantly increased. We must set the parameters to control fit such as the search space for the transformer and the usual hyperparameters including number and size of layers, dropout, epoch, batch, and validation set, The training code must generate the sequences every batch if virtual windows are used. The long Tensorflow training runs ( up to over an hour per epoch) sometimes fail for Colab system glitches and jobs are often set up to restart from the backups. Further the weight architecture is independent of window size W, so large window runs W □ 60 can be initialized with results of faster runs with smaller window size -- typically choosing a stage which was not fully converged but had a loss that was perhaps 30% above the final value. We use a simple custom loss function which certainly needs to recognize any missing prediction data designated by NaN in value. This is easy to test on and as loss function is additive over predictions it is sufficient to just skip over such points in calculations in the custom loss function.
5) The last notebook activity is the many post fit visualization and analysis steps where care is needed to generate accurate efficient predictions.

## 4. Application to Hydrology Problems

This work was motivated by an NCAR summer school including a video lecture [36] describing the use of recurrent neural networks in hydrology. There is also a good review [37] of deep learning in Hydrology with 129 papers. Many of the 129 papers are based on the CAMELS dataset from NCAR [15] where we focus on one of the most sophisticated analyses by Kratzert [16], [18]. A summary of this is contained in the resource [38] produced for a summer undergraduate research project [39]. The CAMELS data has 671 easily used locations (catchments) where 6 observables (rainfall, runoff, etc.) are defined on each location for 20 years. As well as this dynamic data there are 27 static variables for each location. The goal is to build a single model describing these 671 locations. This can be posed in many ways with different input and output choices. It can also be formulated as a sequence to sequence model or as a sequence to forecast model (as Covid and Kratzert did). The loss function can be MSE or similar or more subtly the Nash–Sutcliffe efficiency NSE normalized by the measured standard deviation.

In this paper, we are not aiming at science but rather a technology evaluation of LSTM and the hybrid transformer. So we asked how well these models described the full dataset rather than using half the dataset to predict the other half as was done in [16], [18]. We looked at the 671 locations and 7031 daily data and windows W from 5 to 120. We did not see any advantages of large windows for the CAMELS dataset and present results for W=25. Interestingly the hydrology analysis preferred the spatial and temporal search strategies, (a) and b) in fig. 7, while COVID-19 analysis hybrid transformer analysis preferred the full search, c) in fig. 7. In both cases, the hybrid transformer obtained a final loss MSE that was 20% lower than the LSTM. Further, we found 2 Encoder layers gave good answers with quick convergence and either 4 (usual choice) or 8 heads per layer were sound. We explored different choices of the network embedding for Q K V and input data but a simple single layer with SELU activation for each embedding worked well. The internal representation of the space time points used 128 bits. However this hyperparameter search was not thorough.

The 27 static variables p_mean, pet_mean, aridity, p_seasonality, frac_snow_daily, high_prec_freq, high_prec_dur, low_prec_freq, low_prec_dur, elev_mean, slope_mean, area_gages2, forest_frac, lai_max, lai_diff, gvf_max, gvf_diff, soil_depth_pelletier, soil_depth_statsgo, soil_porosity, soil_conductivity, max_water_content, sand_frac, silt_frac, clay_frac, carb_rocks_frac, geol_permeability, were used with details given in [40]. The 6 dynamic data with daily values are given in the table below

| | **Dynamic attribute** | **description** | **unit** |
|---|---|---|---|
| **1** | **prcp(mm/day)** | daily cumulative precipitation | mm/day |
| **2** | **srad(W/m2)** | average short-wave radiation | W/m2 |
| **3** | **tmax(C)** | daily maximum air temperature | C |
| **4** | **tmin(C)** | daily minimum air temperature | C |
| **5** | **vp(Pa)** | vapor pressure | Pa |
| **6** | **QObs(mm/d)** | Discharge (cubic meters per day/ basin area) | mm/day |

**Table: CAMELS data recorded daily**

All this data was scaled between 0 and 1 while QObs and prcp had the cube root taken before this scaling to give greater representation quality by increasing scaled standard deviation.

We investigated pure LSTM and hybrid transformer representations of the hydrology data with a variety of choices for window size W  and the different attention mechanisms described in section 3.2. We did not see significant dependence on W for values ≧ 21 and typical results are shown in fig. 8 for W = 25. There are a lot of fluctuations in the rainfall (prcp observable) making figure 8 a) left) hard to interpret and it is shown with an expanded scale in fig. 8 d).  As mentioned already, we obtained the best representations using the stratified search strategies across spatial or time directions, a) and b) in fig.7.

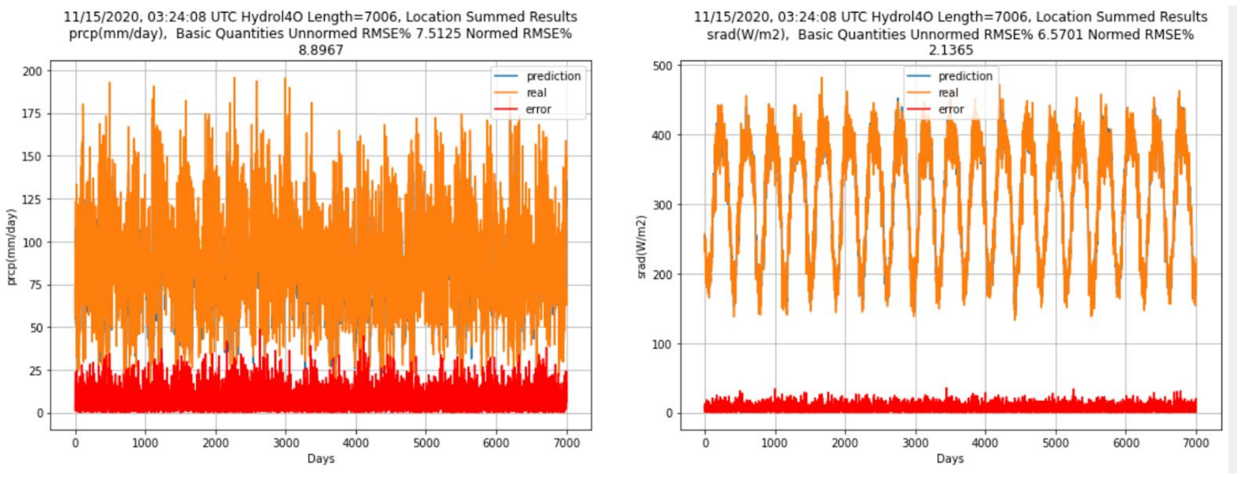

*Fig. 8(a) Results of the hybrid transformer for the first two daily time series prcp and srad,predicting one day in the future. This had 4 attention heads and 2 encoder layers and used search strategies a) and b). The left plot is expanded in fig. 8(d). The data in figures 8 a) to d) are the sum over locations of normalized data with cube root taken for prcp and QObs.*

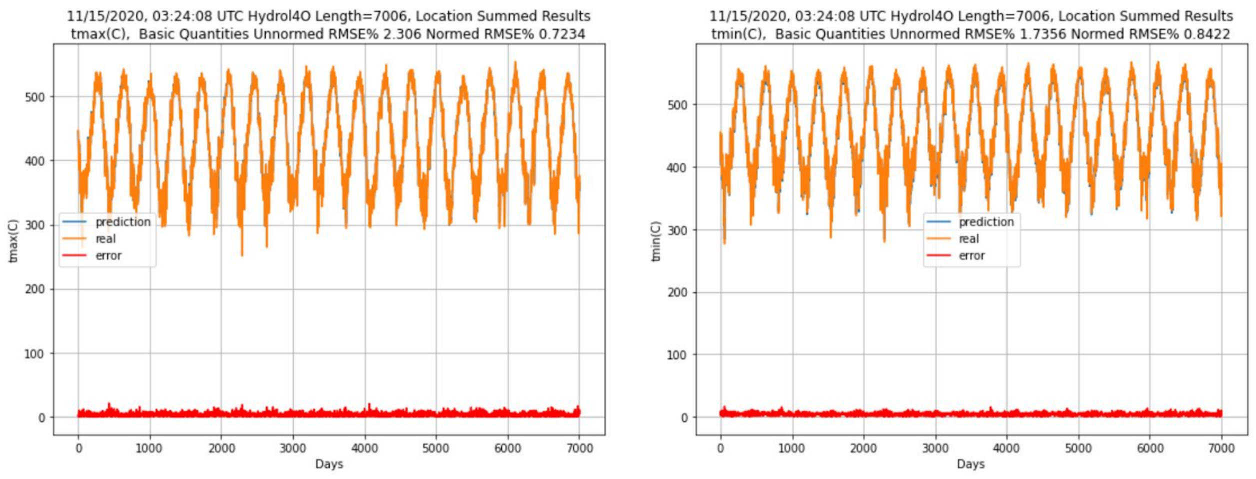

*Fig. 8(b) Results of the hybrid transformer for the middle two daily time series tmax and tmin,predicting one day in the future. This had 4 attention heads and 2 encoder layers and used search strategies a) and b)*

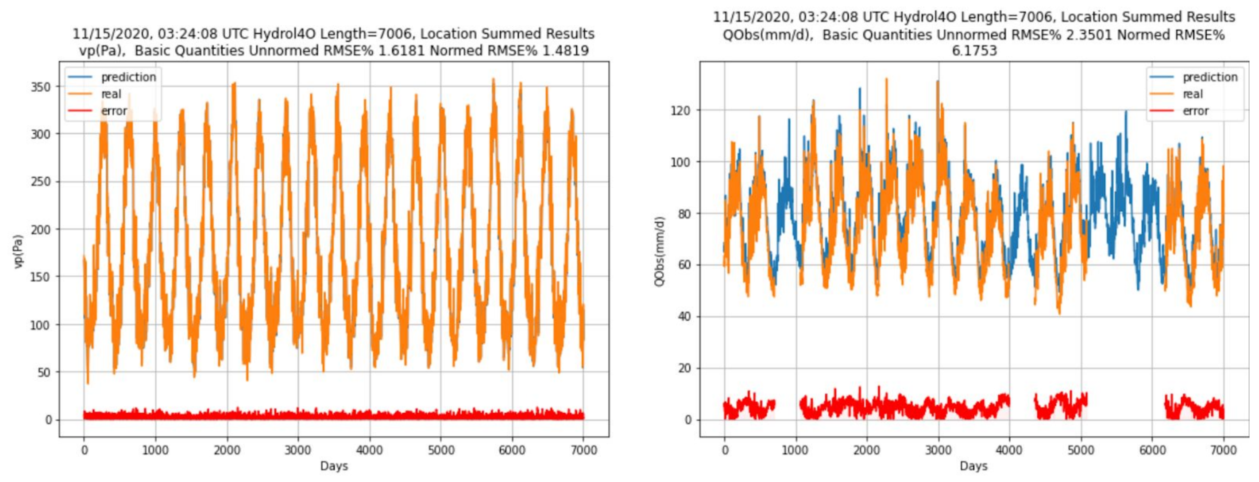

*Fig. 8(a) Results of the hybrid transformer for the final two daily time series vp and QObs, predicting one day in the future. This had 4 attention heads and 2 encoder layers and used search strategies a) and b). Note a few measurements of QObs are missing and so summed data is not available for some days in the right plot. The fit does all 671 locations separately and so the region where the sum is not plotted is in fact well covered in the fit.*

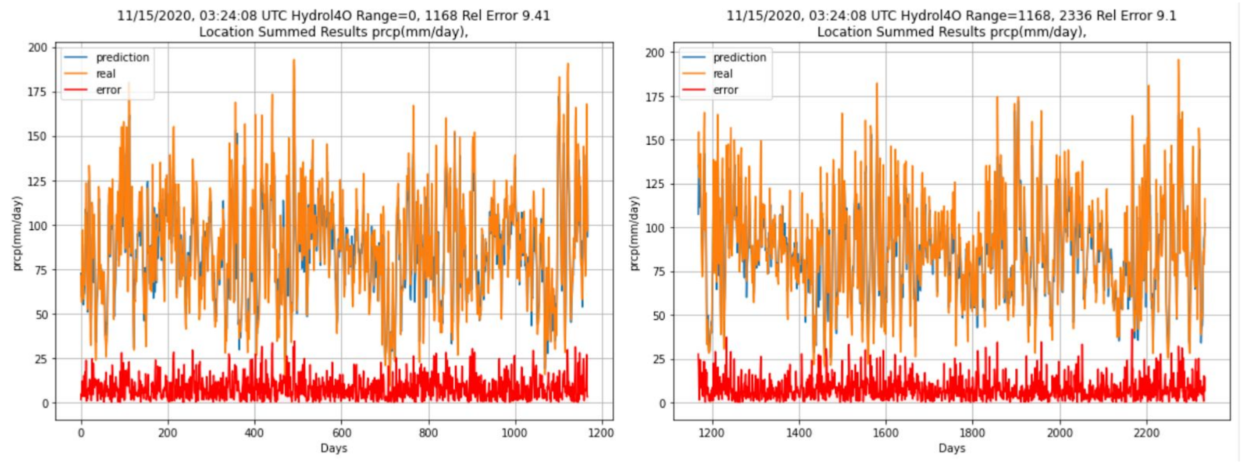

*Fig. 8(d) Expansion of the plot in fig. 8(a) left with x-axis expanded a factor of 6. We only record a third of the days as these illustrate what is going on quite well. The other 4 plots for remaining days are similar*

We also applied the hybrid transformer model to the COVID-19 data introduced in sec. 2. Typical results are shown in figure 9 that not only gives the summed totals as in figs.2, 3, 8 but also the particular description of data from New York City. Unlike the hydrology data, the full attention search fig. 7 c) gave the best description.

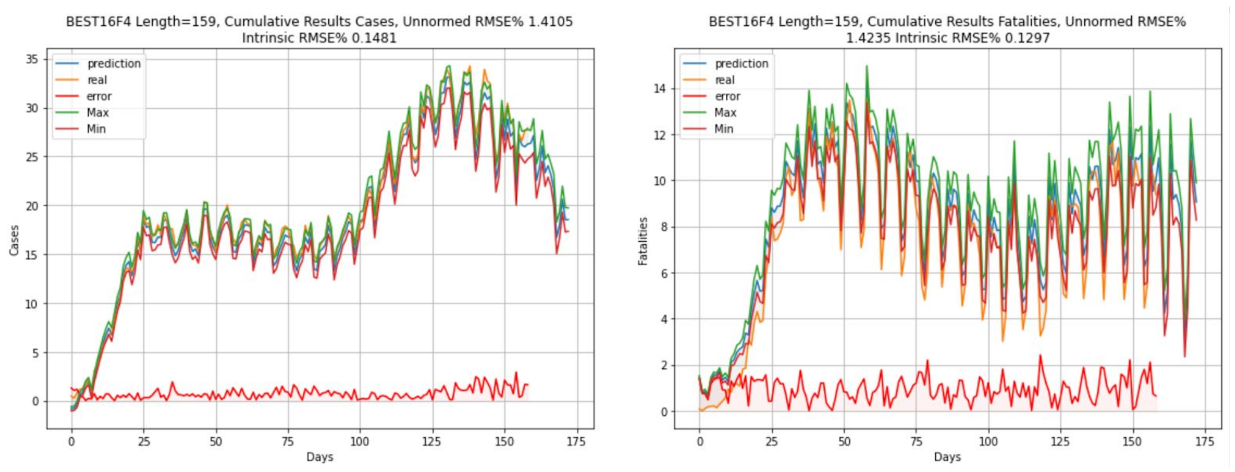

*Fig. 9(a) Results of the hybrid transformer for the COVID-19 infections (left) and fatalities (right) for a larger sample than that shown in fig. 2. The network had 4 attention heads and 2 encoder layers and used search strategy c) and a window size of 9. We show variation in predictions across different samplings of the attention search space. The figure shows sum over counties/cities of normalized fitted data which is square root of individual counts*

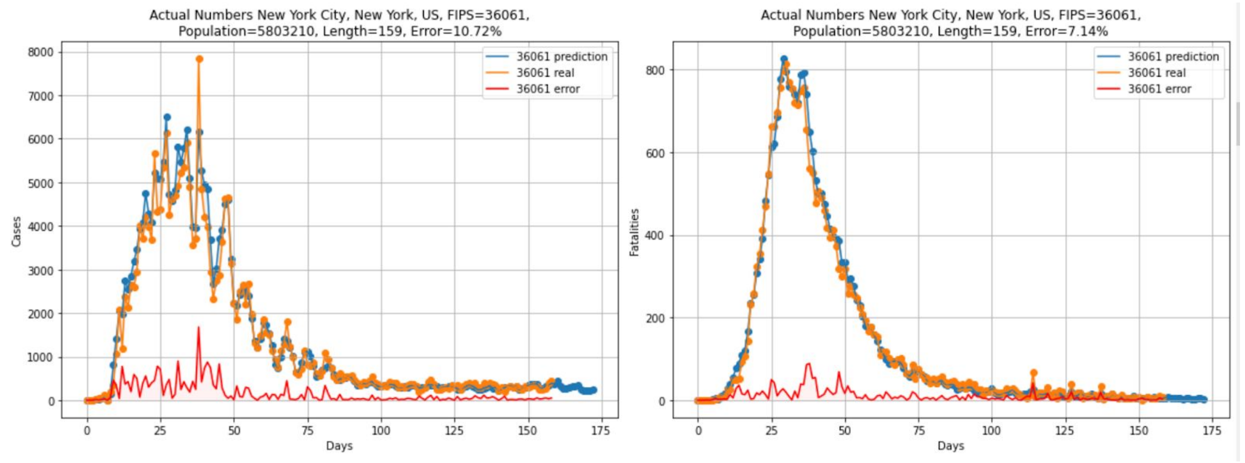

*Fig. 9(b) Results of the hybrid transformer for the COVID-19 infections (left) and fatalities (right) in New York City for the fit presented in fig. 9 a). This figure shows true counts*

## *5.* **Conclusions**

Above we have given examples of recurrent networks of the time evolution operator for complex systems and we are extending this to other areas. We see the mix of networks used above as a base approach applicable to many problems. Some examples need additional features: earthquakes (with fault lines) and transportation (road systems) need some variant of graph networks while mixtures of convolutional and recurrent networks (such as convLSTM) are used in weather and again earthquakes where the time series features can consist of images. Here we identify a new problem class -- spatial geometry where unlike spatial bag models, the spatial locality of the points in the problem is important

We intend to study deep learning based time evolution operators for different complex systems and identify patterns as to which type of network describes which problem classes and the amount of data needed to get good results. Hopefully, we will also make research advances in the best networks to use; this is already seen in the move from recurrent networks to transformer and reformer architectures but this was largely motivated by sequence to sequence mapping and not by time series. We suggest more research in multiple or hierarchical time scales as this is needed in many applications.

We intend to build a benchmark set of time series datasets and reference implementations as playing the same role for time series that ImageNet ILSRVC and AlexNet played for images. The different implementations establish best practice or get chosen for different application areas to either suggest an architecture or an initial network by transfer learning. Interesting complex systems that we can quickly look at, include virtual tissues [41], [42] and epidemiology[43] for Covid related applications. Such evolution operators are also seen[3] in finance, networking, security, monitoring of complex systems from Tokamaks[44] to operating systems, and environmental science.

MLPerf benchmarks aim to quantitatively study the highest performance hardware and software systems. However, they also serve as examples of best practice and can help advance a field by documenting best practices. We intend to combine the open datasets and clean reference implementations available in MLPerf with documentation and tutorials which will allow MLPerf benchmarks to encourage the broad community to study these examples and use the ideas in other applications as well improving on our base reference implementations [45]. This work will be performed in the MLPerf Science Data which was just set up and is led by Fox and Hey (Chief Data Scientist at the Rutherford Appleton Laboratory). We will build multiple time series of the “MLPerf tutorial style” starting with initial projects from Indiana and identifying other examples from either the working group compilation [3] or identified in the meetings of MLPerf working groups. We will also look at areas including anomaly/failure detections, device metrics analysis, troubleshooting, and many IoT related data streams; examples have been compiled in cybersecurity and industrial operation categories by MLPerf [3].

**Acknowledgements**
This work is partially supported by the National Science Foundation (NSF) through awards CIF21 DIBBS 1443054, nanoBIO 1720625, CINES 1835598 and Global Pervasive Computational Epidemiology 1918626. I thank Lijiang Guo, Gregor von Laszewski, Saumyadipta Pyne, Xinyuan Huang, Bo Feng, Russell Hofmann, JCS Kadupitiya, and Vikram Jadhao for great discussions and Niranda Perera for preparing the Hydrology dataset. I am grateful to Cisco University Research Program Fund grant 2020-220491 for supporting this research.

**Software**
A recent FFFFWNPF Google Colab notebook can be found at https://colab.research.google.com/drive/1dUwrxlUq8G8HHTzImJXmJ9qiUFabdIKf?usp=sharing. This defines hyperparameters and networks and is open source. The name FFFFWNPF remembers the name of the optimization package that I developed over 50 years ago and used in physics data analysis such as [46]. FFFF stands for Fee-fi-fo-fum [47] and WNPF is just Well Nigh Perfect Fit. The original FFFWNPF suffered the fate of computer cards and retired electronic stores past their glory days [48] at LBNL. This fate was entirely my fault as I was

thinking about other things; FFFWNPF was very grateful for example for the extensive work done on the CDC 6600 and 7600's at LBNL.

## References

[1] P. Mattson, C. Cheng, G. Diamos, C. Coleman, P. Micikevicius, D. Patterson, H. Tang, G.-Y. Wei, P. Bailis, V. Bittorf, D. Brooks, D. Chen, D. Dutta, U. Gupta, K. Hazelwood, *et al.*, "MLPerf Training Benchmark," in *Proceedings of Machine Learning and Systems 2020*, 2020, pp. 336–349.

[2] "MLPERF benchmark suite for measuring performance of ML software frameworks, ML hardware accelerators, and ML cloud platforms." [Online]. Available: https://mlperf.org/. [Accessed: 08-Feb-2019]

[3] X. Huang, G. C. Fox, S. Serebryakov, A. Mohan, P. Morkisz, and D. Dutta, "Benchmarking Deep Learning for Time Series: Challenges and Directions," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 5679–5682 [Online]. Available: http://dx.doi.org/10.1109/BigData47090.2019.9005496

[4] Yan Liu, "Artificial Intelligence for Smart Transportation Video." [Online]. Available: https://slideslive.com/38917699/artificial-intelligence-for-smart-transportation. [Accessed: 08-Aug-2019]

[5] Geoffrey Fox, "Class Presentation: Studying Time Series with Deep Learning." [Online]. Available: https://docs.google.com/presentation/d/1ddAsHq-uikjPnomnSE21E0R0y7xBIrLGvaP_b6CM0dw/edit?usp=sharing. [Accessed: 13-Nov-2020]

[6] Geoffrey Fox, "Video of Studying Time Series for Deep Learning." [Online]. Available: https://youtu.be/teXeAdX6_Cg. [Accessed: 13 November, 2020]

[7] Geoffrey Fox, "Deep Learning Based Time Evolution." [Online]. Available: http://dsc.soic.indiana.edu/publications/Summary-DeepLearningBasedTimeEvolution.pdf. [Accessed: 08-Jun-2020]

[8] C. Widanage, N. Perera, V. Abeykoon, S. Kamburugamuve, T. A. Kanewala, H. Maithree, P. Wickramasinghe, A. Uyar, G. Gunduz, and G. Fox, "High Performance Data Engineering Everywhere," *arXiv [cs.DC]*, 19-Jul-2020 [Online]. Available: http://arxiv.org/abs/2007.09589

[9] Vibhatha Abeykoon, Niranda Perera, Chathura Widanage, Supun Kamburugamuve, Thejaka Amila Kanewala, Hasara Maithree, Pulasthi Wickramasinghe, Ahmet Uyar, Geoffrey Fox, "Data Engineering for HPC with Python," in *SC20 Proceedings*, Atlanta [Online]. Available: https://www.researchgate.net/profile/Geoffrey_Fox/publication/344211401_Data_Engineering_for_HPC_with_Python/links/5f5c2f3892851c07895fdbce/Data-Engineering-for-HPC-with-Python.pdf

[10] Geoffrey Fox, "FFFFWNPF Time Series Deep Learning Jupyter Notebook." [Online]. Available: https://colab.research.google.com/drive/1dUwrxlUq8G8HHTzImJXmJ9qiUFabdIKf?usp=sharing. [Accessed: 14-Nov-2020]

[11] Copyright 2019 The TensorFlow Authors., "Transformer model for language understanding: Google Colab Tutorial." [Online]. Available: https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/text/transformer.ipynb. [Accessed: 13-Nov-2020]

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," *arXiv [cs.CL]*, 12-Jun-2017 [Online]. Available: http://arxiv.org/abs/1706.03762

[13] Geoffrey C. Fox, Gregor von Laszewski, Fugang Wang, Saumyadipta Pyne, "AICov: An Integrative Deep Learning Framework for COVID-19 Forecasting with Population Covariates," Arxiv 2010.03757, Jul. 2020 [Online]. Available: http://dsc.soic.indiana.edu/publications/paper_covid.pdf, https://arxiv.org/abs/2010.03757

[14] Kratzert, Frederik, "CAMELS Extended Maurer Forcing Data." [Online]. Available: https://www.hydroshare.org/resource/17c896843cf940339c3c3496d0c1c077/. [Accessed: 14-Jul-2020]

[15] N. Addor, A. J. Newman, N. Mizukami, and M. P. Clark, "The CAMELS data set:Catchment attributes

and meteorology for large-sample studies," *Hydrol. Earth Syst. Sci.*, vol. 21, no. 10, p. 21, Oct. 2017 [Online]. Available: https://ueaeprints.uea.ac.uk/id/eprint/65434/. [Accessed: 05-Jul-2020]

[16] Frederik Kratzert, "Catchment-Aware LSTMs for Regional Rainfall-Runoff Modeling GitHub." [Online]. Available: https://github.com/kratzert/ealstm_regional_modeling. [Accessed: 14-Jul-2020]

[17] A. J. Newman, M. P. Clark, K. Sampson, A. Wood, L. E. Hay, A. Bock, R. J. Viger, D. Blodgett, L. Brekke, J. R. Arnold, and Others, "Development of a large-sample watershed-scale hydrometeorological data set for the contiguous USA: data set characteristics and assessment of regional variability in hydrologic model performance," *Hydrol. Earth Syst. Sci.*, vol. 19, no. 1, p. 209, 2015 [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.668.2612&rep=rep1&type=pdf

[18] F. Kratzert, D. Klotz, G. Shalev, G. Klambauer, S. Hochreiter, and G. Nearing, "Towards learning universal, regional, and local hydrological behaviors via machine learning applied to large-sample datasets," *Hydrology & Earth System Sciences*, vol. 23, no. 12, 2019 [Online]. Available: https://arxiv.org/abs/1907.08456

[19] JCS Kadupitiya, Geoffrey C. Fox, Vikram Jadhao, "GitHub repository for Simulating Molecular Dynamics with Large Timesteps using Recurrent Neural Networks." [Online]. Available: https://github.com/softmaterialslab/RNN-MD. [Accessed: 01-May-2020]

[20] J. C. S. Kadupitiya, G. C. Fox, and V. Jadhao, "Simulating Molecular Dynamics with Large Timesteps using Recurrent Neural Networks," *arXiv [physics.comp-ph]*, 12-Apr-2020 [Online]. Available: http://arxiv.org/abs/2004.06493

[21] J. C. S. Kadupitiya, G. Fox, and V. Jadhao, "Recurrent Neural Networks Based Integrators for Molecular Dynamics Simulations," in *APS March Meeting 2020*, 2020 [Online]. Available: http://meetings.aps.org/Meeting/MAR20/Session/L45.2. [Accessed: 23-Feb-2020]

[22] Robert Marsland and Pankaj Mehta, "Data-driven modeling reveals a universal dynamic underlying the COVID-19 pandemic under social distancing," *arXiv [q-bio.PE]*, 21-Apr-2020 [Online]. Available: http://arxiv.org/abs/2004.10666

[23] Luca Magri and Nguyen Anh Khoa Doan, "First-principles Machine Learning for COVID-19 Modeling," *Siam News*, vol. 53, no. 5, Jun. 2020 [Online]. Available: https://sinews.siam.org/Details-Page/first-principles-machine-learning-for-covid-19-modeling

[24] A. Adiga, L. Wang, A. Sadilek, A. Tendulkar, S. Venkatramanan, A. Vullikanti, G. Aggarwal, A. Talekar, X. Ben, J. Chen, B. Lewis, S. Swarup, M. Tambe, and M. Marathe, "Interplay of global multi-scale human mobility, social distancing, government interventions, and COVID-19 dynamics," *medRxiv - Public and Global Health*, 07-Jun-2020 [Online]. Available: http://dx.doi.org/10.1101/2020.06.05.20123760

[25] A. Galassi, M. Lippi, and P. Torroni, "Attention in Natural Language Processing," *arXiv [cs.CL]*, 04-Feb-2019 [Online]. Available: http://arxiv.org/abs/1902.02181

[26] D. A. Kaji, J. R. Zech, J. S. Kim, S. K. Cho, N. S. Dangayach, A. B. Costa, and E. K. Oermann, "An attention based deep learning model of clinical events in the intensive care unit," *PLoS One*, vol. 14, no. 2, p. e0211057, Feb. 2019 [Online]. Available: http://dx.doi.org/10.1371/journal.pone.0211057

[27] T. Gangopadhyay, S. Y. Tan, Z. Jiang, R. Meng, and S. Sarkar, "Spatiotemporal Attention for Multivariate Time Series Prediction and Interpretation," *arXiv [cs.LG]*, 11-Aug-2020 [Online]. Available: http://arxiv.org/abs/2008.04882

[28] N. Xu, Y. Shen, and Y. Zhu, "Attention-Based Hierarchical Recurrent Neural Network for Phenotype Classification," in *Advances in Knowledge Discovery and Data Mining*, 2019, pp. 465–476 [Online]. Available: http://dx.doi.org/10.1007/978-3-030-16148-4_36

[29] R. S. Kodialam, R. Boiarsky, and D. Sontag, "Deep Contextual Clinical Prediction with Reverse Distillation," *arXiv [cs.LG]*, 10-Jul-2020 [Online]. Available: http://arxiv.org/abs/2007.05611

[30] J. Gao, X. Wang, Y. Wang, Z. Yang, J. Gao, J. Wang, W. Tang, and X. Xie, "CAMP: Co-Attention Memory Networks for Diagnosis Prediction in Healthcare," in *2019 IEEE International Conference on Data Mining (ICDM)*, 2019, pp. 1036–1041 [Online]. Available: http://dx.doi.org/10.1109/ICDM.2019.00120

[31] R. Sen, H.-F. Yu, and I. Dhillon, "Think Globally, Act Locally: A Deep Neural Network Approach to High-Dimensional Time Series Forecasting," *arXiv [stat.ML]*, 09-May-2019 [Online]. Available: http://arxiv.org/abs/1905.03806

[32] H. Song, D. Rajan, J. J. Thiagarajan, and A. Spanias, “Attend and diagnose: Clinical time series analysis using attention models,” in *Thirty-second AAAI conference on artificial intelligence*, 2018 [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/viewPaper/16325

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. U. Kaiser, and I. Polosukhin, “Attention is All you Need,” in *Advances in Neural Information Processing Systems*, 2017, vol. 30, pp. 5998–6008 [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[34] A. Zeyer, P. Bahar, K. Irie, R. Schlüter, and H. Ney, “A Comparison of Transformer and LSTM Encoder Decoder Models for ASR,” in *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2019, pp. 8–15 [Online]. Available: http://dx.doi.org/10.1109/ASRU46091.2019.9004025

[35] Z. Zeng, V. T. Pham, H. Xu, Y. Khassanov, E. S. Chng, C. Ni, and B. Ma, “Leveraging Text Data Using Hybrid Transformer-LSTM Based End-to-End ASR in Transfer Learning,” *arXiv [eess.AS]*, 21-May-2020 [Online]. Available: http://arxiv.org/abs/2005.10407

[36] Chaopeng Shen, Penn State University, “D2 2020 AI4ESS Summer School: Recurrent Neural Networks and LSTMs.” [Online]. Available: https://www.youtube.com/watch?v=vz11tUgoDZc. [Accessed: 01-Jul-2020]

[37] M. A. Sit, B. Z. Demiray, Z. Xiang, G. Ewing, Y. Sermet, and I. Demir, “A Comprehensive Review of Deep Learning Applications in Hydrology and Water Resources,” 2020 [Online]. Available: https://eartharxiv.org/xs36g/

[38] Fugang Wang, “LATEST Short demo-LSTM-Tutorial.ipynb Hydrology on CAMELS Notebook.” [Online]. Available: https://colab.research.google.com/drive/1MePoMs1mNmsiPMxUi2RxExgVQKCzhM17?usp=sharing. [Accessed: August 1,. 2020]

[39] Geoffrey Fox, “Hydrology AIHEC REU Page for summer research 2020.” [Online]. Available: https://docs.google.com/document/d/1gC7jyhfGjYihZhn8AF-4qhquJlQYUd97sqXrr7FPZfs/edit?usp=sharing. [Accessed: 01-Jul-2020]

[40] NCAR, “CAMELS: CATCHMENT ATTRIBUTES AND METEOROLOGY FOR LARGE-SAMPLE STUDIES - DATASET DOWNLOADS.” [Online]. Available: https://ral.ucar.edu/solutions/products/camels. [Accessed: 17-Nov-2020]

[41] T. J. Sego, J. O. Aponte-Serrano, J. F. Gianlupi, S. Heaps, K. Breithaupt, L. Brusch, J. M. Osborne, E. M. Quardokus, and J. A. Glazier, “A Modular Framework for Multiscale Spatial Modeling of Viral Infection and Immune Response in Epithelial Tissue,” *BioRxiv*, 2020 [Online]. Available: https://www.biorxiv.org/content/10.1101/2020.04.27.064139v2.abstract

[42] Yafei Wang, Gary An, Andrew Becker, Chase Cockrell, Nicholson Collier, Morgan Craig, Courtney L. Davis, James Faeder, Ashlee N. Ford Versypt, Juliano F. Gianlupi, James A. Glazier, Randy Heiland, Thomas Hillen, Mohammad Aminul Islam, Adrianne Jenner, Bing Liu, Penelope A Morel, Aarthi Narayanan, Jonathan Ozik, Padmini Rangamani, Jason Edward Shoemaker, Amber M. Smith, Paul Macklin, “Rapid community-driven development of a SARS-CoV-2 tissue simulator,” *BioRxiv*, 2020 [Online]. Available: https://www.biorxiv.org/content/10.1101/2020.04.02.019075v2.abstract

[43] D. Machi, P. Bhattacharya, S. Hoops, J. Chen, H. Mortveit, S. Venkatramanan, B. Lewis, M. Wilson, A. Fadikar, T. Maiden, C. L. Barrett, and M. V. Marathe, “Scalable Epidemiological Workflows to Support COVID-19 Planning and Response,” May 2020.

[44] J. Kates-Harbeck, A. Svyatkovskiy, and W. Tang, “Predicting disruptive instabilities in controlled fusion plasmas through deep learning,” *Nature*, vol. 568, no. 7753, pp. 526–531, Apr. 2019 [Online]. Available: https://doi.org/10.1038/s41586-019-1116-4

[45] G. Fox, “SciDatBench AI for Science Benchmark Activity working with MLPerf.” [Online]. Available: https://github.com/DSC-SPIDAL/SciDatBench/. [Accessed: 01-Aug-2020]

[46] Geoffrey Fox, “Veni, Vidi, Vici Regge Theory; Comments Nucl.Part.Phys. 3 (1969) 190-197. (journal disappeared).” [Online]. Available: https://www.dsc.soic.indiana.edu/sites/default/files/Veni%2C%20Vidi%2C%20Vici%20Regge%20Theory.pdf. [Accessed: 17-Nov-2020]

[47] “Wikipedia Fee-fi-fo-fum.” [Online]. Available: https://en.wikipedia.org/wiki/Fee-fi-fo-fum. [Accessed: 17-Nov-2020]

[48] “IBM 1360 Chip Store.” [Online]. Available: https://en.wikipedia.org/wiki/IBM_1360