Accelerating Wildfire Spread Prediction for Guidance: A ConvLSTM Deep Learning Approach

Hamza Chakraa, Murat Bronz

Accelerating Wildfire Spread Prediction for Guidance: A ConvLSTM Deep Learning Approach

Hamza Chakraa * and Murat Bronz †
Fédération ENAC ISAE-SUPAERO ONERA, Université de Toulouse, F-31400 Toulouse, France

Wildfire simulations play a critical role in emergency response planning, resource allocation, and the deployment of autonomous systems such as Unmanned Aerial Systems (UAS) for monitoring and suppression. However, physically accurate models for wildfire propagation are often too computationally expensive to be deployed in real-time applications, particularly those involving iterative decision-making like Reinforcement Learning (RL)-based guidance systems. In this work, we propose a fast deep learning-based model that approximates the dynamics of a wildfire propagation simulator. We use a Recurrent Neural Network (RNN) architecture based on Convolutional Long Short-Term Memory (ConvLSTM) units, trained on data generated from a deterministic Cellular Automata (CA) fire model. Our model learns to predict the fire state at the next time step, conditioned on the current state to simulate multiple time steps forward, producing the fire spread patterns with significantly reduced computational cost. The network is designed to capture long-range spatial–temporal dependencies that commonly drive nonlinear wildfire behavior. This enables the model to generalize across diverse ignition patterns and environmental conditions present in the training set. Results indicate good accuracy compared to the ground truth and up to two orders of magnitude faster propagation times, showing promise for high-frequency deployment in guidance loops.

I. Nomenclature

Acronyms:

CA	Cellular Automata
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
ConvLSTM	Convolutional LSTM
BCE	Binary Cross Entropy
MSE	Mean Squared Error
LR	Learning Rate
PA	Pixel Accuracy
JSC	Jaccard Similarity Coefficient
HD	Hausdorff Distance
IoU	Intersection over Union
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
RL	Reinforcement Learning

*Post-Doc Researcher, ENAC F-31055 Toulouse, France, AIAA Member, e-mail: hamza.chakraa@enac.fr
†Assistant Professor, ENAC F-31055 Toulouse, France, AIAA Member, e-mail: murat.bronz@enac.fr

Symbols:

x_t	Input vector to LSTM at time t
h_t	LSTM hidden state at time t
c_t	LSTM cell state at time t
$\tilde{c}_t$	Candidate LSTM cell state
i_t	LSTM input gate
f_t	LSTM forget gate
o_t	LSTM output gate
W_i, W_f, W_o, W_c	Input weight matrices
U_i, U_f, U_o, U_c	Recurrent weight matrices
b_i, b_f, b_o, b_c	Bias vectors
X_t	ConvLSTM input tensor
H_t	ConvLSTM hidden state tensor
C_t	ConvLSTM cell state tensor
$\tilde{C}_t$	ConvLSTM candidate cell state
l	Input history length
Y_t	Ground-truth map at time t
$\hat{Y}_t$	Predicted map at time t
$\hat{\mathcal{Y}}$	Predicted frame sequence
$\mathcal{H}$	Sliding input sequence
$\mathcal{H}^{(0)}$	Initial input sequence
$\mathcal{M}$	Trained model
y_i	Ground-truth pixel label
$\hat{y}_i$	Predicted fire probability
$\mathcal{L}$	Loss function
P	Predicted fire pixels
G	True fire pixels
η_t	Learning rate
γ	LR reduction factor
v_t	Validation loss
p_a	Scheduler patience
N_H	Grid height
N_W	Grid width
T	Simulation time steps
$\mathcal{G}$	Grid domain
$\mathcal{S}_t$	Fire state
$\mathcal{F}_t$	Fuel state
ρ_{veg}	Vegetation density
θ_w	Wind direction
ω	Wind strength
$\mathcal{E}$	Elevation
$\mathcal{M}$	Moisture
$\alpha_M, \beta_M, \gamma_M$	Moisture parameters
$\mathcal{K}$	Slope factor
$\mathcal{K}_{\min}, \mathcal{K}_{\max}$	Slope bounds
$\mathcal{R}$	Burn rate
$\mathcal{R}_{\min}, \mathcal{R}_{\max}$	Burn rate range
α_B	Burn rate parameter
k	Neighbor index
θ_k	Neighbor angle
$\delta_i^{(k)}$	Row offset to neighbor k
$\delta_j^{(k)}$	Column offset to neighbor k
$\mathcal{N}(i, j)$	Moore neighborhood
$\mathcal{I}_t$	Ignition rule

II. Introduction

Wildfires are uncontrolled fires that ignite in vegetated, non-urban areas and can rapidly spread, threatening nearby communities. Once initiated, the fire spreads as thermal energy heats surrounding vegetation to its ignition point. The leading boundary of this advancing flame is known as the fire front, which continues to expand as long as combustible fuel remains available and suppression efforts are ineffective or absent. The dynamics of wildfire spread are influenced by a complex interplay of environmental factors, including terrain topology, wind, fuel type, and weather conditions, resulting in behaviors ranging from predictable elliptical fronts to highly irregular propagation (Figure 1 illustrates this with a satellite image of a wind-driven wildfire). Understanding and accurately predicting wildfire behavior is critical for the development of effective emergency response strategies and the optimal allocation of firefighting resources. Thus, considerable research efforts have been devoted to creating predictive models that simulate wildfire propagation, aiming to support early warning systems, optimize firefighting strategies, and minimize environmental and human impacts [1]. Efforts to model wildfire behavior began in the 1920s, aiming to establish systematic relationships between wildfire spread and environmental factors [2, 3]. Wildfire behavior is shaped by a complex mix of physical and chemical interactions, making it extremely difficult to create fully accurate models. As a result, researchers adopt various modeling strategies depending on the fire type and the particular behavior they aim to understand in more detail. Current models generally fall into two main categories: physical and empirical models [4, 5].

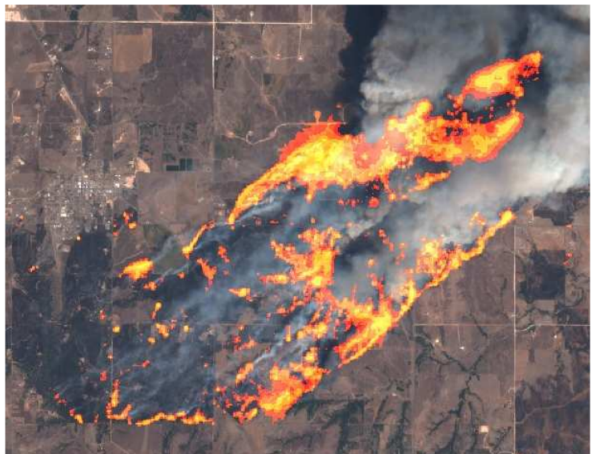


Fig. 1 A satellite view of a wildfire driven by strong winds

Physical models simulate wildfire behavior using the fundamental principles of physics and chemistry. They incorporate detailed representations of heat transfer, fluid dynamics, combustion, and interactions with environmental factors such as wind, topography, and fuel moisture. These models aim to replicate the actual mechanisms driving fire spread, allowing for accurate simulations that can adapt to a wide range of conditions. For example, FIRETEC [6] and WFDS (Wildland-Urban Interface Fire Dynamics Simulator) [7] are sophisticated simulators based on physical models that use computational fluid dynamics to analyze how fires evolve in complex environments. Although highly accurate in theory, physical models require significant computational resources and detailed input data, making them more suitable for research and planning rather than real-time management. The wildfire simulation model introduced in [8] is a comprehensive, multi-scale framework designed to realistically simulate wildfire behavior in 3D environments. This model captures the complexity of wildfires by integrating detailed representations of vegetation structure, varying fuel types, atmospheric dynamics, and soil conditions. A key innovation lies in its physically plausible modeling of fuel moisture and heat transfer, allowing for nuanced simulations of fire progression from ground to crown fires across different ecosystems. The result is a highly detailed, interactive simulation platform useful not only for visual effects and scientific analysis but also for firefighting training. On the other hand, empirical models rely heavily on observed data and statistical relationships derived from historical wildfire events. Unlike physics-based models, empirical models focus on identifying patterns and correlations in data to forecast key fire characteristics, such as rate of spread, flame length, fire line intensity, and fuel consumption. These models are built by analyzing extensive datasets that include variables like weather conditions, fuel properties, and topography. However, empirical models have limitations. Their accuracy depends on the quality and representativeness of the historical data used to train them, and they may struggle with scenarios that weren't considered in the dataset.

Table 1 Wildfire behaviour models and simulators

Model type	Models or simulators	References
Physical models	FIRETEC	[6]
	WFDS	[7]
	SCINTILLA	[8]
Empirical models	Rothermel	[9]
	BehavePlus	[10]
	FARSITE	[11]
	FOREFIRE	[12]
	Stochastic model	[13]

In practice, modern operational models are typically quasi-empirical, combining elements of physical and empirical models to achieve a compromise between scientific precision and easy implementation. Rothermel's model, first introduced by Richard C. Rothermel in 1972 [9], remains the most widely used wildfire spread model. Despite undergoing various corrections and enhancements over the years, it continues to serve as the core of wildfire simulation tools such as BehavePlus [10] and FARSITE [11]. Yet, authors in [13] introduced a stochastic forest fire growth model that complements the models discussed previously. Unlike deterministic approaches that yield the same outcome for fixed inputs, this stochastic model captures randomness in fire spread, including variability due to local weather, topography, and fire spotting. It operates on two conditions: burning and burnt states. Based on local conditions and probabilistic rules, the model includes time-varying weather effects and wind direction. Simulation results from a real fire event demonstrated its capacity to replicate complex dynamics. Table 1 summarizes the wildfire behavior models and simulators discussed.

Despite decades of progress in wildfire modeling, there remains a need for fast models that can be integrated into modern autonomous systems. For example, in Unmanned Aerial Vehicle (UAV) guidance systems operating within wildfire environments, the ability to simulate fire propagation rapidly is essential since agents must evaluate numerous potential situations in real time [14]. However, existing models are computationally intensive and too slow to support such real-time decision-making requirements. To address this gap, this research proposes a Convolutional Long Short-Term Memory (ConvLSTM) based model that approximates wildfire propagation dynamics. The neural network is trained on data generated using a deterministic Cellular Automata (CA) model and is designed to produce accurate predictions of fire spread. Existing work on deep-learning wildfire models has generally focused on direct prediction rather than emulating a specific wildfire model while providing substantial computational speed-ups.

While some recent works [15] introduced hierarchical CNNs capable of continuous time fire spread prediction, it has been shown that the most suitable neural architecture for such spatio-temporal prediction tasks is the ConvLSTM networks, due to their ability to capture spatial and temporal dependencies within sequential data [16]. In their work, Burge *et al.* [16] demonstrated that ConvLSTMs can successfully reproduce realistic fire dynamics, outperforming traditional CNNs and autoencoder architectures in both accuracy and stability, especially using auto-regressive multi-step predictions. Meanwhile, in [17], the authors extended this concept by incorporating attention mechanisms into ConvLSTM architectures to better capture both local and long-range dependencies and improve predictive fidelity.

In this study, and unlike previous approaches that concentrate on predicting fire intensity, our model specifically focuses on fire versus non-fire state prediction. Consequently, both the network architecture and training strategy, including the loss formulation and data pre-processing framework, are designed to optimize discrete fire state transitions. Furthermore, beyond predictive fidelity, we use ConvLSTMs also to serve as a real-time simulation foundation within intelligent planning architectures. The proposed ConvLSTM structure can further be integrated as a real-time simulation foundation in UAV guidance systems contexts for wildfire monitoring, where every time-step requires rapid inference [18].

III. Theoretical Background

A. Convolutional Neural Networks (CNNs)

CNNs are deep learning models designed to process structured grid data, most notably images. Their main advantage lies in automatically learning relevant features directly from raw input. CNNs achieve this by applying convolution operations with filters that capture local patterns such as edges, textures, and shapes. These features are progressively combined across layers, allowing the network to form hierarchical representations that are well-suited for tasks like image classification, recognition, and prediction. A typical CNN architecture alternates between convolutional and pooling layers to extract and condense information while reducing computational cost. Max pooling, in particular, enhances translation invariance and suppresses noise by preserving only the strongest activations in each region. Finally, the extracted features are flattened and passed through fully connected layers, where complex non-linear relationships are learned. Training through backpropagation refines these weights over multiple iterations, enabling CNNs to distinguish dominant patterns and classify inputs.

CNNs have also been applied to environmental monitoring, particularly in modeling and predicting the propagation of wildfires. By treating satellite imagery, topographic maps, and meteorological data as structured grid inputs, CNNs can learn spatial patterns associated with fire spread, such as vegetation density, wind direction, and terrain slope. The hierarchical feature extraction enables the network to identify high-risk zones and predict the likely path of a wildfire over time.

B. Long Short-Term Memory (LSTM) Networks

In time-series modeling, Long Short-Term Memory (LSTM) networks represent one of the most successful extensions of Recurrent Neural networks (RNNs). They demonstrated robustness in capturing long-range dependencies, overcoming common issues faced by traditional RNNs such as vanishing or exploding gradients [19]. The key innovation in LSTMs is the *memory cell*, denoted as c_t, which functions as a dedicated storage mechanism. This cell is designed to accumulate state information over extended time horizons, allowing the network to preserve and recall important contextual signals. The flow of information into, within, and out of the cell is governed by three parameterized gating mechanisms: the input gate i_t, the forget gate f_t, and the output gate o_t. At each time step t, given the input vector x_t and the previous hidden state h_{t-1}, the gates are computed as follows:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i), \tag{1}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f), \tag{2}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o), \tag{3}$$

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c). \tag{4}$$

Here, $\sigma(\cdot)$ denotes the sigmoid activation, and $\tanh(\cdot)$ is the hyperbolic tangent function. W and U denote the weights for input and previous cell output, and b are the bias terms. The cell state c_t is updated by combining the past memory c_{t-1} with the candidate memory $\tilde{c}_t$ as follows:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t. \tag{5}$$

where $\odot$ denotes element-wise multiplication. The forget gate f_t decides how much of the previous state c_{t-1} should be retained, while the input gate i_t controls the amount of new information to be stored. Finally, the hidden state h_t, which serves as the output of the LSTM at time step t, is obtained by filtering the updated memory cell through the output gate:

$$h_t = o_t \odot \tanh(c_t). \tag{6}$$

The gating mechanism in LSTM networks enables them to selectively manage the flow of information by retaining relevant data, discarding irrelevant details, and exposing critical information at the appropriate time. Figure 2 represents a visual overview of the LSTM unit, emphasizing the flow of information through its internal structure. It illustrates how inputs, previous states, and gating signals interact within the unit and how the pathways of addition, multiplication, and activation functions combine to form the recurrent dynamics.

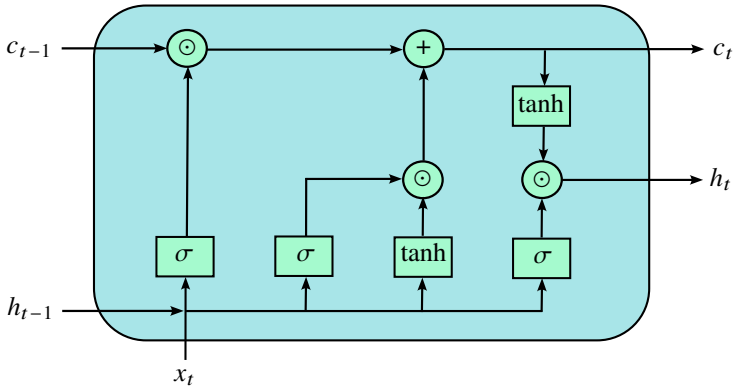


Fig. 2 Architecture of a LSTM Unit

C. Convolutional Long Short-Term Memory (ConvLSTM) Networks

In this section, we present the Convolutional Long Short-Term Memory Networks (ConvLSTM) framework [20]. ConvLSTM extends the traditional LSTM model by incorporating convolution operations into its gating mechanisms. In this framework, convolution plays two essential roles, similar to its function in standard CNNs: feature integration and feature transformation, but with recurrent memory for temporal modeling. Feature integration is achieved through fixed-size convolutional filters with trainable weights that aggregate information from local spatial neighborhoods. This allows ConvLSTM to capture spatial dependencies, setting it apart from conventional LSTM models that lack this capability. Feature transformation, on the other hand, is accomplished through linear mappings followed by nonlinear activations, enabling the network to refine integrated features into more complex, nonlinear representations. Together, these processes enhance ConvLSTM's ability to model spatio-temporal sequences and predict future states with greater accuracy.

The outputs, hidden states, and gates of ConvLSTM are represented as 3D tensors, where the last two dimensions correspond to spatial rows and columns. Unlike traditional LSTM, ConvLSTM performs convolutions instead of fully connected multiplications during both input-to-state and state-to-state transitions. The ConvLSTM equations can be expressed as:

$$i_t = \sigma(W_i \otimes X_t + U_i \otimes H_{t-1} + b_i)^*, \tag{7}$$
$$f_t = \sigma(W_f \otimes X_t + U_f \otimes H_{t-1} + b_f), \tag{8}$$
$$o_t = \sigma(W_o \otimes X_t + U_o \otimes H_{t-1} + b_o), \tag{9}$$
$$\tilde{C}_t = \tanh(W_c \otimes X_t + U_c \otimes H_{t-1} + b_c), \tag{10}$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t, \tag{11}$$
$$H_t = o_t \odot \tanh(C_t). \tag{12}$$

Note that the fully connected structure of standard LSTM networks is limited in its ability to capture rich spatial context when processing spatio-temporal data, such as 2D temporal sequences in fire modelling. ConvLSTM addresses this limitation by replacing the fully connected operations with convolutional ones, thereby enabling the model to effectively learn both spatial and temporal dependencies. The overall common model is constructed as a sequence of ConvLSTM blocks, each processing a series of input lattices that represent successive time steps. Within each block, convolutions transform the input sequence into a new spatio-temporal representation. The block's output preserves the spatial dimensions of the input while expanding the channel dimension according to the number of convolution filters applied. Figure 3 illustrates the ConvLSTM architecture, where inputs pass through multiple ConvLSTM and Batch Normalization layers, followed by a convolution layer (2D for predicting next time step and 3D to predict a sequence of predictions simultaneously).

*The symbol $\otimes$ denotes convolution.

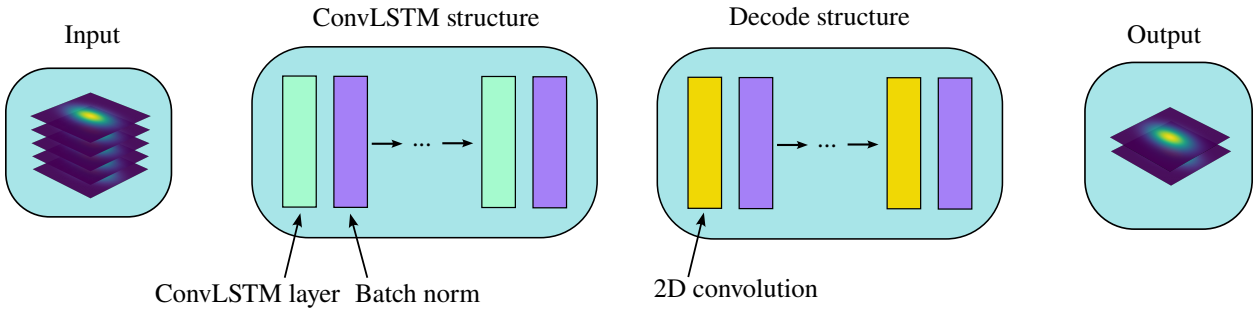


Fig. 3 Architecture of a ConvLSTM network

IV. Methodology

A. Data generation

In our experiments, we used wildfire propagation data generated from a customized CA simulator (See *Appendix*). Our model deterministically evaluates fire ignition based on local fuel density, wind influence, slope, and moisture. This model is conceptual and is not calibrated to replicate specific real-world fire events but rather to simulate general wildfire behaviors under controlled conditions. The use of this model allows us to explore the impact of various parameters on fire dynamics in a methodical manner, which is essential for testing and training the ConvLSTM models. Each data sample in this set represents a propagation of 400 different scenarios of 130×130 grid map over 150 time steps.

To capture a wide range of wildfire dynamics, we run multiple simulations with randomized initial conditions, including diverse elevation maps, different fuel density and moisture, diverse ignition point locations, and varying wind directions. The resulting dataset enables the neural model to learn complex and diverse fire propagation patterns across heterogeneous landscapes. The simulator's deterministic nature ensures reproducible data, which is essential for consistent training and evaluation of the model. Each scenario begins with a single or multiple ignition points, randomly placed on the grid, and evolves over the time steps, allowing the model to capture both short-term fire spread and long-term behavior under varying conditions. This structured yet varied dataset benefits the training part, enabling the model to generalize across a spectrum of wildfire scenarios.

B. Network framework

The ConvLSTM-based network designed for this study consists of a sequence of ConvLSTM blocks followed by a 2D convolutional decoder, as illustrated in Figure 3. The architecture is intentionally kept compact using 1, 2, or 4 ConvLSTM layers to balance predictive performance with computational efficiency, particularly given the real-time and large-scale nature of wildfire forecasting. Deeper networks are found to offer higher computational cost. The selected design therefore reflects an empirically grounded trade-off between model complexity and operational feasibility.

Table 2 ConvLSTM and decoder layer parameters

Layer	Parameter	Specification
ConvLSTM	Kernel size	3×3
	Padding	1
	Activation	Tanh (candidate $\tilde{C}_t$)
	Recurrent Activation	Sigmoid (gates i_t, f_t, o_t)
Convolution	Kernel size	1×1
	Padding	1
	Activation	Sigmoid

Within each ConvLSTM block, a 3×3 convolutional kernel is used to capture local spatial interactions associated with short-range fire spread. Although larger kernels (e.g., 5×5 or 7×7) can increase the receptive field, they also

introduce more parameters and a higher risk of over-smoothing fire boundaries. Preliminary experiments indicated that larger temporal kernels did not yield significant accuracy improvements. For these reasons, the 3×3 kernel was chosen as the most effective balance between spatial sensitivity and model generalization.

A 1×1 convolutional layer is used in the decoder to transform the final hidden representation into the output map while preserving spatial resolution. More complex decoding schemes, such as stacked 3D convolutional blocks, were considered but deemed unsuitable for two reasons: the first is that 3D convolutions considerably increase computational cost, and the second is that the temporal dependencies are already explicitly modeled within the ConvLSTM recurrent state. Table 2 summarizes the parameters used in the ConvLSTM blocks and decoder. Overall, the adopted architecture provides an effective compromise between stability and computational efficiency, enabling the network to capture both the spatial continuity of fire and its temporal evolution while remaining suitable for real-world deployment scenarios.

Table 3 Proposed ConvLSTM network configurations for tests

Model	ConvLSTM layers	Hidden Dimension	Decode layers
A	1	16	1
B	1	32	1
C	1	64	1
D	2	16	1
E	2	32	1
F	2	64	1
G	4	16	1
H	4	32	1
I	4	64	1

Table 3 details the structure of nine ConvLSTM architectures. This allows us to test several configurations in order to evaluate the influence of both network depth (number of ConvLSTM layers) and model capacity (hidden dimension size) on the ability to learn and predict fire propagation dynamics. By systematically varying these parameters, we can identify an optimal balance between model complexity and predictive performance.

C. Sequential prediction

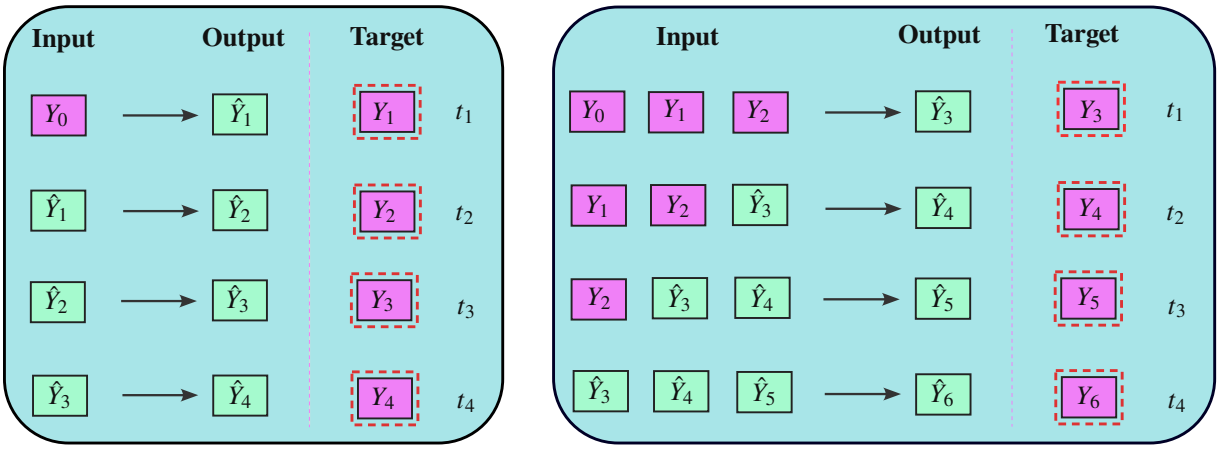
To assess the temporal generalization capability of the ConvLSTM models, we adopt an *auto-regressive prediction* framework. In this setting, the model is provided with an initial sequence of frames and iteratively predicts future fire and fuel map states. At each prediction step, the model output is appended to the temporal input window, replacing the oldest frame. This recursive strategy enables long-term forecasting without access to future ground-truth data during prediction. The approach naturally extends to multi-step prediction by defining a memory length parameter l, which specifies the number of past frames retained in the model's input sequence. Consequently, for a given trained model and input lattice, the model's prediction serves as the updated lattice, which is subsequently incorporated into the next iteration of auto-regressive prediction as detailed in Algorithm 1.

Algorithm 1 Auto-regressive sequential prediction with ConvLSTM

Require: Trained model $\mathcal{M}$, initial history $\mathcal{H}^{(0)} = \{Y_0, \ldots, Y_{l-1}\}$ of length l, prediction horizon T
Ensure: Predicted sequence $\hat{\mathcal{Y}} = \{Y_0, \hat{Y}_1, \hat{Y}_2, \ldots, \hat{Y}_T\}$

$\hat{\mathcal{Y}} \leftarrow \mathcal{H}^{(0)}$
$\mathcal{H} \leftarrow \mathcal{H}^{(0)}$
for $t = l$ **to** T **do**
  $\hat{Y}_t \leftarrow \mathcal{M}(\mathcal{H})$ ▷ Predict next frame using ConvLSTM network
  $\hat{\mathcal{Y}} \leftarrow \hat{\mathcal{Y}} \cup \{\hat{Y}_t\}$
  $\mathcal{H} \leftarrow \text{concat}(\mathcal{H}_{[1:l]}, \hat{Y}_t)$ ▷ Slide window forward and append new prediction in input
end for
return $\mathcal{Y}$

Figure 4 illustrates two examples of the auto-regressive prediction process with different memory lengths l. In both cases, the purple frames correspond to the real observed data used as the model's input context, while the green frames represent the ConvLSTM's predicted outputs. Figure 4(a) shows a short-term forecasting scenario with $l = 1$, where predictions rely solely on the most recent frame. In contrast, Figure 4(b) depicts the case with $l = 3$, where the model incorporates a longer temporal history to generate the next frame.



((a)) Short-term forecasting with $l = 1$ **((b)) Long-term forecasting with $l = 3$**

Fig. 4 Implementation of multi-step auto-regressive prediction process. Ground truth frames are presented in purple and predicted frames in green.

D. Loss function and evaluation metrics

The model for fire propagation prediction is trained using Binary Cross-Entropy (BCE) loss, which quantifies the discrepancy between the predicted fire map and the ground truth map at each time-step. BCE loss is particularly well-suited for pixel-wise binary classification tasks, where each grid cell represents the presence or absence of fire. Formally, the BCE loss is defined as:

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N_H \times N_W} \sum_{i=1}^{N_H \times N_W} \Big[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \Big]. \tag{13}$$

where $y_i \in \{0, 1\}$ is the ground-truth label for pixel i, $\hat{y}_i \in [0, 1]$ is the predicted probability of fire at that pixel, and $N_H \times N_W$ represents the total number of pixels in each frame. While fire propagation could alternatively be formulated as a regression problem to predict continuous fire intensity values, this work focuses on a binary classification framework (fire versus non-fire). Consequently, BCE loss is preferred over regression-based losses such as the Mean Squared Error (MSE) since BCE more effectively captures the probabilistic nature of fire occurrence at each spatial location.

In order to evaluate the ConvLSTM model in the wildfire auto-regressive fire-propagation prediction task, three types of metrics are considered: pixel-wise accuracy, spatial-overlap measure, and boundary-focused metric. Together, these metrics provide a comprehensive understanding of model performance, capturing not only the correctness of individual pixel prediction but also the structural consistency and geometric fidelity of the predicted fire regions over time.

In particular, the interpretation of these metrics in wildfire modeling differs from conventional computer vision segmentation tasks. Wildfire spread prediction involves dynamic temporal boundary evolution and error accumulation over multiple auto-regressive steps. As a result, numerical performance ranges reported in the wildfire modeling literature tend to be lower (for overlap metrics) or higher (for displacement metrics) than those typically observed on standard image segmentation studies. The ranges presented in the following tables are therefore based on empirical values reported in recent deep learning–based wildfire spread studies [21–23].

Pixel Accuracy (PA): A fundamental metric for evaluating how accurately the model assigns labels to pixels is the *pixel accuracy* (PA), defined as the proportion of predicted pixels that match the ground truth:

$$\mathrm{PA} = \frac{|(P \cap G) \cup (P^c \cap G^c)|}{N_H N_W} = \frac{\sum_{i=1}^{N_H N_W} (1 - |\hat{y}_i - y_i|)}{N_H N_W}. \tag{14}$$

Where:

- P is the set of pixels predicted as fire by the model
- G is the set of pixels labeled as fire in the ground truth
- $(\cdot)^c$ denotes set complement

While simple and interpretable, PA can be misleading in wildfire modeling due to extreme class imbalance (fire pixels typically covering less than 5% of the domain). High PA may result simply from correctly predicting background pixels rather than accurately predicting fire spread. For this reason, spatial overlap metrics such as the Jaccard Similarity Coefficient (JSC) are preferred [22].

Jaccard Similarity Coefficient (JSC): A widely used segmentation metric is the *Jaccard Similarity Coefficient* (JSC), also known as *Intersection over Union* (IoU), originally introduced by Jaccard [24]. It quantifies the spatial agreement between predicted and ground-truth fire regions:

$$\mathrm{JSC} = \frac{|P \cap G|}{|P \cup G|} = \frac{\sum_{i=1}^{N_H N_W} (\hat{y}_i y_i)}{\sum_{i=1}^{N_H N_W} (\hat{y}_i + y_i - \hat{y}_i y_i)}. \tag{15}$$

In wildfire propagation, JSC is a more significant indicator than PA because it measures whether the model predicts the correct shape and extent of the fire. However, JSC values in wildfire prediction are substantially lower than in image segmentation since the data is temporal, fire moves rapidly, and auto-regressive multi-step forecasting accumulates small spatial errors over time. As a result, even slight misalignments in early predictions can grow into large discrepancies in later steps, reducing average JSC despite the model capturing general fire behavior.

Values above 70% represent strong spatial overlap, but in practice these occur only for short prediction windows or under relatively stable fire conditions (our case here). More typically, wildfire models achieve JSC values in the $50\% - 70\%$ range, which corresponds to moderate overlap and indicates that the model is capturing the general direction and extent of spread even if the precise perimeter is imperfect. Scores below 50% signify weak overlap and generally indicate that the prediction diverges substantially from the observed fire growth.

Table 4 Interpretation of overlap-based metrics for fire propagation prediction

Metric	Role	Range	Interpretation
PA	Pixel-wise correctness	> 0.95	High accuracy
		$0.9 – 0.95$	Moderate accuracy
		< 0.9	Low accuracy
JSC	Spatial overlap	> 0.70	Strong overlap
		$0.50 – 0.70$	Moderate overlap
		< 0.50	Weak overlap

Hausdorff Distance (HD): To complement region-overlap metrics, boundary-focused measures such as the *Hausdorff Distance* (HD) [25, 26] quantify the geometric displacement between predicted and actual fire fronts. It provides an effective way to assess how similar the edge contours of two objects are, quantifying the degree to which their boundaries match. It is used to evaluate how closely the predicted fire front aligns with the ground truth.

$$\begin{cases} \mathrm{HD} = \max\{h(P,G), h(G,P)\} \\ h(P,G) = \max_{p \in P} \min_{g \in G} \|p - g\|_2 \\ h(G,P) = \max_{g \in G} \min_{p \in P} \|g - p\|_2 \end{cases} \tag{16}$$

Here, $\|.\|_2$ is the Euclidean distance. HD captures the maximum boundary error, making it especially relevant for fire front estimation where small spatial displacement can significantly change spread outcomes. Moreover, since HD is highly sensitive to outliers, many studies use a percentile version such as HD_{95} [25], which we also adopt.

Table 5 summarizes the interpretation of different ranges of the HD in the context of fire front prediction. Very small values (below 5%) of the grid size indicate that the predicted perimeter is almost perfectly aligned with the ground-truth fire boundary, reflecting highly accurate spatial localization. Values between 5% and 10% still correspond to good performance, with only minor spatial deviations. Moderate discrepancies (10% to 20%) remain acceptable, as the overall fire spread pattern is preserved. In contrast, HD values above 20% indicate substantial boundary mismatches and poor predictive quality.

Table 5 Interpretation of Hausdorff Distance (HD) for fire-front prediction

HD (% of grid size)	Interpretation
$< 5\%$	Boundary almost perfectly aligned with ground truth
$5\% - 10\%$	Minor displacement, propagation direction and structure preserved
$10\% - 20\%$	Moderate spatial error, general fire-spread pattern still captured
$> 20\%$	Large deviation, fire propagation poorly reproduced

E. Training

All models were implemented using *PyTorch 2.6.0* and trained on a workstation equipped with an *NVIDIA RTX 3080 GPU*. Further, to avoid overfitting, an early stopping strategy was employed during training. The validation loss was monitored after each epoch, and training was terminated if no improvement was observed for 10 consecutive epochs. The model parameters corresponding to the epoch with the lowest validation loss were retained for the final evaluation. This strategy ensures that the model generalizes effectively, thereby preserving its ability to perform well on unseen data.

In order to improve convergence and optimize the learning process, the Learning Rate (LR) step decay *ReduceLROnPlateau* scheduler was employed. This adaptive mechanism dynamically adjusts the LR based on the evolution of the validation loss. When the rate of improvement in training performance slows down, the LR is gradually reduced. Lowering it enables the optimizer to make finer, more controlled adjustments within the parameter space. This helps prevent premature convergence or oscillation during optimization, allowing the model to more thoroughly explore the loss space and settle into more stable minima that strengthen generalization.

Formally, given a monitored validation loss v_t at epoch t, the LR is updated according the following rule:

$$\eta_{t+1} = \begin{cases} \eta_t \times \gamma, & \text{if } v_t \geq \min(v_{t-p}, \ldots, v_t) \text{ for } p_a \text{ consecutive epochs,} \\ \eta_t, & \text{otherwise,} \end{cases}$$

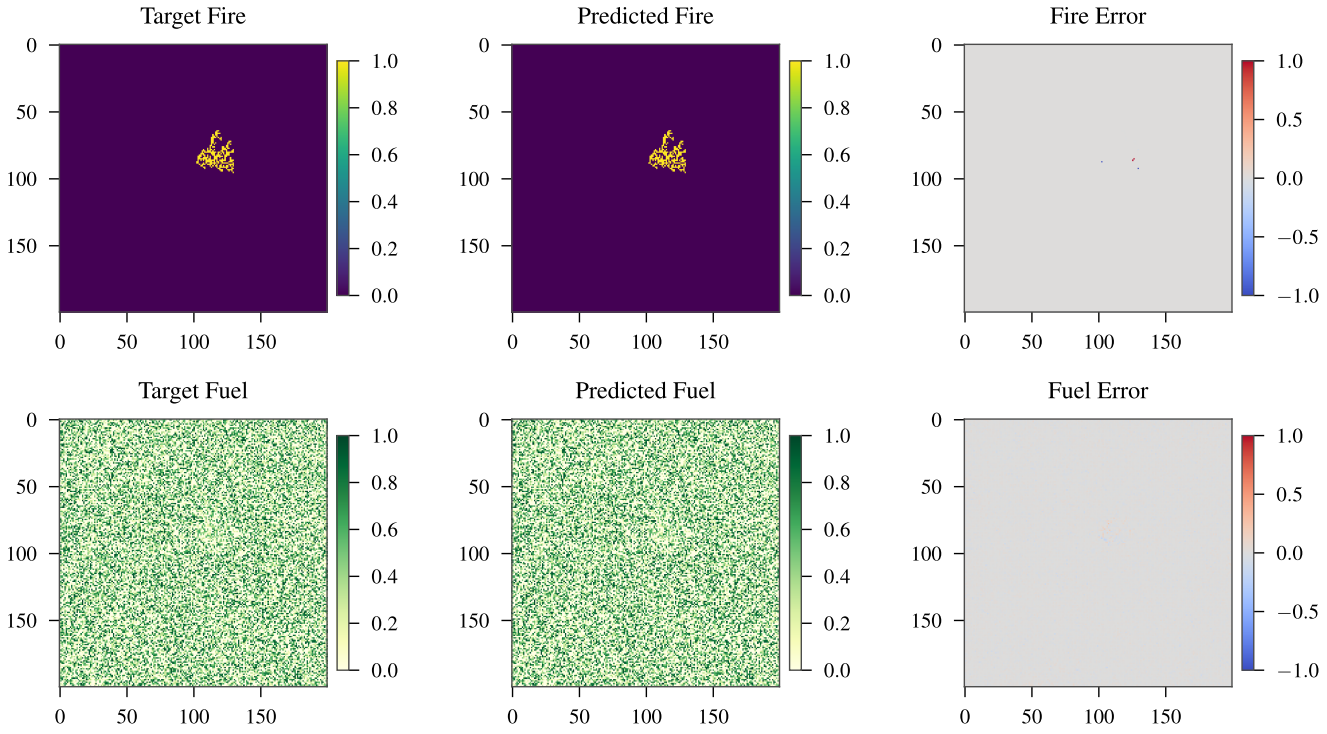
where:

- η_t and η_{t+1} denote the LRs before and after the update, respectively.
- γ is the reduction factor.
- p_a represents the patience parameter (the number of epochs without improvement before the rate is reduced).
- v_t is the monitored validation loss.

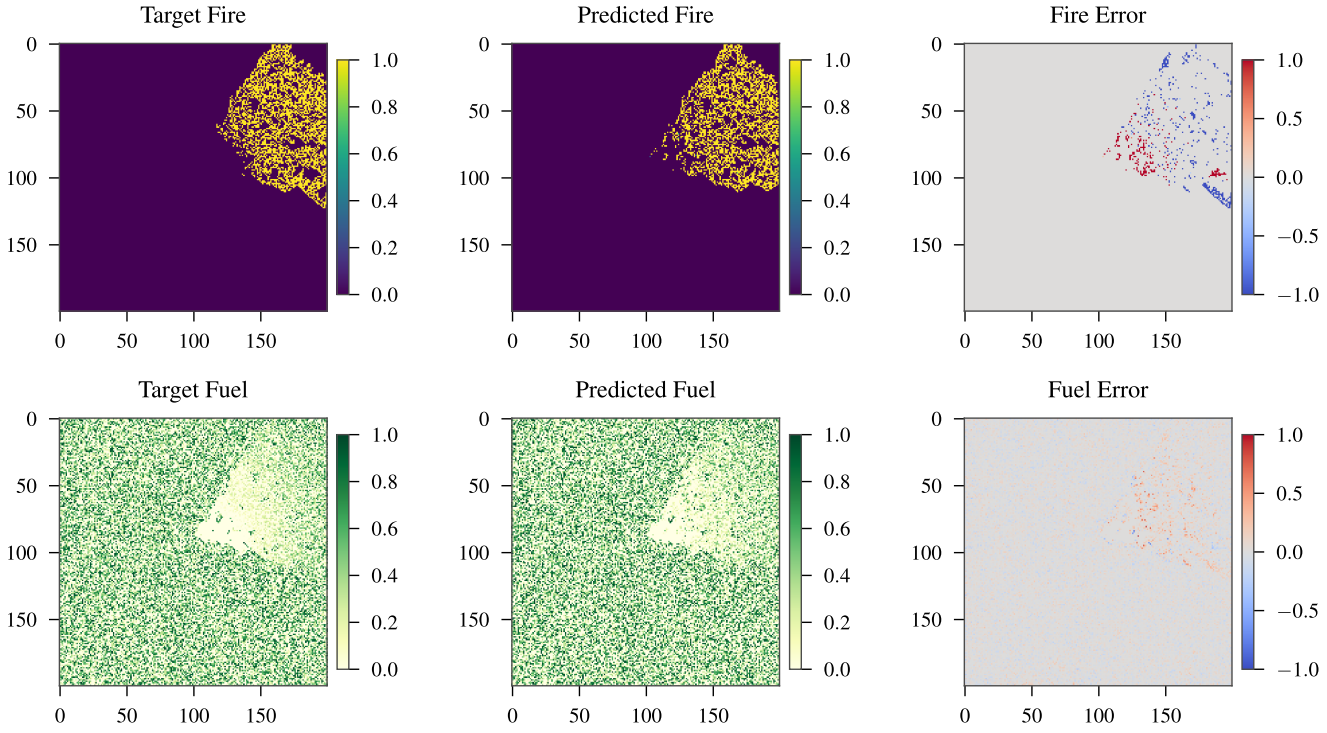
This adaptive scheduling mechanism allows the optimizer to maintain a relatively high LR during periods of rapid progress, favoring efficient exploration of the parameter space. Once progress slows, the LR is reduced, enabling finer parameter updates and a smoother descent toward convergence. Therefore, early stopping and adaptive LR scheduling help ensure that the resulting model achieves strong generalization without unnecessary computational cost.

V. Numerical experiments and analysis

In order to evaluate the performance of the proposed ConvLSTM-based model, a set of experiments was carried out using the dataset produced as described in Section IV.A. This dataset provides controlled fire spread patterns under varying environmental conditions (wind, moisture, elevation, fuel).



((a)) Predicted fire spread after few time steps



((b)) Predicted fire spread after several time steps

Fig. 5 Visualization of prediction errors between the target and model outputs for a selected test sample. The figure shows the spatial distribution of fuel and fire, with the rightmost panel presenting the error map. Color intensity represents the magnitude of the prediction error: red indicates over-predicted fire (fire predicted where none exists), and blue indicates under-predicted fire (missed fire).

Figure 5 presents for a specific test sample, a visual comparison between the ground-truth fire propagation maps generated by the simulator and the corresponding predictions produced by the trained ConvLSTM model. It clearly indicates that the proposed model is capable of capturing the spatial dynamics of wildfire spread, including the main direction of propagation, the shape of the fire front, and the interaction between neighboring cells. The model maintains a high degree of spatial coherence with the ground truth, even after multiple recursive predictions, demonstrating its ability to learn and replicate complex, non-linear dependencies in fire behavior.

As can be seen, the network is able to reproduce the global evolution of the wildfire. The model captures both the main direction of spread and the deformation of the fire perimeter as it interacts with heterogeneous fuel and topography. Errors are concentrated near the leading edge of the fire front, and grow gradually over long horizons as small spatial misalignment accumulate across auto-regressive steps. This behavior is expected for sequence-to-sequence forecasting models and motivates the more detailed quantitative analysis presented next.

A. Model performance

In order to evaluate the ConvLSTM model performance, we rely on the nine configurations presented in Table 3. Each configuration is evaluated over ten independent scenarios of 200×200 map designed to reflect diverse fire propagation conditions. For every scenario, the metrics PA, JSC, and HD are computed at each time-step and then averaged across the full temporal sequence to obtain a single representative value per scenario. These are subsequently averaged over the ten scenarios to produce a robust estimate of the model's predictive capability. The results are reported in Table 6, Figure 6, and 7. Overall, the high PA value is promising, but it does not necessarily imply that the model's performance is sufficient. Therefore, it is important to also examine the JSC. The JSC values indicate a strong spatial agreement between the predicted and ground-truth fire extents. At the same time, the HD values fall within the range classified as good according to the thresholds defined in Table 5.

Table 6 Model performance metrics

Model	PA	JSC	HD
A	0.988	0.724	6.098
B	0.990	0.761	5.492
C	0.990	0.764	5.163
D	0.987	0.731	6.687
E	0.988	0.781	5.153
F	0.993	0.817	3.378
G	0.988	0.758	5.779
H	0.991	0.790	4.682
I	0.995	0.836	2.525

The results clearly show that increasing the model complexity, whether by adding more layers or more hidden states, leads to improved predictive performance. Deeper and wider configurations can capture more complex spatial-temporal dependencies, improving metric values. This trend highlights the capacity of larger ConvLSTM models to better encode the dynamics that characterize wildfire spread. However, the trade-off between model size and computational demand is critical. While larger configurations achieve higher accuracy, they also produce significantly higher computational cost. Thus, a comparison of runtime performance across model sizes is discussed in more detail in Subsection V.D.

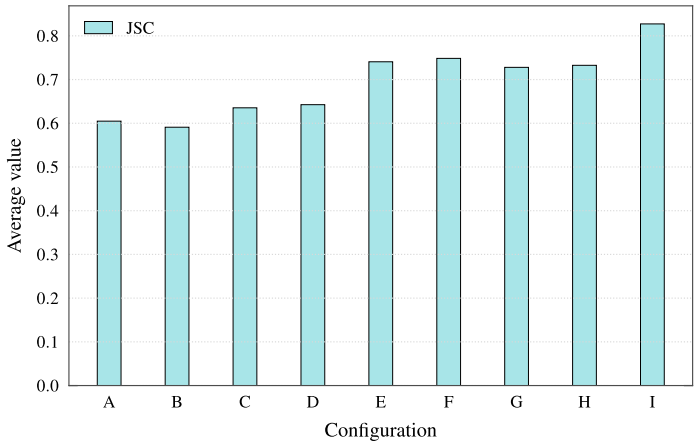


Fig. 6 Average JSC for each ConvLSTM configurations introduced in Table 3

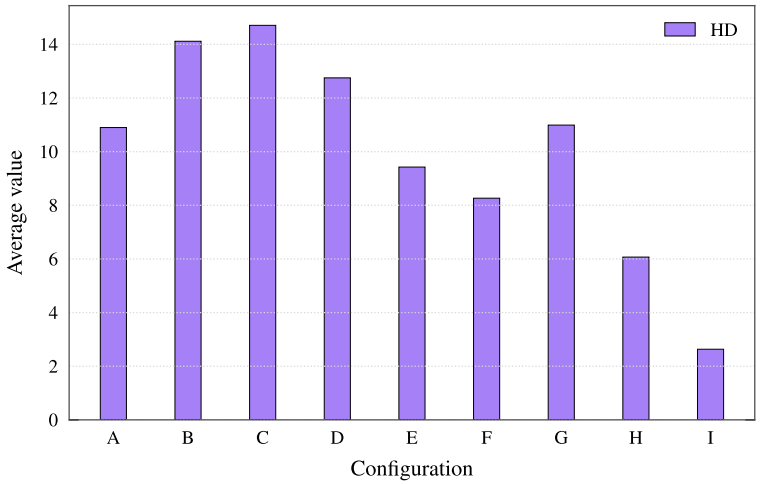


Fig. 7 Average HD for each ConvLSTM configurations introduced in Table 3

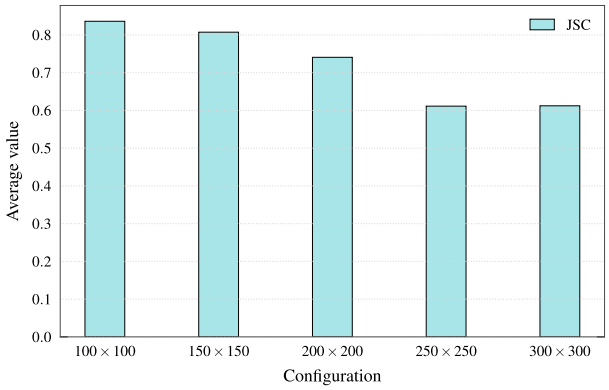
B. Scalability

An important requirement for practical deployment is the ability to handle grids of different sizes without retraining. To examine this aspect, we evaluate the trained ConvLSTM model on five spatial resolutions: 100×100, 150×150, 200×200, 250×250 and 300×300 cells. In addition to spatial scaling, we also adjust the temporal horizon proportionally, using 100, 125, 150, 175 and 200 auto-regressive time steps, respectively. For each resolution, ten independent test scenarios are generated, and propagated by the ConvLSTM over the corresponding number of time steps. For every scenario, the evaluation metrics are computed at each time-step and then averaged over the full prediction sequence. Finally, these per-scenario averages are aggregated to produce a mean score over the ten test scenarios. This setup allows us to assess how well the model generalizes across both spatial and temporal scales. Table 7 reports metrics PA, JSC and HD for all scenario and resolutions. The corresponding average values per resolution are shown in Figure 8.

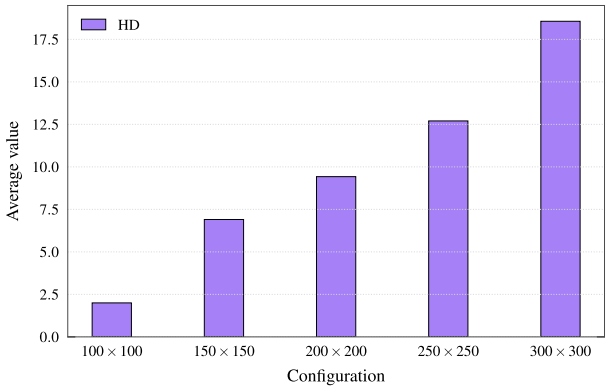
Table 7 Performance metrics (PA, JSC, HD) for 10 test scenarios with multiple grid resolutions

No	100x100			150x150			200x200			250x250			300x300		
	PA	JSC	HD	PA	JSC	HD	PA	JSC	HD	PA	JSC	HD	PA	JSC	HD
1	1.000	0.701	0.756	1.000	0.899	0.920	0.982	0.792	11.869	0.981	0.517	28.379	0.981	0.305	47.383
2	0.995	0.931	0.555	0.990	0.844	8.738	0.962	0.525	30.306	0.995	0.912	0.876	0.999	0.552	6.189
3	0.996	0.948	0.329	0.993	0.944	0.377	0.991	0.920	0.585	0.997	0.809	6.626	0.984	0.612	26.642
4	0.969	0.770	6.062	0.970	0.765	13.556	0.991	0.750	4.211	0.998	0.752	12.236	0.999	0.698	11.248
5	1.000	0.745	0.000	0.966	0.581	20.625	0.974	0.530	23.266	1.000	0.473	2.836	0.998	0.667	18.692
6	0.998	0.929	0.804	0.996	0.821	4.870	0.999	0.819	7.107	0.990	0.664	15.390	0.995	0.226	27.294
7	1.000	0.816	0.000	0.997	0.687	7.556	0.998	0.867	3.686	0.981	0.606	24.757	1.000	0.922	0.413
8	0.991	0.855	2.636	0.997	0.923	1.367	0.980	0.770	11.448	0.998	0.317	16.418	0.987	0.557	30.523
9	0.999	0.900	1.099	0.997	0.761	7.350	1.000	0.710	1.471	0.998	0.657	14.350	1.000	0.711	13.693
10	0.971	0.767	7.755	0.992	0.847	3.690	1.000	0.722	0.311	1.000	0.404	5.162	0.993	0.872	3.567

Overall, the results show that both PA and JSC remain high across all resolutions, with only a modest decrease as the grid becomes larger. The average JSC consistently falls within the 0.6–0.8 range, indicating reliable overlap between the predicted and ground-truth fire even at a resolution of 300×300 cells. In contrast, the HD metric increases with grid size, which is expected since the same physical displacement corresponds to a greater number of pixels on finer grids. Nonetheless, the HD values remain within the threshold considered good, confirming that the geometric accuracy of the predicted contours is still acceptable. These findings demonstrate that the ConvLSTM generalizes well across different spatial discretizations, maintaining stable predictive performance while adapting to varying map scales without requiring retraining. Figure 8 provides an aggregated view of these results across the ten scenarios, allowing for a clear comparison of model behavior at different resolutions.



((a)) Average JSC with different grid sizes



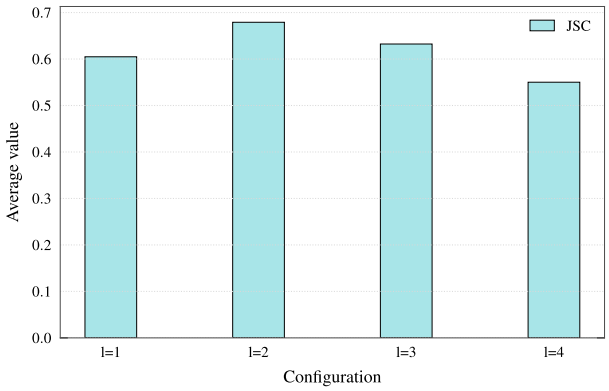
((b)) Average HD with different grid sizes

Fig. 8 Scalability of the ConvLSTM model with respect to spatial resolution. Each bar corresponds to the average over the ten scenarios reported in Table 7

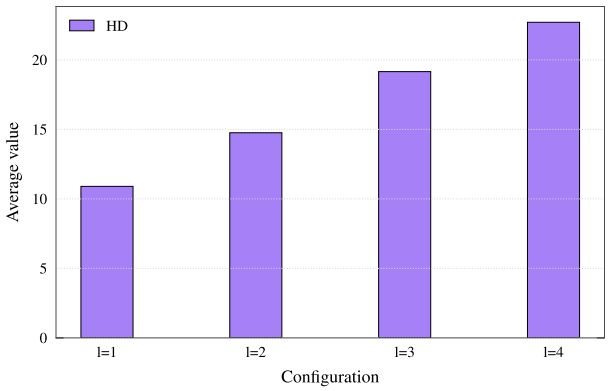
C. Multi-horizon prediction performance

The impact of the input sequence length l on the ConvLSTM forecasting performance was evaluated by comparing four configurations with $l \in \{1, 2, 3, 4\}$ past frames. Figure 9 reports the JSC and HD scores for each configuration. Figure 9(a) shows that JSC performance improves from $l = 1$ to $l = 2$, but then progressively degrades for $l = 3$ and $l = 4$. On the other hand, Figure 9(b) shows that HD performance degrades after $l = 1$. This suggests that the inclusion of less past frames provides sufficient temporal context for capturing short-term wildfire dynamics, while longer input sequences introduce redundant or noisy temporal information that impacts the model's ability to predict precise and accurate fire propagation.

This behavior aligns with the nature of wildfire evolution, where the state at time $t + 1$ is predominantly shaped by the most recent fire-front geometry and local environmental conditions, resulting in limited long-term temporal dependencies. Notably, the finding that fewer past frames offer the best balance between JSC and HD scores also supports our goal of accelerating wildfire spread simulations. Increasing the input length not only fails to improve predictive quality but also raises computational cost, memory usage, and data-processing latency. Therefore, adopting a minimal yet informative temporal window is advantageous: it preserves predictive performance while reducing inference time. These results highlight the sensitivity of temporal models to the length of their input history. In the context of wildfire forecasting, providing more frames does not necessarily translate into richer or more useful temporal information. Instead, excessive temporal depth may impact the information about the most relevant patterns.



((a)) Average JSC with different l



((b)) Average HD with different l

Fig. 9 Performance of the ConvLSTM with respect to the input sequence length l

D. Runtime and acceleration analysis

Figure 9 compares the computational time required to generate an entire fire-spread trajectory using the reference CA simulator with the runtimes obtained for each of the trained ConvLSTM configurations. As in the previous experiments, ten representative scenarios were evaluated, and the average runtime across these scenarios is reported for both the ConvLSTM models and the baseline CA implementation.

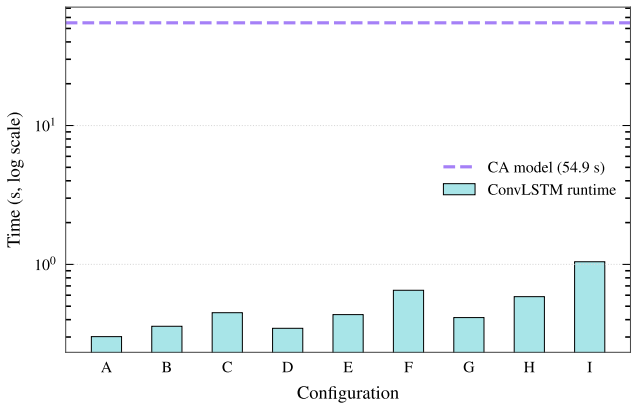


Fig. 10 Computational time required to generate a full propagation for each configuration in Table 4, compared with the time required by the CA simulator.

The results clearly demonstrate that the learned ConvLSTM models outperform the traditional CA simulator in terms of computational efficiency. As summarized in Table 8, all architectures achieve significant reductions in computation time. Configuration A yields the highest acceleration factor, running approximately 182× faster than the baseline CA simulator, which highlights the considerable benefit of using compact ConvLSTM structures. Similarly, configurations B, C, D, and E deliver strong performance, achieving speed-ups up to two orders of magnitude as well.

Configurations F and H achieve substantial acceleration, with speed-ups ranging from 84× to 94×. Despite employing deeper architectures and larger hidden sizes (up to 64 units), both configurations still outperform the baseline by a wide margin. Configuration I yields the smallest improvement, with an acceleration of about 53×. This result aligns with its increased architectural complexity, which includes the greatest number of stacked layers and the largest internal state size. Nevertheless, it still represents a considerable speed-up compared with the CA baseline. It is also worth noting that Configuration E (highlighted in bold in Table 8) offers the best trade-off between computational accuracy and computational cost.

Table 8 Approximate computational acceleration achieved by each ConvLSTM configuration compared to the baseline CA simulator

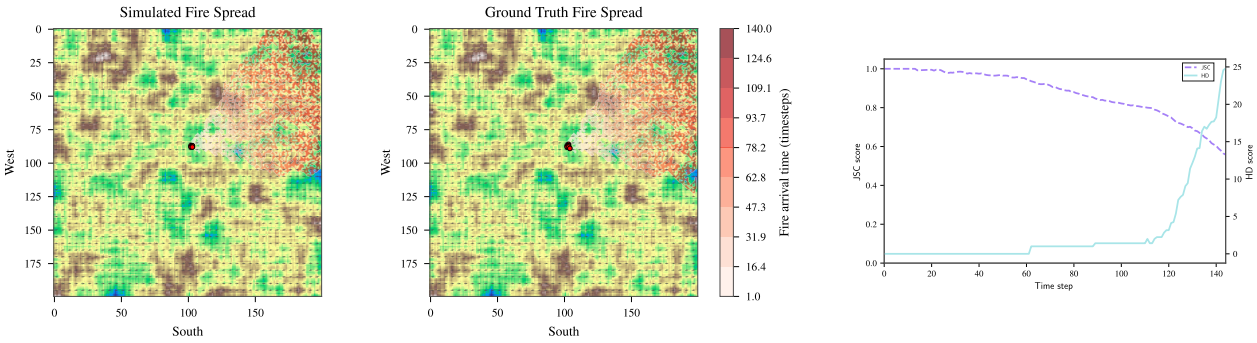
Configuration	Acceleration factor
A	≈ 182×
B	≈ 153×
C	≈ 122×
D	≈ 158×
E	**≈ 126×**
F	≈ 84×
G	≈ 133×
H	≈ 94×
I	≈ 53×

Overall, these findings confirm that ConvLSTM models can significantly accelerate large-scale or long-horizon simulations, enabling faster experimentation and quicker iteration cycles. This level of acceleration makes such models highly suitable for integration into real-time or near-real-time decision-making pipelines, including control, optimization, or interactive simulation frameworks. In our case, this opens the door to Reinforcement Learning (RL)-based workflows enabling faster learning cycles that wouldn't be feasible with the original CA simulator, especially including UAV-based guidance strategies for wildfire monitoring, where autonomous aerial agents learn to adapt their flight paths in response to evolving fire dynamics [27].

E. Limitations

The ConvLSTM-based wildfire propagation model was trained and evaluated on a generated dataset derived from our deterministic CA simulator. Although this synthetic dataset provides controlled and reproducible fire spread scenarios, it does not capture the physical realism in actual wildfire behavior. This underscores the need to assess how the model responds under more physically demanding fire conditions. Future work should therefore include validation against physically based wildfire simulators or observational datasets to properly evaluate the model's generalization capability beyond simplified synthetic environments.

Furthermore, the auto-regressive prediction framework introduces cumulative error over long forecasting horizons, as small inaccuracies at the fire boundary progressively accumulate across successive predictions. Figures 11(a) and 11(b) illustrate this limitation: although the model generates visually coherent fire progression and successfully captures the dominant propagation trends, discrepancies arise in the finer structural details of the fire pattern, as well as in the temporal evolution of overlap and boundary-based metrics. Deeper ConvLSTM configurations can partially mitigate these issues by increasing the model's representational capacity, but this improvement comes at the cost of higher computational demand. As shown in Figure 11(b), performance deteriorates most noticeably at the final time steps, particularly as the fire approaches full extinction, where the model tends to drift and accumulate larger deviations from the ground truth.



((a)) Fire trajectory: prediction vs. ground truth **((b)) Evolution of JSC and HD over time**

Fig. 11 Example of simulated wildfire spread and corresponding metric roll-out

VI. Conclusion and perspectives

In this work, we introduced a ConvLSTM-based model designed to approximate a wildfire propagation simulator while achieving significant reductions in computational cost. By training the network on large sets of simulated fire spread trajectories, the proposed approach successfully captures the dynamics of fire growth and produces multi-step predictions that closely match those of the original CA model. Our experiments demonstrate that compact ConvLSTM configurations can accelerate prediction by up to nearly one order of magnitude while maintaining acceptable accuracy, making them suitable candidates for real-time decision systems. Importantly, the speedup remains stable across different spatial resolutions, indicating that the computational gains persist when scaling to finer grids. Nevertheless, some limitations must be considered for future work. The ConvLSTM model relies entirely on training data generated by the underlying simulator and therefore adopts its structural modeling assumptions. Extending the approach to real wildfires will require creating new datasets, handling uncertainties in meteorological inputs, and accounting for vegetation heterogeneity, which are typically simplified in synthetic simulations.

Future research can extend this work in several directions. A first avenue involves exploring hybrid architectures that combine recurrent layers with attention mechanisms or transformer-based modules, enabling improved modeling of long-range dependencies. In addition, deploying the model with high-performance inference frameworks such as TensorRT or ONNX Runtime would further reduce computational cost. Moreover, coupling it with RL frameworks could support a decision-support loop powered by fast and scalable fire propagation forecasts. Overall, the results presented in this study confirm the potential of deep recurrent neural architectures as efficient tools for computationally intensive wildfire simulators.

Appendix

This appendix provides a detailed description of the deterministic CA wildfire propagation model used in our simulations. The purpose of the model is to create a physically accurate representation of wildfire spread, suitable for real-time prediction and UAV-driven wildfire monitoring. Each component of the model is designed to capture fundamental physical drivers: fuel availability, topographic slope, wind direction, and local moisture effects. The simulation domain is a discrete 2-D grid that evolves in discrete steps $t = 0, 1, 2, \ldots, T$:

$$\mathcal{G} = \{1, \ldots, N_H\} \times \{1, \ldots, N_W\},$$

Each cell $(i, j) \in \mathcal{G}$ stores the following two fields:

Fire state:

The binary fire state $\mathcal{S}_t(i, j) \in \{0, 1\}$ indicates whether cell (i, j) is burning at time t (1 = burning, 0 = not burning) [28]. A necessary condition for ignition is the presence of fuel. Only cells with positive fuel can transition from the non-burning to the burning state while ignitions are forbidden in fuel-depleted cells. Once a cell ignites, it consumes fuel according to a burn-rate model (discussed below), and returns to the non-burning state when its fuel is exhausted.

Fuel / Vegetation:

$\mathcal{F}_t(i, j) \in \mathbb{R}_{\geq 0}$ (fuel amount at time t). The initial fuel field $\mathcal{F}_0(i, j)$ is generated by assigning vegetation to a fraction ρ_{veg} of the cells, and sampling positive fuel loads for those cells. Therefore, from a set of randomly selected vegetated cells, they receive sampled fuel values [13].

Elevation:

Elevation $\mathcal{E}(i, j) \in [0, 1]$ is computed by smoothing a random noise field to create realistic large-scale terrain variations [29]:

$$\widetilde{\mathcal{E}}(i, j) = (K_e * \xi)(i, j), \qquad \mathcal{E}(i, j) = \frac{\widetilde{\mathcal{E}}(i, j) - \min \widetilde{\mathcal{E}}}{\max \widetilde{\mathcal{E}} - \min \widetilde{\mathcal{E}}},$$

where K_e is a filter and ξ is a uniform noise.

Local slope estimation:

With the elevation field defined, we incorporate topographic effects on fire spread [30]. Since flames tilt upslope and preheat the fuel above them, steeper slopes accelerate fire propagation. To model this, we introduce a slope amplification factor $\mathcal{K}(i, j) \in \mathbb{R}_{\geq 0}$, computed in three steps.

$$\widetilde{\mathcal{K}}(i, j) = (K_s * \mathcal{E})(i, j),$$
$$\overline{\mathcal{K}}(i, j) = 1 + \alpha_k \big(\mathcal{E}(i, j) - \widetilde{\mathcal{K}}(i, j)\big),$$
$$\mathcal{K}(i, j) = \min\Big\{\mathcal{K}_{\max},\ \max\Big(\mathcal{K}_{\min},\ \overline{\mathcal{K}}(i, j)\Big)\Big\}.$$

Typical values are $\mathcal{K}_{\min} =$ (e.g., 0.5) to prevent negative or vanishing spread, and $\mathcal{K}_{\max} =$ (e.g., 1.5) to prevent unrealistically strong slope acceleration.

Fuel moisture:

Fuel moisture $\mathcal{M}(i, j)$ is generated as follows: first, a normalized smoothed moisture precursor $\overline{\mathcal{M}}(i, j) \in [0, 1]$ is generated in the same way elevation is generated. Then, by applying a shift toward a prescribed mean moisture level α_M and a variability amplitude β_M, the final moisture field is generated:

$$\widetilde{\mathcal{M}}(i,j) = (K_m * \eta)(i,j), \qquad \overline{\mathcal{M}}(i,j) = \frac{\widetilde{\mathcal{M}}(i,j) - \min \widetilde{\mathcal{M}}}{\max \widetilde{\mathcal{M}} - \min \widetilde{\mathcal{M}}},$$

$$\mathcal{M}(i,j) = \min\Big\{1,\ \max\Big[0,\ \alpha_M + \beta_M \left(\overline{\mathcal{M}}(i,j) - 0.5\right)\Big]\Big\},$$

Higher moisture suppresses ignitability and reduces burning intensity, while spatial variability (controlled by β_M) introduces realistic wet and dry patches across the environment.

Burn rate:

The intrinsic burn rate $\mathcal{R}(i,j)$ quantifies how fast fuel is consumed once a cell ignites [4, 5]. Let $\mathcal{F}_{\max} = \max_{i,j} \mathcal{F}_0(i,j)$. The normalized initial fuel is :

$$\widetilde{\mathcal{F}_0}(i,j) = \frac{\mathcal{F}_0(i,j)}{\mathcal{F}_{\max}}$$

The burn-rate activation, combining fuel availability, slope, and moisture is

$$\widetilde{\mathcal{R}}(i,j) = \widetilde{\mathcal{F}_0}(i,j)\ \left(1 - \alpha_B \mathcal{M}(i,j)\right)$$

$$\overline{\mathcal{R}}(i,j) = \min\Big\{1,\ \max\Big(0,\ \widetilde{\mathcal{R}}(i,j)\Big)\Big\}$$

Finally,

$$\mathcal{R}(i,j) = \mathcal{R}_{\min} + (\mathcal{R}_{\max} - \mathcal{R}_{\min})\, \overline{\mathcal{R}}(i,j)$$

Thus, dry, fuel-rich, upward-sloping terrain burns fastest. Typical values are $\mathcal{R}_{\min}$ = (e.g., 1) and $\mathcal{R}_{\max}$ = (e.g., 10) to prevent unrealistically fuel extinction.

Wind:

Wind is characterized by a global direction $\theta_w \in [0, 2\pi)$ and a wind-strength parameter $\omega \geq 0$. Wind modifies the ignition rule by increasing the likelihood of fire spread in the downwind direction. Therefore, to model directional influence, we use the standard 8-neighbor (Moore neighborhood) []:

$$\mathcal{N}(i,j) = \{(i + \delta_i^{(k)},\ j + \delta_j^{(k)}) \mid k = 1, \ldots, 8\}.$$

Here, $\delta_i^{(k)}$ and $\delta_j^{(k)}$ represent the row and column offsets of the k-th neighbor in the eight-direction neighborhood. Furthermore, each neighbor k is associated with an orientation angle θ_k corresponding to one of directions.

Ignition rule:

Wind enhances ignition downwind and suppresses upwind spread. The ignition influence at a time t is:

$$\mathcal{I}_t^{(w)}(i,j) = \omega \sum_k \mathcal{S}_t\Big(i + \delta_i^{(k)},\ j + \delta_j^{(k)}\Big)\ \max(0, \cos(\theta_k - \theta_w))\ .$$

Only burning neighbors contribute, and only those aligned with the wind contribute positively. Further, the ignition potential combining wind, slope, and moisture is

$$\mathcal{I}_t(i,j) = \mathcal{I}_t(i,j)\, \mathcal{K}(i,j)\, \big(1 - \gamma_M \mathcal{M}(i,j)\big).$$

A cell ignites if it contains fuel, is not already burning, and receives any positive ignition stimulus (three following conditions):

$$\mathcal{F}_t(i,j) > 0, \qquad \mathcal{S}_t(i,j) = 0, \qquad \mathcal{I}_t(i,j) > 0.$$

When ignition occurs:

$$\mathcal{S}_{t+1}(i,j) = 1, \qquad \mathcal{F}_{t+1}(i,j) = \mathcal{F}_t(i,j) - \mathcal{R}(i,j).$$

Code and Data Availability

All code and data associated with this work can be accessed through the following Git repository:
`https://github.com/drones-fireflies/fire-convlstm.git`

Acknowledgments

The authors acknowledge the support of the French National Research Agency (Agence Nationale de la Recherche, ANR) under reference ANR-23-IAS2-0002-01 for the FireFlies TSIA ANR Project.

References

[1] Keane, R. E., *Wildland Fuel Fundamentals and Applications*, Springer International Publishing, 2015. https://doi.org/10.1007/978-3-319-09015-3.

[2] Hawley, L. F., "Theoretical Considerations Regarding Factors which Influence Forest Fires," *Journal of Forestry*, Vol. 24, No. 7, 1926, pp. 756–763. https://doi.org/10.1093/jof/24.7.756.

[3] Gisborne, H. T., "The Objectives of Forest Fire-Weather Research," *Journal of Forestry*, Vol. 25, No. 4, 1927, pp. 452–456. https://doi.org/10.1093/jof/25.4.452.

[4] Sullivan, A. L., "Wildland surface fire spread modelling, 1990 - 2007. 1: Physical and quasi-physical models," *International Journal of Wildland Fire*, Vol. 18, No. 4, 2009, p. 349. https://doi.org/10.1071/WF06143.

[5] Sullivan, A. L., "Wildland surface fire spread modelling, 1990 - 2007. 2: Empirical and quasi-empirical models," *International Journal of Wildland Fire*, Vol. 18, No. 4, 2009, p. 369. https://doi.org/10.1071/WF06142.

[6] Linn, R., Reisner, J., Colman, J. J., and Winterkamp, J., "Studying wildfire behavior using FIRETEC," *International Journal of Wildland Fire*, Vol. 11, No. 4, 2002, p. 233. https://doi.org/10.1071/WF02007.

[7] McNamara, D., and Mell, W., "Computer modelling of wildland-urban interface fires," *Fire Materials*, 2011.

[8] Kokosza, A., Wrede, H., Gonzalez Esparza, D., Makowski, M., Liu, D., Michels, D. L., Pirk, S., and Palubicki, W., "Scintilla: Simulating Combustible Vegetation for Wildfires," *ACM Trans. Graph.*, Vol. 43, No. 4, 2024. https://doi.org/10.1145/3658192.

[9] Rothermel, R. C., "A mathematical model for predicting fire spread in wildland fuels," *Res. Pap. INT-115. Ogden, UT: U.S. Department of Agriculture, Intermountain Forest and Range Experiment Station. 40 p.*, Vol. 115, 1972.

[10] Andrews, P. L., "Current status and future needs of the BehavePlus Fire Modeling System," *International Journal of Wildland Fire. 23: 21-33.*, Vol. 23, 2014, pp. 21–33. https://doi.org/10.1071/WF12167.

[11] Finney, M. A., "FARSITE: Fire Area Simulator-model development and evaluation," *Res. Pap. RMRS-RP-4, Revised 2004. Ogden, UT: U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station. 47 p.*, Vol. 4, 1998. https://doi.org/10.2737/RMRS-RP-4.

[12] Filippi, J.-B., Bosseur, F., and Grandi, D., "ForeFire: open-source code for wildland fire spread models," *Advances in forest fire research*, Imprensa da Universidade de Coimbra, 2014, pp. 275–282. https://doi.org/10.14195/978-989-26-0884-6_29.

[13] Boychuk, D., Braun, W. J., Kulperger, R. J., Krougly, Z. L., and Stanford, D. A., "A stochastic forest fire growth model," *Environmental and Ecological Statistics*, Vol. 16, No. 2, 2009, pp. 133–151. https://doi.org/10.1007/s10651-007-0079-z.

[14] Khoshdel, S., Luo, Q., and Afghah, F., "PyroTrack: Belief-Based Deep Reinforcement Learning Path Planning for Aerial Wildfire Monitoring in Partially Observable Environments," *2024 American Control Conference (ACC)*, 2024, pp. 601–607. https://doi.org/10.23919/ACC60939.2024.10644894.

[15] Jiang, W., Qiao, Y., Su, G., Li, X., Meng, Q., Wu, H., Quan, W., Wang, J., and Wang, F., "WFNet: A hierarchical convolutional neural network for wildfire spread prediction," *Environmental Modelling Software*, Vol. 170, 2023, p. 105841. https://doi.org/https://doi.org/10.1016/j.envsoft.2023.105841.

[16] Burge, J., Bonanni, M., Ihme, M., and Hu, L., "Convolutional LSTM Neural Networks for Modeling Wildland Fire Dynamics," , 2020. https://doi.org/10.48550/ARXIV.2012.06679, URL https://arxiv.org/abs/2012.06679.

[17] Masrur, A., Yu, M., and Taylor, A., "Capturing and interpreting wildfire spread dynamics: attention-based spatiotemporal models using ConvLSTM networks," *Ecological Informatics*, Vol. 82, 2024, p. 102760. https://doi.org/10.1016/j.ecoinf.2024.102760.

[18] Julian, K. D., and Kochenderfer, M. J., "Distributed Wildfire Surveillance with Autonomous Aircraft Using Deep Reinforcement Learning," *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 8, 2019, pp. 1768–1778. https://doi.org/10.2514/1.G004106.

[19] Hochreiter, S., and Schmidhuber, J., "Long Short-Term Memory," *Neural Computation*, Vol. 9, No. 8, 1997, pp. 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

[20] Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-k., and Woo, W.-c., "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting," , 2015. https://doi.org/10.48550/ARXIV.1506.04214, URL https://arxiv.org/abs/1506.04214.

[21] Anastasiou, N., Kondylatos, S., and Papoutsis, I., "Wildfire spread forecasting with Deep Learning," , 2025. https://doi.org/10.48550/arXiv.2505.17556, URL http://arxiv.org/abs/2505.17556.

[22] Andrianarivony, H. S., and Akhloufi, M. A., "Machine Learning and Deep Learning for Wildfire Spread Prediction: A Review," *Fire*, Vol. 7, No. 12, 2024, p. 482. https://doi.org/10.3390/fire7120482.

[23] Radke, D., Hessler, A., and Ellsworth, D., "FireCast: Leveraging Deep Learning to Predict Wildfire Spread," *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019, pp. 4575–4581. https://doi.org/10.24963/ijcai.2019/636.

[24] Jaccard, P., "The Distribution of the Flora in the Alpine Zone," *New Phytologist*, Vol. 11, No. 2, 1912, pp. 37–50. https://doi.org/10.1111/j.1469-8137.1912.tb05611.x.

[25] Huttenlocher, D., Klanderman, G., and Rucklidge, W., "Comparing images using the Hausdorff distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, No. 9, 1993, pp. 850–863. https://doi.org/10.1109/34.232073.

[26] Li, X., Zhang, M., Zhang, S., Liu, J., Sun, S., Hu, T., and Sun, L., "Simulating Forest Fire Spread with Cellular Automation Driven by a LSTM Based Speed Model," *Fire*, Vol. 5, No. 1, 2022, p. 13. https://doi.org/10.3390/fire5010013.

[27] Raoufi, M., Telikani, A., Zhang, T., and Shen, J., "Fire front path planning and tracking control of Uncrewed Aerial Vehicles using deep reinforcement learning," *Robotics and Autonomous Systems*, Vol. 193, 2025, p. 105076. https://doi.org/10.1016/j.robot.2025.105076.

[28] Karafyllidis, I., and Thanailakis, A., "A model for predicting forest fire spreading using cellular automata," *Ecological Modelling*, Vol. 99, No. 1, 1997, pp. 87–97. https://doi.org/10.1016/S0304-3800(96)01942-4.

[29] Perlin, K., "An image synthesizer," *Proceedings of the 12th annual conference on Computer graphics and interactive techniques - SIGGRAPH '85*, 1985, pp. 287–296. https://doi.org/10.1145/325334.325247.

[30] Freire, J. G., and DaCamara, C. C., "Using cellular automata to simulate wildfire propagation and to assist in fire management," *Natural Hazards and Earth System Sciences*, Vol. 19, No. 1, 2019, pp. 169–179. https://doi.org/10.5194/nhess-19-169-2019.