Fire Technology, 59, 3327–3354, 2023

https://doi.org/10.1007/s10694-023-01469-6

Recurrent Convolutional Deep Neural Networks for Modeling Time-Resolved Wildfire Spread Behavior

*John Burge**, *Google Research, Mountain View, CA 94043, USA*
Matthew R. Bonanni, Department of Mechanical Engineering, Stanford University, Stanford, CA 94305, USA
R. Lily Hu, Google Research, Mountain View, CA 94043, USA
Matthias Ihme, Google Research, Mountain View, CA 94043, USA; Department of Mechanical Engineering, Stanford University, Stanford, CA 94305, USA; Department of Photon Science, SLAC National Accelerator Laboratory, Menlo Park, CA 94025, USA

Received: 18 October 2022/**Accepted:** 19 July 2023/**Published online:** 7 August 2023

Abstract. The increasing incidence and severity of wildfires underscores the necessity of accurately predicting their behavior. While high-fidelity models derived from first principles offer physical accuracy, they are too computationally expensive for use in real-time fire response. Low-fidelity models sacrifice some physical accuracy and generalizability via the integration of empirical measurements, but enable real-time simulations for operational use in fire response. Machine learning techniques have demonstrated the ability to bridge these objectives by learning first-principles physics while achieving computational speedups. While deep learning approaches have demonstrated the ability to predict wildfire propagation over large time periods, time-resolved fire-spread predictions are needed for active fire management. In this work, we evaluate the ability of deep learning approaches in accurately modeling the time-resolved dynamics of wildfires. We use an autoregressive process in which a convolutional recurrent deep learning model makes predictions that propagate a wildfire over 15 min increments. We apply the model to four simulated datasets of increasing complexity, containing both field fires with homogeneous fuel distribution as well as real-world topologies sampled from the California region of the United States. We show that even after 100 autoregressive predictions representing more than 24 h of simulated fire spread, the resulting models generate stable and realistic propagation dynamics, achieving a Jaccard score between 0.89 and 0.94 when predicting the resulting fire scar. The inference time of the deep learning models are examined and compared, and directions for future work are discussed.

Keywords: Wildland fires, Machine learning, Deep neural networks, Recurrent neural networks

*Correspondence should be addressed to: John Burge, E-mail: lawnguy@google.com

1. Introduction

In recent decades, climate change and excessive fire suppression have resulted in an increase in both the size and severity of wildfires [1–3]. In addition to the impact of wildfires on property damage and loss of life [4], they also have long-term effects on both health [5] and the environment [6]. Understanding and modeling wildfire spread is critical to informing a direct response to ongoing fires and is an important tool for evaluating fire risk.

Computational models predicting wildfire behavior for fire management, risk assessment, and wildfire mitigation have seen significant advancements [7]. These models can be categorized as physical, empirical, and mathematical analogues [8–10]. Physical models are built from first principles and provide a high level of fidelity in predicting the fire behavior by solving conservation equations to capture relevant physical processes. Notable examples of these include FIRETEC [11], models developed by IUSTI [12], and WFDS [13]. The computational complexity of these models currently prevents their application to real-time simulations, so that they are often used in augmenting field experimentation or in the detailed analysis of wildfire dynamics [8].

In contrast, purely empirical models rely solely on data, making no assumptions about fire behavior based on theory [9]. Quasi-empirical models combine observations from real fires and laboratory experiments with knowledge of the underlying combustion and heat-transfer processes. One such model is Rothermel's formulation [14], which serves as cornerstone for several fire prediction methods used in the United States, such as BEHAVE [15] and FARSITE [16]. While these models offer real-time simulations, enabling their use in operational settings, their development typically relies on a manual process of model calibration, based on current understanding of wildland-fire spread.

In the past decade, the potential of machine learning (ML) methods for application to wildfires has been recognized, with ML techniques used across tasks such as: fuel characterization, risk assessment, fire behavior modeling, and fire management [17, 18]. While there are various approaches within the ML community, the sub-field of deep learning and the development of deep neural networks (DNNs) have led to breakthroughs in many domains [19]. However, only a few studies have examined the utility of deep learning methods for predicting the dynamics of wildfire propagation [20–23]. Hodges and Lattimer [20] demonstrated that a specific DNN, the Deep Converse Inverse Graphical Model (DCIGN) [24], could effectively predict the state of a fire six hours into the future in a single time step.

They used FARSITE to generate a database to train the model, and this single 6-h prediction was computed orders of magnitude faster than explicitly time-advancing the FARSITE simulation. This was possible because DNN models do not rely on time-step limitations present in many other simulations, e.g., the CFL constraints made in PDE-based methods, or the low-order estimation of the fire's rate of spread made in FARSITE. Further demonstrating the flexibility of DNNs, Radke et al. [21] performed a similar study by modeling an even longer duration: the 24-h evolution of fire propagating over the geography of the Rocky Mountain region of the United States using real-world data instead of simulated data.

While performing a single 6- or 24-h prediction with a DNN was shown to be much faster than running FARSITE over the required number of simulation steps, the lack of temporally resolving the wildfire dynamics limits the application for active fire interventions and fire management and may ultimately constrain predictive efficacy. By addressing this issue, our objective is to examine the utility of DNNs for modeling the wildfire propagation dynamics on fine temporal scales with time increments of minutes. To provide predictions over larger time frames, we use an autoregressive process where the output for the prediction at time τ is used as input for the prediction at time $\tau + \Delta\tau$, where $\Delta\tau \in \{15, 30, 60\}$ minutes.

We show that when dealing with temporal dynamics, the DCIGN model is not capable of accurately replicating propagation beyond, at most, ten autoregressive predictions before failing to make any realistic predictions at all. To address this issue, we employ an Encoder-Processor-Decoder (EPD) model [25, 26] to represent the spatial relationships in the data and add recurrent transformations to represent temporal relationships that have not been considered in prior work.

The remainder of this paper has the following structure. In Sect. 2, we describe the generation of our training and evaluation datasets. Our model, the EPD-ConvLSTM model, is summarized in Sect. 3, and the process for training and evaluating the models are presented in Sect. 4. Results are discussed in Sect. 5 and the manuscript closes in Sect. 6 with conclusions and potential future research directions.

2. Generation of Dataset

All DNN models were trained and evaluated using synthetic datasets created from FARSITE [16]. A total of 40,000 fire simulations were generated. Each of these used a domain that was discretized by 128×128 cells with a cell size of 30 m. This resolution was selected as it was the highest resolution available from the LANDFIRE dataset based on Landsat imagery [27], which was used to collect real-world patches of terrain data. The domain size was selected to facilitate large-scale, heterogeneous landscapes while remaining computationally tractable.

The ranges for parameters and operating conditions used to generate the dataset are summarized in Table 1. All parameters are held constant over time. Fuel types across all four datasets were chosen from the dynamic fuel models defined by Scott and Burgan [28], without considering ground fuels. The method introduced by Finney [16] was applied for calculating the crown fire potential, with a specified foliar moisture content of 100%. Spotting was not considered. The starting location for each fire was randomly selected to be within the central 50% of the field. An octagonal fire front with a width of approximately 75 m ($\sim$ 2% of the width of the field) was then placed at this location, and allowed to propagate. The duration of each simulation was held constant at 72 h with a time step size of 15 min, yielding sequences with 289 time steps. Note that the fire dynamics were not constrained by this time frame. Some fires burned quickly through the patch in far fewer steps, and some fires burned more slowly and never fully burnt out in the simulation.

Table 1
Parameter Ranges Used for Generating Synthetic Datasets from FARSITE

Parameter [units]	Range
Slope [°]	[0, 45]
Aspect [°]	[0, 360]
Wind direction [°]	[0, 360]
Wind velocity [km/h]	[0, 50]
Fuel model [#]	[1, 40]
1-h moisture [%]	[2, 40]
10-h moisture [%]	[2, 40]
100-h moisture [%]	[2, 40]
Live herbaceous moisture [%]	[30, 100]
Live woody moisture [%]	[30, 100]
Canopy cover [%]	[0, 100]
Canopy height [m]	[3, 50]
Crown ratio [–]	[0.1, 1]
Canopy bulk density [kg/m^3]	[0, 4000]

The effects of varying solar radiation due to the earth's rotation were not considered, with the latitude, longitude, and elevation of each landscape set identically for each simulation to 0°, 0°, and 0 m, respectively. The sun's position in the sky was held constant at 90° of elevation and 0° of azimuth. Finally, in order to eliminate boundary effects within FARSITE, one cell was trimmed from the border, resulting in a final field size of 126×126 cells.

The output of each simulation is a series of data points for the position of the fire front as it advances over time. To transform the results into gridded data for training DNN models, the burned area for each cell was computed as the fraction of the burned region inside each cell.

From these simulations, a dataset was constructed with 17 channels, where each channel represents a specific parameter. The state of the fire at a fixed point in time is represented by a three dimensional tensor: $X \in \mathbb{R}^{H \times W \times C}$, where H is the height, W is the width, and C is the number of channels; here $H = W = 126$ and $C = 17$. The first three channels were derived from the burn fraction described above: (1) `vegetation`, the fraction of vegetation that remains unburnt, (2) `fire_front`, the fraction of vegetation that burned in the previous time step, and (3) `scar`, the fraction of vegetation that is currently burnt. The remaining 14 channels were taken directly from the following field variables tracked in FARSITE [16]: (4) `wind_east`, (5) `wind_north`, (6) `moisture_1_hour`, (7) `moisture_10_hour`, (8) `moisture_100_hour`, (9) `moisture_live_herbaceous`, (10) `moisture_live_woody`, (11) `cover`, (12) `height`, (13) `base`, (14) `density`, (15) `slope_east`, (16) `slope_north`, and (17) `fuel`. All channels contain continuous data, with the sole exception of the fuel channel, which contains a discrete index, ranging between 1 and 40, that specifies the fuel model.

Four datasets with varying complexity and characteristics were designed to evaluate specific components of the ML models.

- *single fuel*—Every simulation used the same fuel model across the entire landscape: GR1, or "short, sparse dry climate grass". All other parameters were held uniform across a given landscape, but vary randomly across individual simulations, and were selected from a uniform distribution within the ranges described in Table 1. The terrain is planar, with a constant slope and aspect. This dataset was designed to evaluate the DNN models' ability to respond to each of the changing parameters without spatial variation.
- *multiple fuel*—Generated identically to *single fuel*, except that for each fire sequence, a random fuel model is used for the entire domain. This seemingly minor deviation from the *single fuel* dataset results in approximately 40 times less coverage of the possible combinations of wind magnitudes, terrain slopes and fuel types, which are the three primary drivers of propagation dynamics. Therefore, this dataset is designed to test the performance of the DNN models when dealing with far less training data per fuel type than before.
- *California*—Real-world Landscape (LCP) 40 data was obtained from the LANDFIRE program [27] for the continental United States, which was then bounded by the north–south and east–west extents of the state of California and includes a significant portion of Nevada as well. For each simulation, a random (128×128) field was selected from the region. If the field did not consist of at least 70% burnable fuel, it was resampled. Wind data was randomly generated in the same manner as the *single fuel* and *multiple fuel* datasets.
- *California-WN*—Generated identically to the *California* dataset, but with spatially-varying wind. WindNinja [29] was used to compute a realistic wind pattern based on the prevailing wind vector and the topography of the landscape. While this spatial variation was used in the FARSITE simulations, only the prevailing wind vector was provided as an input to the DNN models, reflecting the sparse availability of such data in real-world applications.

A representative sample from each of the four datasets is shown in Figure 1. In the sample taken from the *single fuel* dataset, note the expected elliptical fire spread pattern, where the direction is influenced by both the slope of the terrain and the presence of the wind. In the *multiple fuel* dataset, while the fuel is still homogeneous, a different fuel has been selected when compared to the first sample. Despite the significantly higher wind speed, the fire has a much faster upwind spread rate, demonstrating the significantly different properties of this fuel. The *California* dataset demonstrates complex spread patterns over its heterogeneous landscape. The fire front is seen responding to different fuels by spreading at different rates, and is even blocked completely in some regions, including a road crossing the terrain in the southwest corner. The primary direction of spread is influenced by the wind, while there are also modulations associated with the terrain, where steep downward slopes impede the rate of spread. Finally, in the *California-WN* dataset, there are significant spatial variations in the wind field, especially in the valley running vertically across the landscape. Here, the wind is

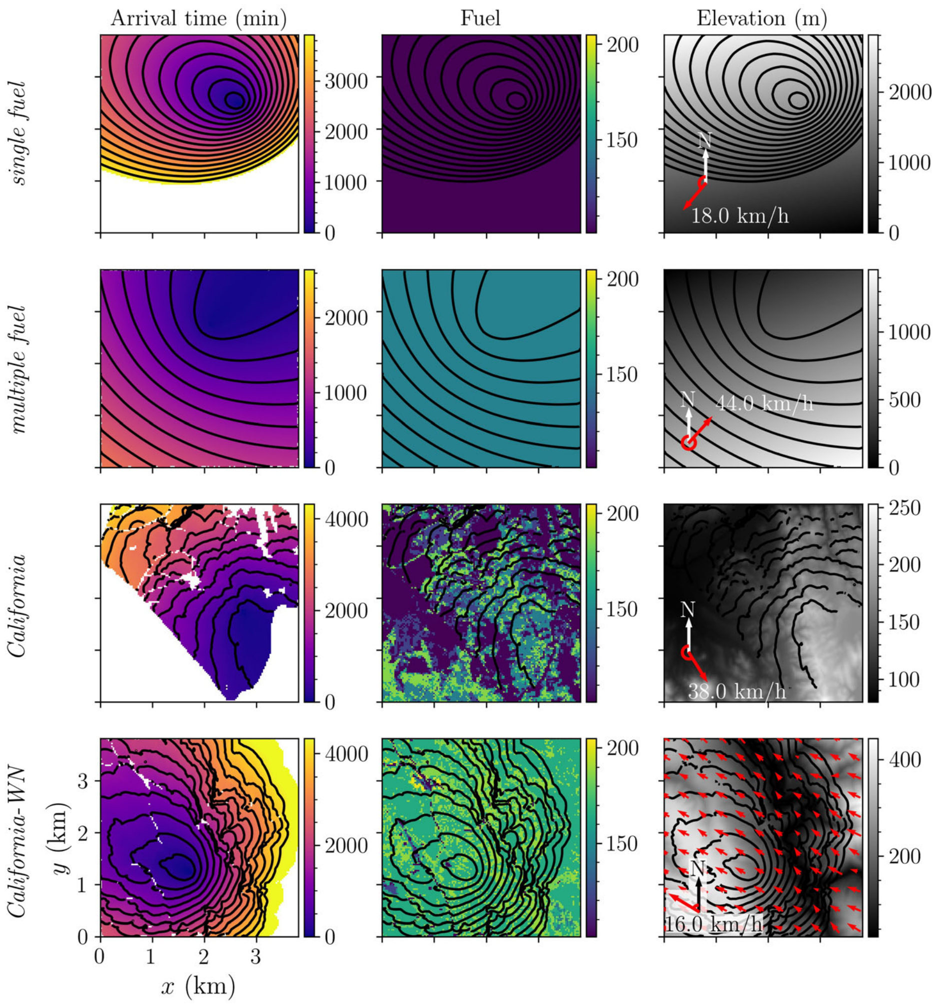


Figure 1. Sample results from the four datasets used for training the DNN models. Isocontours of arrival time are overlaid in *black*. In the *California-WN* dataset, the prevailing wind direction is indicated in the *lower-left corner* as in the other datasets, while the local speed and direction variation is indicated by the vector field.

significantly attenuated and redirected along the valley, and the fire spread is impacted both by this adverse gradient and the resulting wind variation.

These simulations were performed in parallel on a high-performance computing cluster, with each simulation taking an average of 30 s to run single-threaded on 1 CPU, for a total cost of 330 CPU-hours, or 12,000 core-hours on 36-core processors, though this could be significantly accelerated via multithreading. The post-processing of the simulations to compute the fractional burned area of each cell as described above consumed an additional 17,000 core-hours, for a total of approximately 29,000 core-hours.

3. Machine-Learning Models

In the present work, we use DNNs to model fire-spread behavior. A detailed discussion of DNNs is beyond the scope of this paper, but for readers unfamiliar with the commonly employed transformations used in our DNNs, we provide a summary in Appendix 1. For more detail, we refer the interested reader to the text by Foster [30].

In the remainder of this paper, we describe the structure of the DCIGN model, the EPD model and the EPD-ConvLSTM model. We represent the input to any of the models as X. The DCIGN and EPD models require input that contains a single point in time, so for them: $X \in \mathbb{R}^{B \times H \times W \times C}$. The EPD-ConvLSTM model requires temporal input, thus $X \in \mathbb{R}^{B \times T \times H \times W \times C}$, where T is the length of the time series.

3.1. DCIGN Model

Many DNNs can be broken up into two distinct stages. First, an *encoder* transforms the input into a latent space and second, a *decoder* transforms the latent space into the desired output. The DCIGN model used in prior wildland spread models [20, 21] is an instance of such a model, Figure 2. The encoder starts by adding some zero-padding to get the spatial dimensions to be powers of 2. Then, a repeated set of convolutional transformations and downsampling steps are applied. The downsampling stages help the DCIGN model make predictions that can efficiently make large structural changes to the current state of the fire. This ultimately transforms the input of shape $(B, H = 126, W = 126, C = 17)$ into the latent space of shape $(B, H = 8, W = 8, C = 64)$. In essence, the convolutional transformations have moved the input with 17 distinct channels into a new latent space that has 64 distinct channels but a much smaller spatial resolution. The decoder transforms this latent space by using a fully connected layer into 15,876 scalar values, which are subsequently reshaped into a $(B, H = 126, W = 126)$ field that is the output of the DNN model.

The hyperparameters for the model were selected to be as close to the work of Hodges and Lattimer [20] as possible. Thus, the convolutional kernels had the size (10,10) and the number of filters for the five repeated convolutions were set to: 8, 8, 16, 32 and 64 (i.e., these are the values for C^r in the shape of the convolutions in Figure 2). The ideal learning rate was empirically found to be 10^{-4}.

3.2. EPD Model

Figure 3 provides the structure of the EPD model [25]. The EPD's encoder first uses an embedding layer to convert the discrete fuel channel into a continuous embedding space (highlighted in orange). Discrete channels can be challenging for DNNs to incorporate directly since small changes in the discrete value can lead to

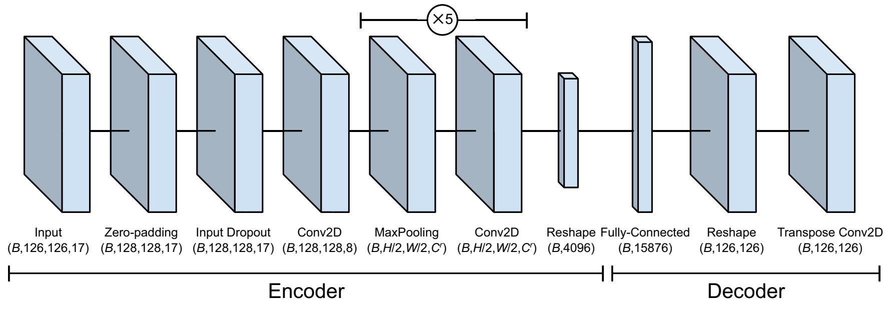

×5
Input (B,126,126,17)
Zero-padding (B,128,128,17)
Input Dropout (B,128,128,17)
Conv2D (B,128,128,8)
MaxPooling (B,H/2,W/2,C′)
Conv2D (B,H/2,W/2,C′)
Reshape (B,4096)
Fully-Connected (B,15876)
Reshape (B,126,126)
Transpose Conv2D (B,126,126)
Encoder
Decoder

Figure 2. Schematic of DCIGN architecture employed in the present work. *Parenthesis* provide the shape of the data produced by the corresponding layer in the model.

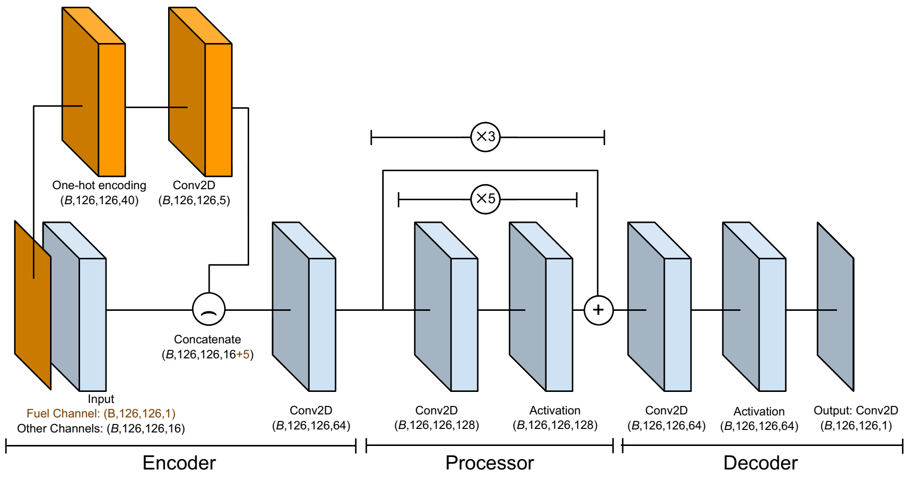

One-hot encoding (B,126,126,40)
Conv2D (B,126,126,5)
×3
×5
Concatenate (B,126,126,16+5)
Input
Fuel Channel: (B,126,126,1)
Other Channels: (B,126,126,16)
Conv2D (B,126,126,64)
Conv2D (B,126,126,128)
Activation (B,126,126,128)
Conv2D (B,126,126,64)
Activation (B,126,126,64)
Output: Conv2D (B,126,126,1)
Encoder
Processor
Decoder

Figure 3. Schematic of EPD architecture employed in the present work. The path taken by the discretely-valued fuel channel is highlighted in *orange* (Color figure online).

large changes in the underlying dynamics. The embedding layer transforms the discrete fuel values into a higher dimensional latent space such that similarly behaving fuel types are close to each other in the embedding space, even if the discrete values are far apart. This allows the information within the fuel index to potentially be more efficiently considered by subsequent layers of the DNN.

The output of the embedding layer is concatenated with the other 16 continuous input channels, and fed into a single convolutional transformation. A total of 15 convolutional transformations are sequentially applied in the processor portion of the model. Long sequences of transformations can be difficult to train, so skip links are added around every block of five sequential convolution layers [31]. The EPD model finishes with a few convolutional transformations as its decoder.

The main differentiators between the DCIGN model and the EPD model are the removal of the fully-connected layers, the removal of the Max Pooling down-sampling stages and using substantially smaller convolutional kernels. These changes allow the EPD model to focus more specifically on local relationships between neighboring cells, potentially at the cost of not recognizing large-scale relationships.

The hyperparameters for the EPD model were empirically determined and include: the number of blocks of convolutions in the processor (3 blocks of 5 convolutions each), the number of filters in each convolution (encoder: 64, processor: 128, decoder: 64), the activation function (encoder: identity, processor: ReLU, decoder: ReLU), the size of the convolutional kernels (encoder: 5, processor: 3, decoder: 5), the training batch size (64) and the learning rate (10^{-4}).

3.3. *EPD-ConvLSTM Model*

While the EPD model effectively considers spatial relationships, it does not account for temporal relationships in the data. For those, we use ConvLSTM recurrent transformations. The ConvLSTM transformation iteratively processes each time step in the input that it receives, building up a *memory* as it does so, and outputs a final prediction that is directly dependent on both the input and that built up memory. The exact mechanism for doing this are outside the scope of this paper. See Appendix 1 for additional details and Foster [30] for further reading.

Like the EPD model, the output is still the location of the fire front at the next time step, but unlike the EPD model, the input is an entire time series. By adding the recurrent layers to the EPD model, we introduce the EPD-ConvLSTM model, Figure 4. Note that the visualization of some of the stages in Figure 4 are vertically split into eight slices. This indicates that that the input to these stages contains a time series. As such, the data shape for each stage in Figure 4 has five dimensions as opposed to the four dimensions in the shapes seen in Figure 3, with the additional dimension being time. The recurrent stages are highlighted in green and the vertical arrow indicates that the recurrent stage is repeated eight times (once for each of the time steps in the temporal dimension).

The input to the EPD-ConvLSTM model can be thought of as eight different inputs for the EPD model, concatenated together into a single data point. As seen in Figure 4, the EPD-ConvLSTM model uses the same encoder, processor and decoder as the EPD model. To facilitate that, the EPD-ConvLSTM model merges the batch and time dimensions together before passing the data to the EPD Encoder. The EPD Processor output is then split back up to restore the time dimension for the recurrent stages of the network.

For simplicity, we have presented the EPD-ConvLSTM model such that it makes all predictions completely independently. Thus, e.g., for predictions made at times τ and $\tau + \Delta\tau$, seven of the eight time steps are reprocessed when making the second prediction. This inefficiency can be addressed by storing the *memory* built up in the recurrent stages when making the first prediction, and using that

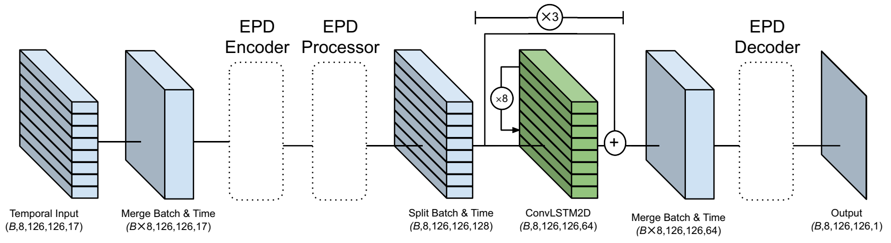

EPD Encoder
EPD Processor
×3
×8
EPD Decoder
Temporal Input (B,8,126,126,17)
Merge Batch & Time (B×8,126,126,17)
Split Batch & Time (B,8,126,126,128)
ConvLSTM2D (B,8,126,126,64)
Merge Batch & Time (B×8,126,126,64)
Output (B,8,126,126,1)

Figure 4. Schematic of EPD-ConvLSTM architecture employed in the present work. The EPD encoder, processor and decoder are the same ones provided in Figure 3. For steps that have a temporal dimension, this visualization breaks up the layer into eight stacked levels to correspond to the eight consecutive time steps in the data. The recurrent layers are highlighted in *green* (Color figure online).

memory in lieu of the overlapping seven time points, when making the second prediction.

The EPD-ConvLSTM model uses the same hyperparameters as the EPD model, with the following additional hyperparameters that were empirically determined: the number of ConvLSTM blocks to use (3), the number of sequential convolutions within each block (5) and the size of the time series (8), where the choice of the latter parameter was constrained by memory.

4. Evaluation Methods

In this section, we describe the evaluation process used to measure the efficacy of the DCIGN, EPD and EPD-ConvLSTM models.

4.1. Autoregressive Predictions and Post-processing

To assess the accuracy of the DNN models in predicting the dynamics of wildfire propagation, we measure the accuracy of each DNN model over a time period of 1500 min (100 autoregressive predictions). The prediction made at time τ is used to construct the input for the prediction made at time $\tau + \Delta\tau$. The state of the fire at time τ is denoted X_τ and the resulting prediction is denoted Y_τ. For the DCIGN and EPD models, X_τ has the shape $(B, H = 126, W = 126, C = 17)$. For the EPD-ConvLSTM model, X_τ has the shape $(B, T = 8, H = 126, W = 126, C = 17)$. As all three models make the exact same prediction, the fraction of a cell's fuel that will burn away at time τ, Y_τ always has the shape $(B, H = 126, W = 126, C = 1)$ for both the EPD and EPD-ConvLSTM models.

Y_τ and X_τ are used to construct $X_{\tau+\Delta\tau}$ in the autoregressive process. Recall from Sect. 2, that there are seventeen channels. Fourteen of these channels are constant across time, so when constructing $X_{\tau+\Delta\tau}$, they can simply be copied from X_τ. Three of the channels represent the state of fuel: the `vegetation`, `fire_front` and `scar` channels. We denote these channels by X^{veg}, X^{front} and X^{scar}, and thus: $X^{front}_{\tau+\Delta\tau} = Y_\tau$, $X^{veg}_{\tau+\Delta\tau} = X^{veg}_\tau - Y_\tau$ and $X^{scar}_{\tau+\Delta\tau} = X^{scar}_\tau + Y_\tau$.

After each prediction, negative results were clipped to 0.0 to prevent invalid predictions due to round-off errors. No other regularization or limiting operations were applied to the EPD or EPD-ConvLSTM models, though, as was also observed in Hodges and Lattimer [20], the DCIGN model's predictions needed to be regularized with a mean filter using a (3×3) kernel.

4.2. Additional Training Details

We follow common practice and split each dataset into training, validation, and testing sets with a ratio of 80:10:10 [18], resulting in a total of 8000 unique training fire sequences, 1000 validation fire sequences and 1000 testing fire sequences. Each model was trained only on data from a single dataset. Training convergence was defined to be the earliest point at which validation loss stopped improving for 24 h of clock time. L_1 and L_2 regularization and drop-out was considered to resolve minor training-loss overfitting that was observed, but always resulted in worse validation loss and ultimately found not to be helpful.

Each hyperparameter for a model was determined individually by training multiple models across a range of values, with all other hyperparameters set to a constant value. The model with the lowest validation loss was used to determine the value for the hyperparameter. Hyperparameter sweeps were done on the *single fuel* dataset, then used across all other datasets. The EPD and EPD-ConvLSTM models were trained four times independently on each dataset with the same set of best-performing hyperparameters, and the model with the lowest validation loss was selected to do a full evaluation on.

All DNN models were built in TensorFlow 2.0 [32], with eager mode enabled. Keras APIs [33] were used to build each of the DNN models. Models were trained on a single machine with 16 CPU cores and eight NVIDIA P100 GPUs. Training models on the *single fuel* dataset took approximately 4 days for the EPD model and 6 days for EPD-ConvLSTM model. Training on the *multiple fuel* and both *California* datasets took 8 days for the EPD model and 13 days for the EPD-ConvLSTM model. The Adam optimizer [34] was used.

4.3. Performance Metrics

To quantify the accuracy of the DNN model in predicting the transient fire dynamics, we consider different metrics. Mean squared error (*MSE*) is one of the most common metrics used to measure the regression efficacy of a prediction, Y, for data point, X, with label, $\widehat{Y}$:

$$MSE(Y, \widehat{Y}) = \frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W}(Y_{ij} - \widehat{Y}_{ij})^2. \tag{1}$$

We use MSE as the loss function during training, but for evaluating how well a model does over an entire dataset, D, we use root mean squared error (RMSE):

$$RMSE(D) = \left(\frac{1}{HW|D|} \sum_{Y,\widehat{Y}}^{D} \sum_{i=1}^{H} \sum_{j=1}^{W} (Y_{ij} - \widehat{Y}_{ij})^2 \right)^{\frac{1}{2}}. \tag{2}$$

Given that predicting the speed of the fire is often of utmost importance, we also measure the error in the size of the predicted fire front, regardless of the location, with the summed total error metric, *STE*:

$$STE(D) = \frac{1}{|D|} \sum_{Y,\widehat{Y}}^{D} \left| \sum_{i=1}^{H} \sum_{j=1}^{W} Y_{i,j} - \sum_{i=1}^{H} \sum_{j=1}^{W} \widehat{Y}_{i,j} \right| \tag{3}$$

The *STE* metric provides a measure for the number of cells worth of fire that the prediction was off by, regardless of whether the predicted fire was in the correct location. Thus, models that have low *STE* are predicting the correct rate of the fire growth whereas models that have low *RMSE* are predicting the correct location of the fire.

In addition to tracking regression statistics, we also track classification statistics by converting the regressed value into a binary classification. This is done by introducing a threshold, below which a cell is considered to be *not on fire* and above which a cell is said to be *on fire*. We use a threshold of 0.1 as a general indicator that the cell at this point is quite noticeably on fire though quantitative results are robust to reasonable changes in this threshold.

We use the Jaccard Similarity Coefficient (JSC) [35] to measure the quality of the classifications. JSC grows as the intersection between the prediction and the ground truth grows, resulting in a maximal score when the intersection and union are identical (for those that prefer the F1 score, the results are qualitatively similar). The Jaccard score can be defined with indicator variables. Let $I^{Y_{i,j}}$ be an indicator variable that is 1 when the value in cell (i, j) is greater than the classification threshold:

$$JSC(D) = \frac{\sum_{Y,\widehat{Y}}^{D} \cap(Y, \widehat{Y})}{\sum_{Y,\widehat{Y}}^{D} \cup(Y, \widehat{Y})} \tag{4}$$

$$\cap(Y, \widehat{Y}) = \sum_{i=1}^{H} \sum_{j=1}^{W} I^{Y_{ij}} \wedge I^{\widehat{Y_{ij}}} \tag{5}$$

$$\cup(Y, \widehat{Y}) = \sum_{i=1}^{H} \sum_{j=1}^{W} I^{Y_{ij}} \vee I^{\widehat{Y_{ij}}}. \tag{6}$$

JSC ranges between 0.0, in which the prediction and the ground truth do not overlap, and 1.0, in which the ground truth and prediction are perfectly aligned.

In addition to tracking the location of the fire *front* over time, the location of the resulting *scar* left by the fire is also tracked. All of the same statistics are evaluated for the fire scar.

4.4. Bootstrapping Confidence Intervals

When assessing the performance of the different DNN models on a given metric, it is important to determine whether an observed difference is real or coincidentally caused by the stochastic process of placing data in the testing, training and validation datasets. To account for this, we use bootstrapping [36] to estimate confidence intervals for each metric. For each of the four test datasets, we build 20 new resampled datasets. Each resampled dataset is constructed by randomly sampling fire sequences from the 1000 sequences in the original testing dataset, with replacement, until the bootstrapping dataset also contains 1000 fire sequences. Data points are then generated from the fire sequences in the resampled dataset, and the metric is computed over those data points. After this is repeated 20 times, there is a distribution of values for the metric. We compute 50% confidence intervals by taking the 0.25 and 0.75 quantiles of this distribution. 90% confidence intervals are computed by taking the 0.05 and 0.95 quantiles.

5. Results

5.1. Time-to-Arrival Maps

In this section, we present results of autoregressive predictions from the DNN models via time-to-arrival (TTA) maps. These maps demonstrate how the models have learned to realistically represent the complex physical interaction between the fire front and aspects of the environment such as fuel type, terrain-slope and wind. Fire sequences were chosen that had roughly the average amount of error for each model, so they are good visualizations of how well predictions are doing on average across the entire dataset.

5.1.1. DCIGN Model Figure 5 shows TTA maps for a typical fire sequence in the *single fuel* dataset generated by the DCIGN model. The DCIGN model immediately made errors in predicting the direction the wildfire propagates. After just 120 min (eight autoregressive predictions), the DCIGN model started making unrealistic predictions of spurious fire spotting in regions far away from the actual fire front. The contour lines in Figure 5a show the small distance the fire travelled in the FARSITE simulation, whereas the contour lines in Figure 5b highlight the errors the DCIGN model made during that same time frame.

Qualitatively, results were similar on the *multiple fuel* and *California* datasets and while our work reveals some shortcomings of the DCIGN model in predicting short-time fire dynamics, it is important to note that it was successfully employed for predictions at larger time scales (6 or 24 h) [20, 21].

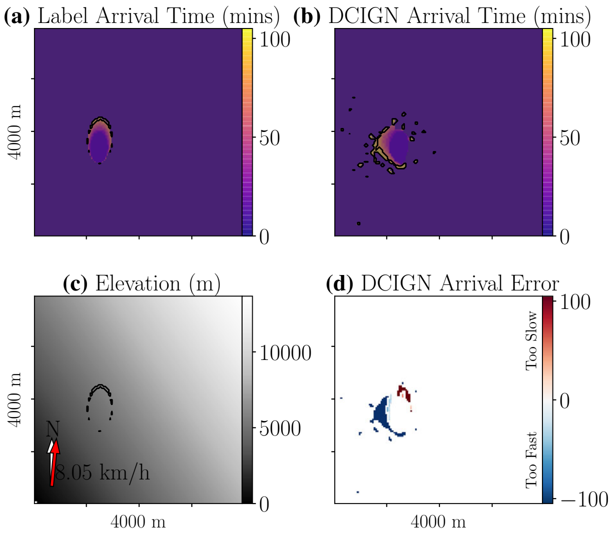


Figure 5. TTA maps for DCIGN model on a randomly selected fire sequence in the *single fuel* test dataset. a Arrival time in the ground truth generated by FARSITE; *background color* denotes that the homogeneous fuel type is GR1. b Arrival time predicted by the DCIGN model. c Elevation and wind with TTA contours. d Error in arrival time prediction. *Blue* and *red colors* denote overprediction and underprediction of fire spreading, respectively (Color figure online).

5.1.2. EPD and EPD-ConvLSTM Models Figure 6 provide results for the EPD and the EPD-ConvLSTM models on predicting the same *single fuel* fire sequence seen in Figure 5, but after 1500 min instead of just 90 min. Unlike the DCIGN model, both the EPD and EPD-ConvLSTM models made realistic, accurate and stable predictions. Even though there are no physical constraints enforced in the training of these models, both EPD and EPD-ConvLSTM models were able to reproduce the elliptical fire scars that are expected in field fires with constant wind and planar sloped terrain. In this context, we note that the introduction of physical principles during the training is expected to further improve the model accuracy and reduces the amount of data needed for training [18].

Figure 7 provides results for the EPD and EPD-ConvLSTM models in predicting the dynamics of a fire sequence in the *multiple fuel* dataset. Because there are variations in the fuel types used across fire sequences in this dataset, training and prediction are more difficult than in the *single fuel* dataset. In Figure 7, both the EPD and EPD-ConvLSTM models make some errors in their predictions. The EPD model struggles to maintain the proper elliptical shape of the fire front, and dramatically overpredicts the fire-spread rate. Conversely, predictions from the EPD-ConvLSTM model achieves a more elliptical shape and more accurately estimates the fire-spread rate. As expected, errors are more significant than in the *single fuel* dataset.

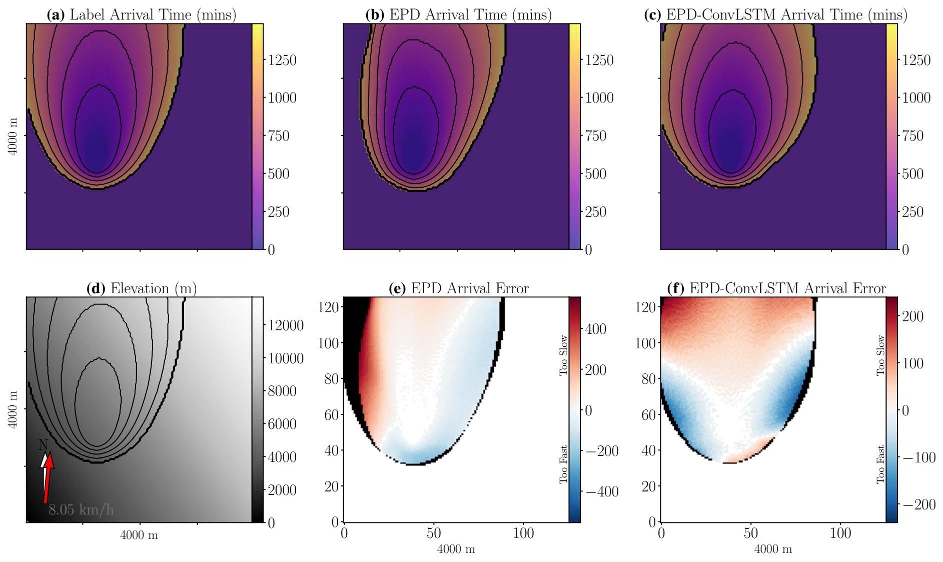


Figure 6. **TTA maps for EPD and EPD-ConvLSTM models on the same *single fuel* fire sequence seen in Figure 5 after 1500 min. a TTA map for the ground truth generated by FARSITE; the *background color* denotes that the fuel type, in this case the homogeneous fuel type is GR1. TTA maps predicted by b EPD model and c EPD-ConvLSTM model. d Elevation of the field and direction of the wind with the TTA contours from (a). e, f Errors in the predictions made in (b, c). *Blue* (or *red*) regions indicate predictions that were too fast (or too slow). The *colorbar* is symmetric around zero for readability, but underlying errors are not necessarily symmetric. *Black regions* indicate locations the fire was never supposed to reach (false positives) or regions that the fire never did reach (false negatives). The EPD's fire scar JSC was 0.86 (dataset average: 0.82). The EPD-ConvLSTM's fire scar JSC was 0.95 (dataset average: 0.94) (Color figure online).**

Figure 8 shows a comparison between the EPD and EPD-ConvLSTM models' predictions on a fire sequence randomly selected from the *California* dataset. The majority of the field is covered by timber litter fuel (green), with many small patches of slash burn fuel (light green). Thus, the general shape of the fire scar is roughly elliptical. The wind pushes the fire in the northeasterly direction but there are patches of grass shrub (light red) in the southern and northern regions where the fire accelerates more quickly than on the timber litter fuel. The southern patch of grass shrub is located on terrain that slopes upward, further accelerating the fire in the opposite direction of the wind. This acceleration can clearly be seen in the TTA contour lines near the bottom of the field.

Both the EPD and EPD-ConvLSTM models performed well and correctly considered the complex interactions between the various fuel types, the changes in elevation and the wind. The EPD model did not make any egregious errors, but did

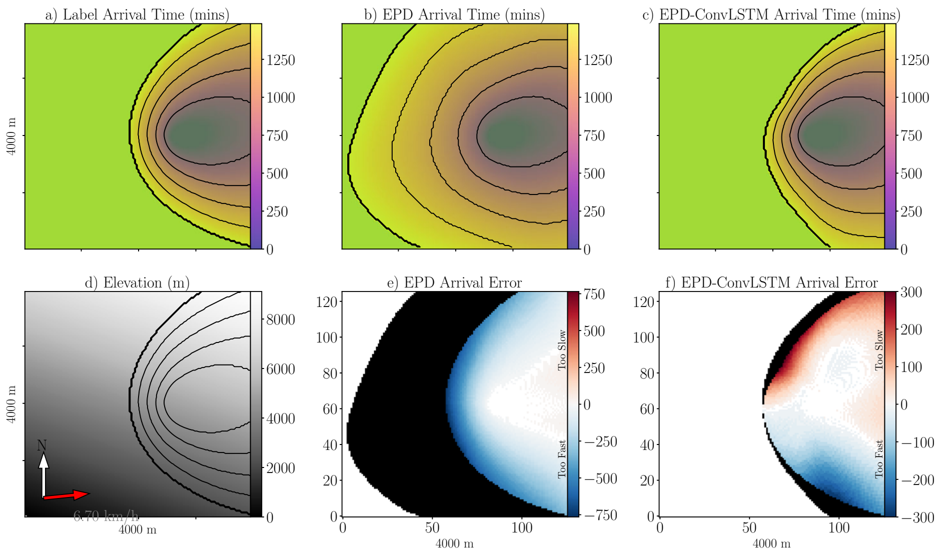


Figure 7. TTA maps for EPD and EPD-ConvLSTM models on a fire sequence in the *multiple fuel* dataset. *Panels* follow the same layout as in Figure 6. EPD's fire scar JSC was 0.47 (dramatically worse than dataset average of 0.74). EPD-ConvLSTM's fire scar JSC was 0.87 (dataset average: 0.89).

not estimate the speed of the fire propagation as accurately as the EPD-ConvLSTM model. The dark blue boundary in Figure 8e) denotes cells the EPD model predicted the fire would reach after 1500 min, but never actually did in the ground truth (false positives). Conversely, in Figure 8f, there are far fewer dark blue cells, but there are also some dark red cells, indicating locations the fire was supposed to reach, but were never predicted to do so (false negatives).

Figure 9 provides an example of a fire burning in the *California-WN* dataset. The EPD model's final Jaccard score for the scar was 0.86 and the EPD-ConvLSTM's final Jaccard score was 0.89, so both models did fairly well, although the EPD-ConvLSTM model did better. The fire can be seen adhering to changes in vegetation (note the lack of spread in the notch on the western front) as well as the prevailing wind. The models were trained only on the prevailing wind direction (the red arrow), but the FARSITE simulation used the more complex wind patterns determined by Wind Ninja. Both models accelerated the fire a little too quickly on the eastern front, but the EPD model also slowed the fire spread down too much on the south-western front.

5.2. Full Evaluation

This section provides the full evaluation made across all four datasets on the three metrics defined in Sect. 4.3: (1) The Jaccard score (higher is better), which measures the success of a binary classification of the cells in the prediction. 0.0 means

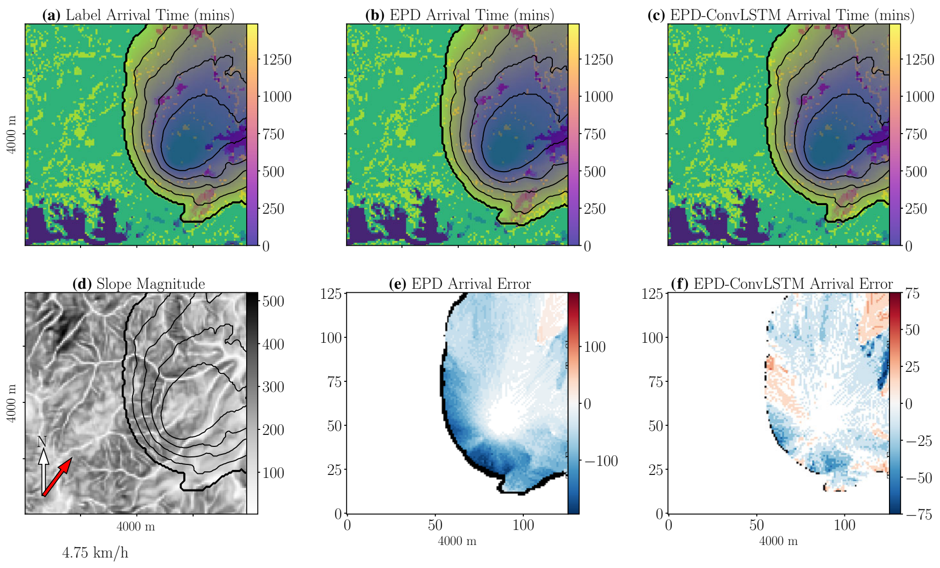

(a) Label Arrival Time (mins)
(b) EPD Arrival Time (mins)
(c) EPD-ConvLSTM Arrival Time (mins)
(d) Slope Magnitude
(e) EPD Arrival Error
(f) EPD-ConvLSTM Arrival Error
4000 m
N
4.75 km/h

Figure 8. TTA maps for the EPD and EPD-ConvLSTM model on a fire sequence in the *California* dataset. *Panels* follow the same layout as in Figure 6, except d provides the magnitude of the slope instead of the elevation. EPD's fire scar JSC was 0.94 (exactly the dataset average). EPD-ConvLSTM's fire scar JSC was 0.99 (better than the dataset average of 0.94) (Color figure online).

no cells in the label overlapped with the cells in the prediction. 1.0 indicates a perfect overlap. (2) The RMSE score (lower is better), which provides a general sense for how much regressive error exists on average across all cells predicted. (3) The STE scores (lower is better), which measures how much error exists in the total size of the predicted fire front and fire scar.

Figures 10, 11, 12 and 13 provides results for each of the four datasets. Solid lines correspond to results computed from one of the test datasets, whereas the dark-shaded and light-shaded regions correspond to 50% and 90% confidence intervals (respectively) estimated from the 20 bootstrapping datasets described in Sect. 4.4. Each column of graphs provides the results across all datasets for a single metric. The top row of graphs in each section provides results for the fire *front* predictions, and the bottom row of graphs provides results for the fire *scar* predictions. The same graph in each of the three sections uses the same scale for the vertical axis, allowing the same metric to easily be compared across the four datasets. Table 2 provides the numeric values for each metric on the 1st, 25th, 50th, 75th and 100th autoregressive predictions.

The fire front JSC and fire scar JSC values (first column) are the classification-based metrics that we used to evaluate the models. The results across all four datasets demonstrated that the EPD-ConvLSTM model consistently outperformed the EPD model. Across all six graphs, the EPD-ConvLSTM's JSC was statisti-

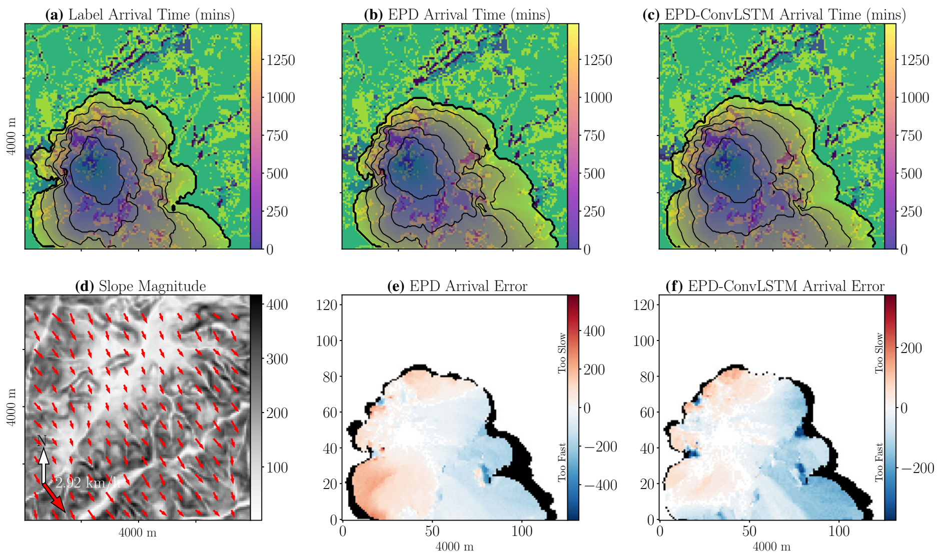

(a) Label Arrival Time (mins)
(b) EPD Arrival Time (mins)
(c) EPD-ConvLSTM Arrival Time (mins)
(d) Slope Magnitude
(e) EPD Arrival Error
(f) EPD-ConvLSTM Arrival Error
4000 m
N
2.92 km/h
Too Slow
Too Fast

Figure 9. TTA maps for the EPD and EPD-ConvLSTM model on the *California-WN* dataset. *Panels* follow the same layout as in Figure 8, except the *countour lines* in d are replaced with the dynamics generated by Wind Ninja. EPD's fire scar JSC was 0.86 (much better than dataset average of 0.38). EPD-ConvLSTM's fire scar JSC was 0.89 (slightly worse than dataset average of 0.93) (Color figure online).

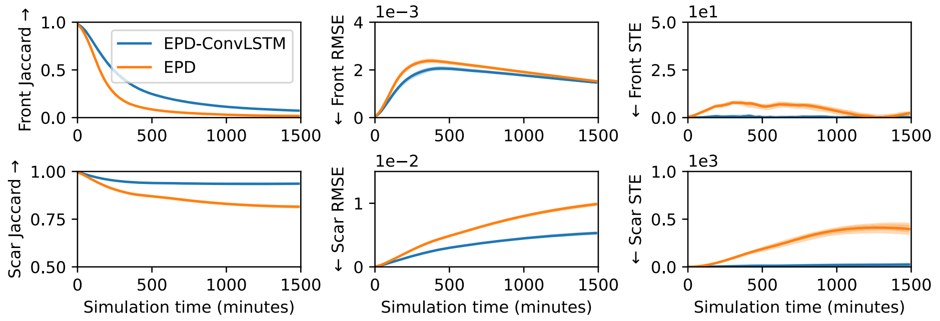


Figure 10. Full evaluation results for the *single fuel* dataset. *Axis arrows* show direction of better score.

cally significantly higher than the corresponding EPD model's score, with the exception of the 75th and 100th JSC on the fire front. This indicates that the EPD-ConvLSTM model did a better job selecting the specific cells that the fire-front spread transitioned to during each of the autoregressive predictions.

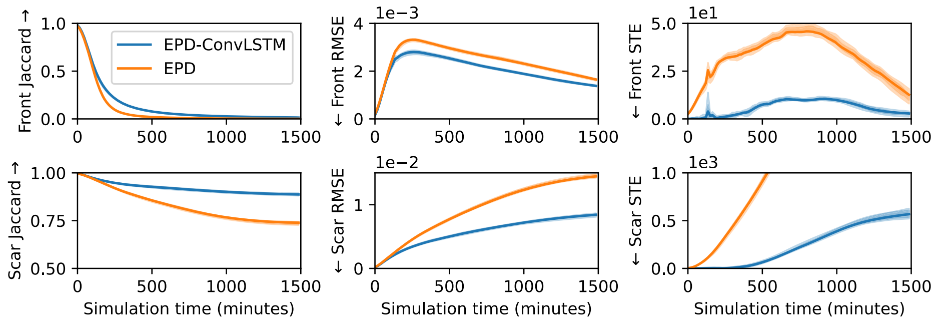


Figure 11. Full evaluation results for the *multiple fuel* dataset. *Axis arrows* show direction of better score.

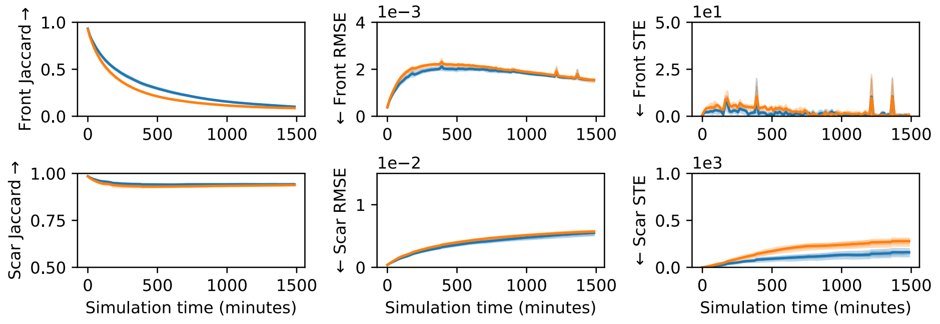


Figure 12. Full evaluation results for the *California* dataset. *Axis arrows* show direction of better score.

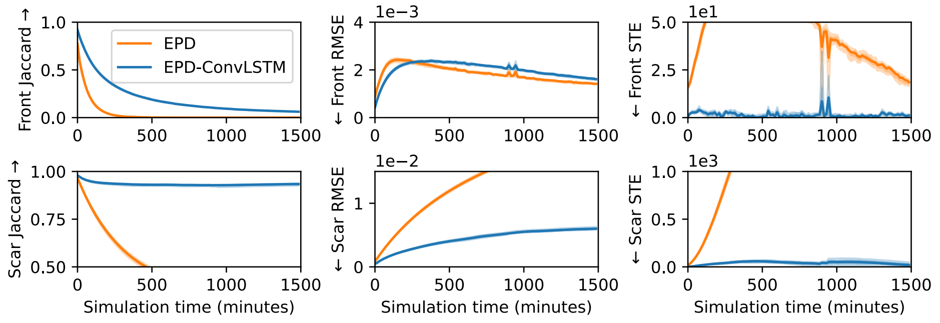


Figure 13. Full evaluation results for the *California-WN* dataset. *Axis arrows* show direction of better score.

Also note the stark differences between the JSCs on the *Front* metrics versus the *Scar* metrics. The front metrics drop much more quickly than the scar metrics do, which is expected given the significantly increased difficulty of predicting the exact time-resolved location of the small fire-front. Consider, for example, that the fire

Table 2
Full Evaluation of the EPD-ConvLSTM and EPD Models on All Four Datasets

	EPD-ConvLSTM						EPD					
Step	Front Jcd	Scar Jcd	Front STE	Scar STE	Front RMSE	Scar RMSE	Front Jcd	Scar Jcd	Front STE	Scar STE	Front RMSE	Scar RMSE
Single fuel												
1	0.98	1.00	1.9E−2	1.9E−2	4.7E−5	4.7E−5	0.98	1.00	3.2E−1	3.2E−1	5.7E−5	5.7E−5
25	0.33	0.94	6.4E−1	8.9e+0	2.0E−3	2.4E−3	0.13	0.88	7.6E+0	1.3E+2	2.4E−3	3.9E−3
50	0.16	0.94	2.6E−1	1.3E+1	1.9E−3	3.8E−3	0.05	0.85	6.4E+0	2.9E+2	2.1E−3	6.6E−3
75	0.10	0.93	9.0E−2	2.0E+1	1.7E−3	4.8E−3	0.02	0.82	1.4E+0	4.0E+2	1.8E−3	8.6E−3
100	0.07	0.94	2.5E−1	2.4E+1	1.5E−3	5.3E−3	0.02	0.82	2.2E+0	3.9E+2	1.5E−3	9.9E−3
Multiple fuel												
1	0.98	1.00	4.6E−2	4.6E−2	1.5E−4	1.5E−4	0.97	1.00	2.5E+0	2.5E+0	2.0E−4	2.0E−4
25	0.12	0.94	3.2E+0	1.9E+1	2.7E−3	4.3E−3	0.04	0.89	3.4E+1	6.5E+2	3.2E−3	6.2E−3
50	0.04	0.91	9.9E+0	2.2E+2	2.2E−3	6.2E−3	0.01	0.81	4.5E+1	1.6E+3	2.6E−3	1.0E−2
75	0.02	0.90	7.6E+0	4.6E+2	1.7E−3	7.6E−3	0.00	0.75	3.3E+1	2.6E+3	2.1E−3	2173–2
100	0.01	0.89	2.8E+0	5.6E+2	1.4E−3	8.4E−3	0.00	0.74	1.2E+1	3.1E+3	1.7E−3	1.4E−2
California												
1	0.93	0.99	4.3E−1	4.3E−1	**3.8E−4**	3.8E−4	0.93	0.98	8.2E−1	8.2E−1	**3.9E−4**	3.9E−4
25	0.36	0.94	2.0E+0	7.1E+1	**2.0E−3**	3.1E−3	0.27	0.93	4.3E+0	1.3E+2	**2.2E−3**	3.5E−3
50	0.21	0.94	1.5E+0	1.1E+2	**1.9E−3**	4.2E−3	0.15	0.93	2.2E+0	2.2E+2	**2.1E−3**	4.7E−3
75	0.14	**0.94**	3.7E−1	1.3E+2	**1.7E−3**	5.0E−3	0.10	**0.94**	1.4E+0	2.4E+2	**1.8E−3**	5.3E−3
100	0.10	**0.94**	4.1E−2	1.5E+2	**1.5E−3**	5.5E−3	0.09	**0.94**	6.2E−1	2.7E+2	**1.5E−3**	5.7E−3
California-WN												
1	0.92	0.98	6.4E−01	6.4E−01	4.3E−04	4.3E−04	0.77	0.97	1.6E+01	1.6E+01	8.7E−04	8.7E−04
25	0.25	0.93	2.0E+00	5.5E+01	2.4E−03	3.4E−03	0.00	0.55	7.0E+01	1.4E+03	2.2E−03	9.7E−03
50	0.13	0.93	7.5E−01	4.2E+01	2.2E−03	4.9E−03	0.00	0.42	5.6E+01	3.2E+03	1.9E−03	1.5E−02
75	0.08	0.93	6.0E−01	5.1E+01	1.9E−03	5.6E−03	0.00	0.38	3.4E+01	4.1E+03	1.6E−03	1.9E−02
100	0.06	0.93	1.1E+00	2.0E+01	1.6E−03	6.0E−03	0.00	0.38	1.9E+01	4.8E+03	1.4E−03	2.1E−02

Bold fonts indicates values with overlapping confidence intervals. With the sole exceptions of the *Front RMSE*, and the 75th and 100th steps of the Scar Jcd, the EPD-ConvLSTM statistically significantly outperformed the EPD model front may only be one cell wide, and thus, even when the prediction is off by just a single cell, the front JSC drops to 0.0. A low front JSC is not an indicator that the front is misplaced by much, and indeed, a high scar JSC score can only exist if the front's misplacement is minor.

Figure 9f provides a TTA map for a prediction made by the EPD-ConvLSTM model in this dataset. The black areas visualize the error in the final scar, but there are many more non-black cells than black cells, which explains the high scar JSC. However, the blue and pink cells correspond to regions in which the fire front was slightly misplaced and the front JSC will be low for all the pink/blue cells in the TTA map, and since at most time steps, there is some error in the exact placement of the front, the front JSC remains low even though throughout the predicted simulation, the fire scar remains highly accurate.

The second column of graphs provides the RMSE metric. As expected, the RMSE results are qualitatively similar to the Jaccard results. The pattern of increased performance on this metric roughly mirrors the same increased performance seen in the JSC results.

The third column of graphs provides the results for the STE metric. This metric measures the overall ability for the DNN models to predict the correct size of the fire front and fire scar, regardless of whether the location was correct. On predicting the overall size of the fire scar, the EPD-ConvLSTM model clearly outperformed the EPD model on all four datasets. On predicting the overall size of the fire front, the EPD-ConvLSTM model clearly outperformed the EPD model on the *single fuel*, *multiple fuel* and *California-WN* datasets, but only outperformed the EPD model in the first 50 autoregressive predictions on the *California* dataset. The strong performance of the EPD-ConvLSTM model on this metric indicates that adding temporal layers to the EPD model results in the model more successfully estimating the speed at which the fire propagates.

The results on the *California-WN* dataset demonstrate that the EPD-ConvLSTM model was capable of effectively coping with the hidden spatially nonuniform wind dynamics. Compared to the *California* dataset, the JSC for the fire scar only decreased from 0.94 to 0.93 and decreased from 0.10 to 0.06 for the fire front. The EPD model's behavior was more nuanced. It outperformed the EPD-ConvLSTM model on the fire front prediction as measured by RMSE (and recall, this metric is basically the same used for the loss function during training), however, results on all other metrics worsened.

This was the only occurrence when the RMSE metric and the JSC metrics disagreed significantly. We suspect this disagreement is due to poor performance of the EPD model on slow-growth fires that impact only small fractions of the fire patch. The EPD model tends to overburn these fires, which only has a nominal impact on the RMSE (since the majority of the unburnt cells are correctly classified), but has a significant impact on JSC (since it takes relatively few errant cells to get a low score from a fire that also only affects a few cells).

We had anticipated that the two *California* datasets would be the most difficult to predict. They contain realistic and complex arrangements of vegetation, moisture content, elevation, wind, etc., whereas the other two datasets contain entirely homogeneous distributions of fuel on planar landscapes. However, the realistic landscapes result in patches of terrain that often significantly constrain the direction and speed that a fire can reasonably propagate. It was not hard for either model to learn, e.g., that fires propagate more slowly through some fuels than others, and patches with either slow-burning fuel or even nonburnable fuel are easier to predict than the freedom the fire has to move in the *single fuel* and especially *multiple fuel* fires.

Indeed, the most difficult dataset to classify was the *multiple fuel* dataset. Consider that if a particular cell is on fire in a *multiple fuel* fire, it will propagate to some degree into all eight neighboring cells. Our models are, intentionally, entirely empirical and make no assumptions about the shape or propagation dynamics that the fire will take. This allows the same method for training a model to be learned on any of the fire datasets.

While the accuracy of the EPD model is clearly improved with the addition of the temporal ConvLSTM layers, that improvement comes at the cost of increased inference time. The EPD model alone takes approximately 7ms to perform a single inference. The EPD-ConvLSTM model takes approximately 61ms to perform the same inference. FARSITE takes anywhere between 20 ms and 100 ms (depending on how much fire is in the patch).

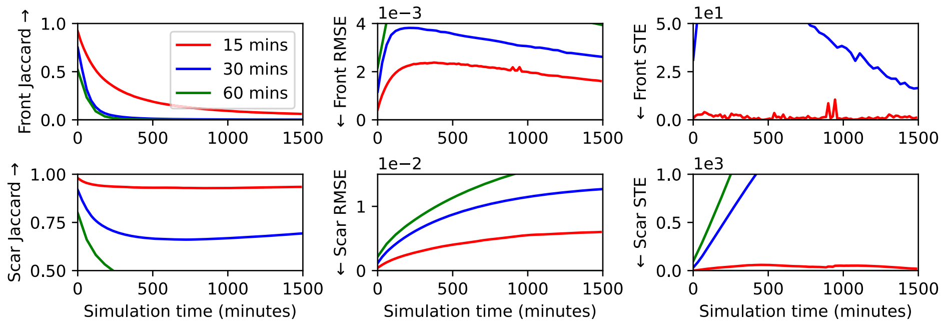
5.3. *Time Step Analysis*

This section provides an analysis on how the EPD and EPD-ConvLSTM models performed with increasing step sizes on the most realistic dataset, the *California-WN* dataset. The original datasets used 15 min time increments. By computing the union of the fire growth on sequential time steps, 30-min and 60-min versions of the original 10,000 fire sequences were generated. New models were trained following the same process detailed in Sect. 4.2.

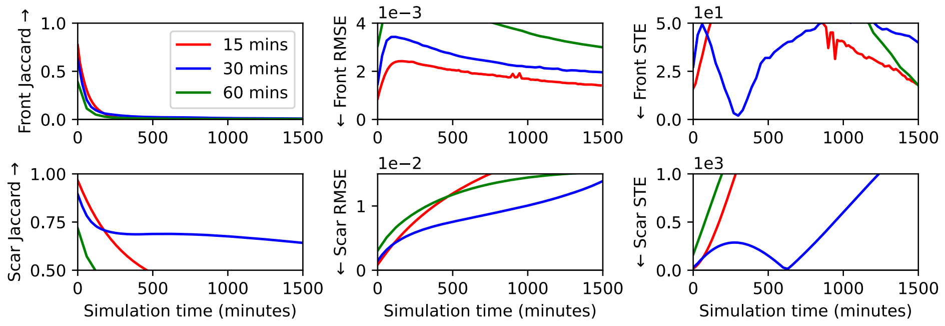
Figure 14a shows how the performance degrades over the increased time-step size. While the 30- and 60-min COLOUR models need fewer time steps to cover the same duration, and thus will suffer less from compounding errors in the autoregressive process, a clear preference for smaller time steps is observable in the results. Performance across all metrics degrades as the time step increases. Figure 14b shows the performance on the EPD model. Note that the fire front RMSE score is basically the same as the score used in the training loss function and thus the decayed performance as the time step increases is clearly present here. However, the 15-min version of the model actually performed more poorly than the 30-min version on all other metrics. This appeared, at least in part, due to the EPD model performing more poorly on slow-burning fires (making errors on slow-burning fires results in only small penalties to the RMSE metric, but much more significant penalties to the Jaccard metric).

6. Conclusions

Our primary contribution is evaluating the efficacy of recurrent convolutional neural networks to predicting the time-dependent behavior of wildfires. Previous studies demonstrated that convolutional neural networks can make effective



(a) EPD-ConvLSTM.



(b) EPD.

Figure 14. Full evaluation of the EPD-ConvLSTM and EPD models on different time steps.

predictions of the fire scar in one single large time step [20, 21]. We instead focus on an autoregressive process in which many small predictions are sequentially made to predict the fire scar after a large amount of time. This method allows for the model to continuously adapt to changing dynamic environment conditions and more precisely models the evolution of the fire front by not only modeling when the fire reached a particular location, but also by explicitly predicting how much fuel burns at each location at each time step.

We demonstrate that a popular previously employed model for similar predictions, the Deep Convolutional Inverse Graphics Network (DCIGN) model, was not well-suited for making the fine-grained temporal autoregressive predictions, as the transformations in that model facilitate large-scale changes to the input state, resulting in the model generalizing poorly in this context. We replace the transformations in the DCIGN model with more suitable transformations, leading basically to the same structure seen in the Encoder-Processor-Decoder (EPD) model. We demonstrate that the EPD model is stable and can realistically propagate a fire front forward in time for upwards of 100 autoregressive predictions. We added recurrent layers to the EPD model, resulting in the EPD-ConvLSTM model, which significantly improves the accuracy of the predictions, achieving

JSCs on the 100th autoregressive prediction of the fire scar of: 0.94 (*single fuel*), 0.89 (*multiple fuel*), 0.94 (*California*) and 0.94 (*California-WN*).

The poor performance of the EPD model on the *California-WN* dataset also suggests that using MSE as the loss function may be introducing a bias against modeling the dynamics of small fires, since errors on small fires do not negatively impact the MSE score nearly as much as they would a metric like the JSC. Updating the loss function to contain a JSC-like term may help.

We believe a primary direction for future work is to identify a DNN model that combines the strengths of our approach (fine-scale time-resolved predictions) with the strengths of prior work (single long-term predictions over large periods of time). Incorporating uncertainty estimation into such a model would allow it to determine the optimal time step to make predictions given the current state of the fire. Complex periods in the simulation could use fine-scaled time-resolved predictions to maintain accurate time-resolved predictive efficacy, whereas less challenging periods could use large time-step predictions to minimize computational overhead. Further, the uncertainty estimates themselves would be particularly valuable, as typical approaches to estimating uncertainty *in fire propagation simulations often* require running the underlying simulation multiple times.

Acknowledgements

We thank the reviewers for their constructive feedback. MB acknowledges financial support through the NASA Fellowship Activity, #80NSSC21K2054, and the Stanford University School of Engineering Departmental Fellowship. We thank the members of Google Research's AI For Weather team: Jason Hickey, Cenk Gazen and Shreya Agrawal, for their support in this project. All of the training, validation and test data, and each of the four EPD and EPD-ConvLSTM models, as well code that will facilitate performing inferences, is available through Google's research GitHub repository at https://github.com/google-research/google-research/tree/master/wildfire_conv_lstm.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix 1: Deep Neural Network Transformations

While a full summary of deep neural networks is outside the scope of this paper, we provide a brief description of the transformations used in the DNNs described in this work. For a deeper background on DNNs, we refer the interested reader to [30].

The fundamental unit processed by a DNN is a tensor. A tensor is an n dimensional collection of scalar values, $n \geq 0$. For example, a single field specifying how much fuel exists in a 2D patch of ground could be stored in a 2D tensor with shape (H, W) where H is the height of the field and W is the width. That field represents a *channel* of information and if there are additional channels, they can be stored in a tensor of shape (H, W, C) where C is the number of channels. A time series of fields can be stored in a tensor of shape (T, H, W, C) where T is the number time points in the series. Training a DNN is usually done in batches, so a DNN that works on (T, H, W, C) data will actually take (B, T, H, W, C) data where B is the number of data points in a single batch.

DNNs are often said to *predict* some outcome based on a given data point though in reality, the output is merely the input after its been passed through a potentially large number of transformations. We sometimes refer to sets of transformations as a *layer*. Some layers contain parameters such that the process of training a DNN attempts to find the optimal set of values that result in transforming as much of the training input into as correct a set of output as possible. These are the primary transformations used in this work:

`fully connected`: Every value in the output is a parameterized combination of every value in the input. Given an input shape (B, M) and output shape (B, N), there are $M \times N$ total parameters. This layer can be useful when many output values need to consider many input values, but can be particularly prone to overfitting and parameter bloat.

`convolutional (2D)`: This layer is appropriate for inputs that contain 2D fields. A 2D convolutional kernel is swept across both dimensions of the input field. At each location, the dot product between the kernel and the 2D field is computed, which generates a new field that is the convolution of the input field with the kernel. There can be multiple independent kernels, resulting in multiple field outputs. Given an input shape of (B, H, W, C), the output shape is (B, H, W, K) where K is the number of kernels.

`recurrent`: A layer which explicitly considers temporal relationships in the input data when generating the output. Unlike spatial relationships, temporal relationships often cannot effectively be modeled by simply considering neighboring input. Temporal dynamics often require considering events that occurred at more distant times in the past. Recurrent layers build up a *memory* by iterating over individual time points one at a time. Cells in the output depend on the memory instead of actual time points in the past. If the input contains fields, the layers will also leverage convolutions such that an output cell can depend on the memory of itself, and the memory of neighboring cells. If the convolutions have K kernels and the input shape is (B, T, H, W, C), then the output shape will be (B, T, H, W, K). Given its wide-spread success at modeling image-to-image type tasks, we use the Convolutional Long-Short Term Memory layer (ConvLSTM) [37].

`activation`: Passes each value in the input through an often non-linear and potentially parameterized function. The output shape is equal to the input shape.

`skip link`: Taking the input for some transformation and concatenating (or adding) it with the output of the transformation. This effectively gives a path for the gradients of the model's parameters to *skip* the transformation during training, *significantly* increasing the efficacy of training large networks.
`max pooling (2D)`: A field transformation that downsamples the spatial dimensions of the input and while typically doubling the number of channels. If the input has shape (*B*, *H*, *W*, *C*), then the output will have shape $(B, H/2, W/2, C \times 2)$.
`zero padding`: This layer adds padding around the field. If the input shape is (*B*, *H*, *W*, *C*) and the padding size is *P* then the output shape will be $(B, H + 2P, W + 2P, C)$.
`one hot encoding`: This layer transforms integer values in the input into a list of all zeros with the exception of a single element in the list corresponding to the integer value being given a value of one. If the input shape is (*B*, *H*, *W*, *C*), the output will be (*B*, *H*, *W*, *C*, *E*) where *E* is the maximum value the input can take.
`embedding`: This layer converts a field that contains discrete valued cells into an field where each cell is replaced by a set of floating-point values. Values in the input that result in similar predictions are placed close to each other in the embedding space, which subsequent layers in the model can leverage more effectively than the discrete values alone. If the input has shape (*B*, *H*, *W*, 1) and the embedding space has rank *E*, then the output shape will be (*B*, *H*, *W*, *E*).

References

1. Covington WW, Moore MM (1994) Southwestern ponderosa forest structure: changes since Euro-American settlement. J Forest 92:39–47
2. Westerling AL, Hidalgo HG, Cayan DR, Swetnam TW (2006) Warming and earlier spring increase Western U.S. forest wildfire activity. Science 313:940–943
3. Mueller SE, Thode AE, Margolis EQ, Yocom LL, Young JD, Iniguez JM (2020) Climate relationships with increasing wildfire in the southwestern US from 1984 to 2015. For Ecol Manage 460:117861
4. Thomas D, Butry D, Gilbert S, Webb D, Fung J (2017) The costs and losses of wildfires: a literature survey. NIST Special Publication 1215. National Institute of Standards and Technology, Gaithersburg
5. Kollanus V, Prank M, Gens A, Soares J, Vira J, Kukkonen J, Sofiev M, Salonen RO, Lanki T (2017) Mortality due to vegetation fire-originated PM2.5 exposure in Europe-Assessment for the years 2005 and 2008. Environ Health Perspect 125:30–37
6. van der Werf GR, Randerson JT, Giglio L, van Leeuwen TT, Chen Y, Rogers BM, Mu M, van Marle MJE, Morton DC, Collatz GJ, Yokelson RJ, Kasibhatla PS (2017) Global fire emissions estimates during 1997–2016. Earth Syst Sci Data 9:697–720
7. Bakhshaii A, Johnson EA (2019) A review of a new generation of wildfire-atmosphere modeling. Can J For Res 49:565–574
8. Sullivan AL (2009) Wildland surface fire spread modelling, 1990–2007. 1. Physical and quasi-physical models. Int J Wildl Fire 18:349–368

9. Sullivan AL (2009) Wildland surface fire spread modelling, 1990–2007. 2. Empirical and quasi-empirical models. Int J Wildl Fire 18:369–386
10. Sullivan AL (2009) Wildland surface fire spread modelling, 199–2007. 3. Simulation and mathematical analogue models. Int J Wildl Fire 18:387–403
11. Linn RR, Harlow FH (1997) FIRETEC: a transport description of wildfire behavior, LA-UR-97-3920. Los Alamos National Laboratory, Los Alamos
12. Larini M, Giroud F, Porterie B, Loraud JC (1998) A multiphase formulation for fire propagation in heterogeneous combustible media. Int J Heat Mass Transf 41:881–897
13. Mell W, Jenkins MA, Gould J, Cheney P (2007) A physics-based approach to modelling grassland fires. Int J Wildl Fire 16:1–22
14. Rothermel RC (1972) A mathematical model for predicting fire spread in wildland fuels. Research Paper INT-115. USDA Forest Service, Intermountain Forest and Range Experiment Station, Ogden
15. P. L. Andrews PL (1986) BEHAVE: fire behavior prediction and fuel modeling system–BURN Subsystem, part 1, General Technical Report INT-194. USDA Forest Service, Intermountain Research Station, Ogden
16. Finney MA (1998) FARSITE: fire area simulator–model development and evaluation. Research Paper RMRS-RP-4. USDA Forest Service, Rocky Mountain Research Station. Revised 2004
17. Jain P, Coogan SCP, Subramanian SG, Crowley M, Taylor S, Flannigan MD (2020) A review of machine learning applications in wildfire science and management. Environ Rev 28:478–505
18. Ihme M, Chung WT, Mishra AA (2022) Combustion machine learning: principles, progress and prospects. Prog Energy Combust Sci 91:101010
19. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521:436–444
20. Hodges JL, Lattimer BY (2019) Wildland fire spread modeling using convolutional neural networks. Fire Technol 55:2115–2142
21. Radke D, Hessler A, Ellsworth D (2019) Proceedings of the twenty-eighth international joint conference on artificial intelligence (IJCAI-19). International Joint Conferences on Artificial Intelligence Organization, pp 4575–4581
22. Burge J, Bonanni M, Ihme M, Hu L (2020) Convolutional LSTM neural networks for modeling wildland fire dynamics. arXiv Preprint. arXiv:2012.06679
23. Bolt A, Huston C, Kuhnert P, Dabrowski JJ, Hilton J, Sanderson C (2022) A spatio-temporal neural network forecasting approach for emulation of firefront models. arXiv Preprint. arXiv:2206.08523
24. Kulkarni TD, Whitney WF, Kohli P, Tenenbaum J (2015) In: Cortes C, Lawrence N, Lee D, Sugiyama M, Garnett R (eds) Advances in neural information processing systems, vol 28. Curran Associates, Montreal
25. Sanchez-Gonzalez A, Godwin J, Pfaff T, Ying R, Leskovec, Battaglia P (2020) In: Daumé III H, Singh A (eds) Proceedings of the 37th international conference on machine learning, vol 119. Proceedings of Machine Learning Research, pp 8459–8468
26. Kochkov D, Smith JA, Alieva A, Wang Q, Brenner MP, Hoyer S (2021) Machine learning accelerated computational fluid dynamics. Proc Natl Acad Sci USA 118:e2101784118
27. Rollins MG, Rollins MG (2009) LANDFIRE: A nationally consistent vegetation, wildland fire, and fuel assessment. Int J Wildl Fire 18:235–249
28. Scott JH, Burgan RE (2005) Standard fire behavior fuel models: a comprehensive set for use with Rothermel's surface fire spread model. General Technical Report RMRS-GTR-153. USDA Forest Service Rocky Mountain Research Station, Fort Collins

29. Forthofer JM, Butler BW, Wagenbrenner NS, Forthofer JM, Butler BW, Wagenbrenner NS (2014) A comparison of three approaches for simulating fine-scale surface winds in support of wildland fire management. Part I. Model formulation and comparison against measurements. Int J Wildl Fire 23:969–981
30. Foster D (2019) Generative deep learning. O'Reilly, Sebastopol
31. He K, Zhang X, Ren S, Sun J (2016) IEEE conference on computer vision and pattern recognition (CVPR), Las Vegas, pp 770–778. https://doi.org/10.1109/CVPR.2016.90
32. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jozefowicz YR, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X (2015) TensorFlow: large-scale machine learning on heterogeneous systems. TensorFlow: large-scale machine learning on heterogeneous systems 2015. https://iwww.tensorflow.org. Accessed Aug 4, 2023
33. Chollet F et al (2015) Keras. https://keras.io. Accessed Aug 4, 2023
34. Kingma DP, Ba J (2014) Proceedings of the international conference for learning representations. arXiv Preprint. arXiv:1412.6980
35. Filippi J-B, Mallet V, Nader B (2014) Representation and evaluation of wildfire propagation simulations. Int J Wildl Fire 23:46
36. Tibshirani RJ, Efron B (1993) An introduction to the bootstrap. Monographs on statistics and applied probability, vol 57. Chapman and Hall, New York
37. Shi X, Chen Z, Wang H, Yeung D-Y, Wong W, Woo W (2015) In: Cortes C, Lawrence N, Lee D, Sugiyama M, Garnett R (eds) Advances in neural information processing systems, vol 28. Curran Associates, Montreal