A Spatio-Temporal Neural Network Forecasting Approach for Emulation of Firefront Models

Andrew Bolt [†], Carolyn Huston [†], Petra Kuhnert [†◇], Joel Janek Dabrowski [†], James Hilton [†], Conrad Sanderson [†‡]

[†] *Data61 / CSIRO, Australia;* [◇] *Australian National University, Australia;* [‡] *Griffith University, Australia*

***Abstract*—Computational simulations of wildfire spread typically employ empirical rate-of-spread calculations under various conditions (such as terrain, fuel type, weather). Small perturbations in conditions can often lead to significant changes in fire spread (such as speed and direction), necessitating a computationally expensive large set of simulations to quantify uncertainty. Model emulation seeks alternative representations of physical models using machine learning, aiming to provide more efficient and/or simplified surrogate models. We propose a dedicated spatio-temporal neural network based framework for model emulation, able to capture the complex behaviour of fire spread models. The proposed approach can approximate forecasts at fine spatial and temporal resolutions that are often challenging for neural network based approaches. Furthermore, the proposed approach is robust even with small training sets, due to novel data augmentation methods. Empirical experiments show good agreement between simulated and emulated firefronts, with an average Jaccard score of 0.76.**

***Index Terms*—forecasting, wildfire, emulation, approximation, surrogate model, spatio-temporal, machine learning.**

I. Introduction

Wildfires pose a serious threat to communities as well as natural flora and fauna in many regions throughout the world [5], [6], [22]. Forecasting spread of wildfires is critical in fire management, planning, and response efforts. Simulated (*in silico*) fires provide valuable data for operational managers to assess potential impacts on populated or sensitive areas, in order to guide active management, mitigation and evacuation efforts.

Several fire behaviour characteristic models have been developed [8], [18], [26]. Such models are generally computationally expensive as they may be based on complex methodologies such as the level-set method [18]. This can hinder their applicability for decision support, especially when large scale simulations or numerous ensemble predictions are required to account for uncertainty.

Model emulation (also known as surrogate modelling) employs a computationally efficient predictive model that approximates a complex physical process model, such as computer or numerical simulators [3], [21]. Emulation may be able to overcome some of the limitations of large scale complex simulations. Early emulation approaches used machine learning techniques such as Gaussian processes [15], followed by random forests [9], [17] and deep neural networks [13], [14], [23]. Neural networks are highly adaptable and have been successfully implemented in several physical system emulation problems [14], [23], [25].

Recent reviews on applications of machine learning to wildfires cover fire susceptibility prediction, fire spread prediction, fuel categorisation, fire occurrence detection, and decision support systems [2], [5], [12]. Deep learning architectures such as convolutional neural networks (CNNs) [4], [10], [19], and recurrent neural networks [7] have also been applied.

Within the literature on neural networks related to model emulation for fire spread and growth prediction, Allaire et al. [4] present a CNN emulator for hazard assessment in a contained region of interest. The emulator predicts the amount of burned land (scalar value). The model does not estimate fire dynamics. Burge et al. [7] and Hodges et al. [10] propose CNN emulators for predicting fire dynamics. Both approaches use a small output array size ($<$ 100 pixels) which limits the spatial resolution or extent that can be evaluated. Radke et al. [19] propose a CNN based emulator, which estimates the likelihood of a pixel outside the firefront burning within a 24 hour window. Rather than evaluating each pixel, the likelihoods of a set of sample pixels are generated. The wide temporal resolution limits the ability of the model to estimate fine timescale dynamics of the fire. Sung et al. [24] propose a neural network model incorporating a U-Net structure [20], trained with a dataset of daily fire perimeters. The model estimates the likelihood of the fire reaching a given pixel. Relatively low accuracy is obtained, possibly due to low temporal resolution.

In this paper we propose a neural network based emulator for estimating high resolution fire spread over large spatial and temporal domains. The model is trained with simulated data produced using empirical rate-of-spread estimates. The proposed emulator design is able to incorporate data of varying spatial and temporal resolutions. Furthermore it is capable of generating estimates over large spatial extents with varying shapes, which is often challenging for emulation approaches. Lastly, the model is robust on small training sets due to employing novel data augmentation methods.

The architecture of the proposed emulator is overviewed in Section II, and its design features are covered in Section III. As there are hyper-parameters that can be adjusted, we provide an empirical evaluation at various configurations in Section IV. The best performing configuration has an average Jaccard score of 0.76, indicating good agreement between simulated and emulated firefronts. The proposed model provides a template upon which further developments can be introduced, especially methods for uncertainty quantification.

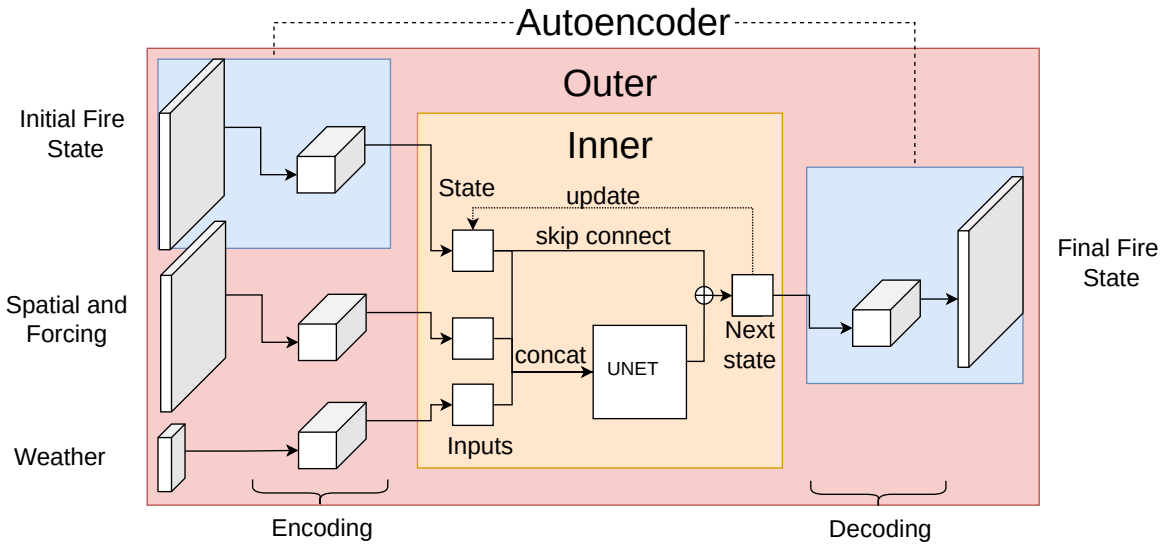

Fig. 1. The proposed emulator architecture, comprising three main components. The autoencoder (blue) encodes and decodes the fire input state. The outer component (red) incorporates the autoencoder, as well as encoding spatial, forcing, and weather features. The inner component (orange) handles the dynamics of the emulator. The latent fire state is updated using information from the spatial and forcing layers, as well as weather feature inputs. These layers are concatenated and passed through a shallow U-Net structure. The sum of the U-Net output and the input latent fire state produce a new latent fire state estimate.

II. Emulator Architecture

Fig. 1 shows a simplified schematic of the proposed emulator architecture. There are three main components: autoencoder component, outer component, and inner component. The autoencoder component is trained separately and its weights are transferred to parts of the outer component. The outer component is a feature engineering and downsampling network which passes inputs to the inner component. The outer component also upsamples the outputs from the inner component. Finally, the inner component handles the dynamics of the system. It forecasts latent fire states based on information from spatial and temporal features.

There are three types of inputs to the model: fire state image, spatial data and forcing terms, and finally weather time series. Each fire state is an image where pixel values represent when the fire reached a given location. The spatial data are images representing heightmaps and land classes (eg. forest, grassland). The forcing terms are curing and drought factors. The weather time series include temperature, wind speed, and relative humidity.

A. Outer Network

The outer components of the model are illustrated by the red shading in Fig. 1. This component incorporates the fire state autoencoder. Convolutional layers are used on the spatial data (heightmap and land class map) for feature extraction; downscaling is performed by strided convolutional layers. This encodes the information into a smaller latent representation. Similarly, forcing inputs (drought and curing factors) are expanded to have the same spatial extent as the latent spatial features and are concatenated together.

The downsampling and upsampling of the fire state is performed by the autoencoder. The latent fire state and latent spatial features form the state component of the inner network.

Weather features are in the form of time-series data. Interpolation via upsampling is performed. Each set of weather values at a given time are passed through two dense layers for feature extraction before being expanded to the same spatial extent as the latent spatial features. These are then passed to the inner network as inputs.

B. Autoencoder

The purpose of the autoencoder is to encode and decode the fire state to and from a lower latent dimension. The autoencoder is represented by joining the blue components in Fig. 1. The encoding component consists of only linear transformations to preserve the relative temporal relationships between pixel values. An average pooling layer is used before a space to depth transformation. The decoding component consists of a depth to space transformation followed by a strided convolutional layers. The autoencoder is trained on fire state data separately from the full emulator.

C. Inner Network

The inner component of the model incorporates the dynamics of the system, taking an initial latent fire state and producing a new updated estimate. The inner component is illustrated by the orange shading in Fig. 1. This component is a one-step ahead forecast module.

The state of the module consists of the latent fire arrival state. This is updated using dynamic inputs from the latent weather features and static inputs from the latent spatial and forcing features. Each successive weather input advances the fire state forward in time.

The inputs and fire state are concatenated and passed into a shallow U-Net [20], [27]. U-Nets are formed by join-

ing a contracting path with an expanding path, formed by down/upsampling convolutional layers. This structure is able to capture dynamics at various spatial resolutions, with deeper levels capturing broader interactions. We incorporate a skip connection between the input latent fire state and the output of the U-Net. In effect the U-Net only needs to learn the change or *residual* between successive latent fire states.

III. Design Features

In this section we will briefly discuss some of the unique challenges that inform our choice of emulator design. Broadly, we want an emulator that is agnostic to temporal and spatial resolutions and is able to generate high resolution outputs.

In order to achieve a model that can incorporate various spatial and temporal resolutions we express features like distance, height, and wind speed in unit-less terms. For example, wind speed is converted from meters per second into pixel lengths per interval. In this way a different dataset operating with a different spatial (or temporal) resolution can be re-sampled to be compatible with the model inputs.

A further challenge is that the training data often consists of large sized image arrays ($>$ 1000 px). Furthermore, the image arrays can have varying sizes. To address this we choose a fully convolutional network, and encode the spatial data into a smaller latent representation through strided convolutional operations. This reduces the complexity of the representation and allows the model to take arbitrarily sized inputs. This is performed by the outer component of the model (Section II-A). In addition, the inner component (Section II-C) employs a shallow U-Net structure. This allows for long range interactions between pixels to be considered by the model using only a modest overhead of complexity.

The inner component of the emulator deals with the dynamic changes to the fire arrival state. To ensure that the outer model does not produce any dynamic changes, we use an autoencoder. The autoencoder is trained on the fire arrival state. Once trained, the autoencoders' weights are frozen. The autoencoder is then used by the outer network to encode and decode the fire arrival state.

Wildfire spread is an inherently stochastic process. In some cases an estimate may incorrectly place the fire slightly ahead of an obstacle such as a body of water. If we consider this as the source of a new firefront propagating through open terrain, then the burned area by the new front will increase quadratically in time [28]. By training the model over a single time interval we limit the training penalty of these mismatches.

As an additional step to the training process, each sample is cropped around a point on the fire's perimeter. This reduces the spatial extent of each sample and generates a uniform size. This allows for batch processing as well as greatly reducing the memory requirements for training.

Using cropping and single intervals for model training represents a novel method of data augmentation. A single fire can produce numerous semi-independent training samples by using various cropping locations and time intervals. Further data augmentation was performed using rotation, reflection and transposition transformations.

A drawback the cropping approach is that cropping removes some information about the fire's position as a whole. The likelihood of fire 'entering' a cropped region cannot be inferred during training. To account for this we pad each sample. In this padded region, each pixel value is assigned to the maximum of the predicted or target values. This removes the loss penalty when the presence of fire is not correctly inferred around the border of a region.

IV. Empirical Evaluation

In this section we first overview the evaluation dataset, and then present an empirical evaluation using several configurations of hyper-parameters in the proposed emulator.

A. Dataset

We use a dataset of 195 simulated fires generated via the SPARK fire simulation platform [18]. 155 fires are used as the training set; 40 fires are reserved for validation. During training and validation the regions are cropped and only a single interval is used. Furthermore, we use a prediction set, composed of the 40 validation fires, but no cropping is performed and the entire duration of the fire is considered. This set is not used in training, but is used to evaluate the performance of the emulator across the full scope of a fire scenario.

The location of the fires is representative of regional South Australia. Land classification maps[1], topology data[2], and weather conditions[3] are sampled to reflect realistic regional conditions. There is a large bias in land classification towards grassland (78.8%), mallee-heath shrubland (10.5%), water/unburnable (6.27%). The weather is representative of high fire risk conditions, often consisting of high temperatures, low humidity, and moderate to high winds.

Spatial data is converted into the same coordinate reference system and has a resolution of 30 meters. The height map is converted into x and y gradient maps. Weather data is polled from weather stations every 30 minutes (one interval). The weather values are interpolated (upscaled) into 4 slices for the majority of our trials. Values are scaled by maximum and minimum values in the training set. Wind speed and direction are converted into x and y components. Finally there are two forcing terms that the model incorporates: *drought factor* and *curing factor*. These terms depend on long term weather trends that are considered fixed for the duration of the fire.

[1]Land classification datasets derived from Department of Agriculture and Water Resources (ABARES) Land Use of Australia 2010-11 dataset. Data is publicly available under Creative Commons Attribution 3.0 Australia Licence.

[2]Topography data sets derived from Geoscience Australia SRTM-derived 1 Second Digital Elevation Models Version 1.0. Data is publicly available under Creative Commons Attribution 4.0 International Licence.

[3]Meteorological time series data sets derived from Bureau of Meteorology automated weather station data.

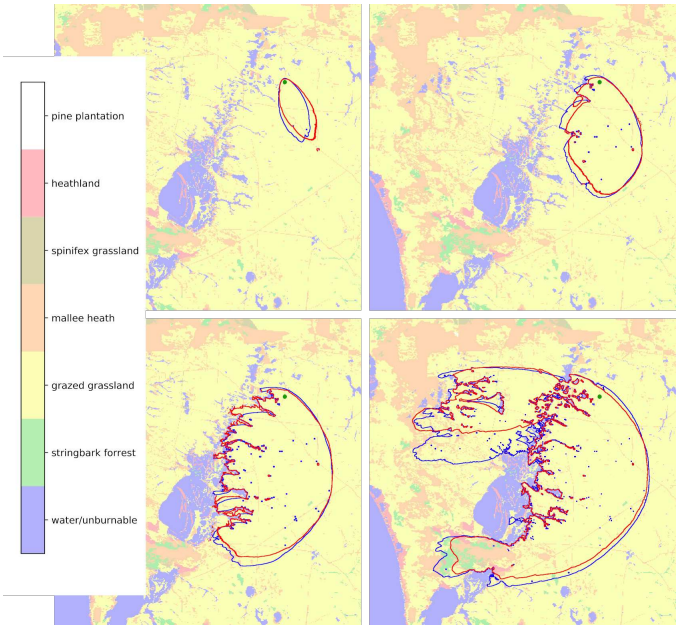

Fig. 2. Evolution of firefront contours for a trial, shown over four panels (left-to-right, top-to-bottom). Emulator (red), simulation (blue) and ignition point (green) are overlaid over land classes. Dominant land classes are grassland (yellow), mallee-heath shrubland (orange), and water (blue). The wind initially drives the fire south, before turning west. Map size is 46.1 km × 46.1 km, 30 meter resolution.

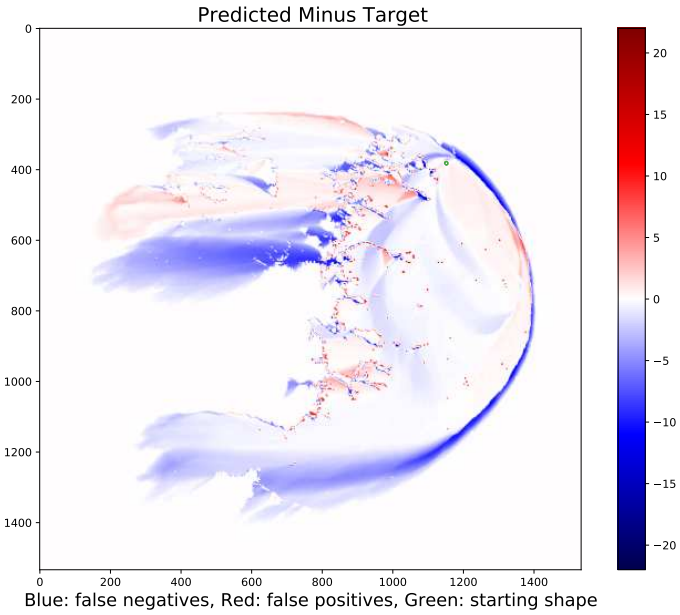

Fig. 3. The difference between predicted and target fire arrival times (measured in 30 minute intervals) for the same test sample as illustrated in Fig. 2. Positive values (red) indicate false-positives while negative values (blue) represent false-negatives. The Jaccard score for this trial is 0.81.

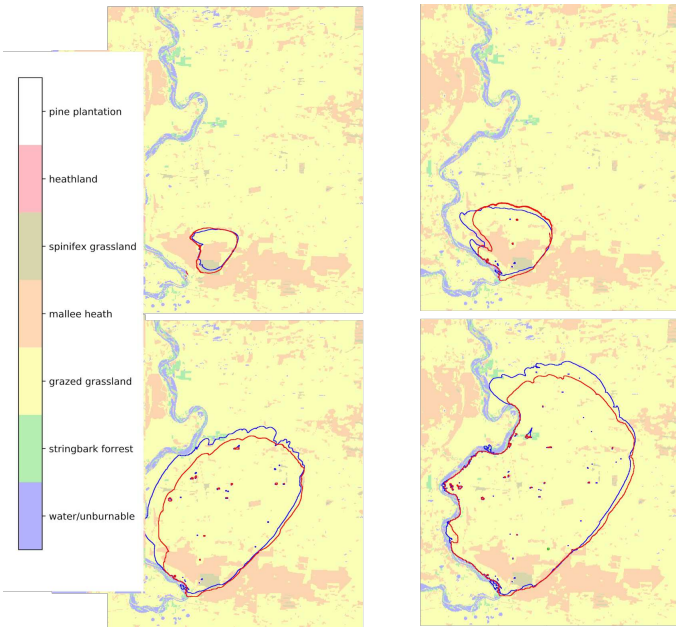

Fig. 4. Evolution of firefront contours for a trial, shown over four panels (left-to-right, top-to-bottom). Emulator (red), simulation (blue) and ignition point (green) are overlaid over land classes. Dominant land classes are grassland (yellow), mallee-heath shrubland (orange), and water (blue). The wind initially drives the fire south-east, before turning north. Map size is 46.1 km × 38.4 km, 30 meter resolution.

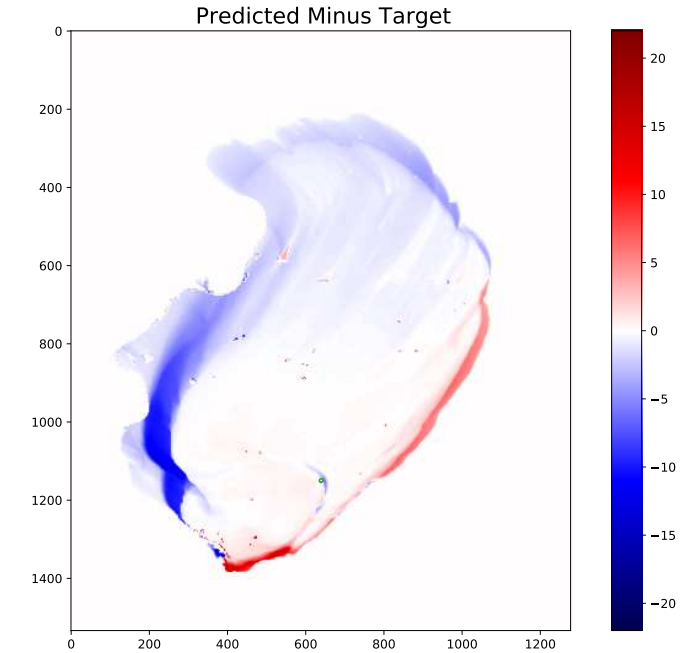

Fig. 5. The difference between predicted and target fire arrival times (measured in 30 minute intervals) for the same test sample as illustrated in Fig. 4. Positive values (red) indicate false-positives while negative values (blue) represent false-negatives. The Jaccard score for this trial is 0.90.

B. Metrics and Loss

For evaluation metrics we choose the Jaccard score, also known as intersection over union (IOU) score. While this straightforward metric may not capture the whole 'goodness of fit', it nevertheless provides a basic grounding [11].

The autoencoder component is trained using mean absolute error (MAE) loss. For training the emulator we introduce a custom loss function, which evaluates how well the predictions perform against a benchmark trivial prediction (where the output is the same as the input). Let y_i, y_t, and y_p be the initial fire state, target fire state, and predicted fire state respectively. Furthermore, let $\mathrm{MAE}(a, b)$ be the mean absolute error across corresponding pixels in images a and b. The loss $\mathcal{L}$ of fire state $\mathcal{P}$ is defined as:

$$\mathcal{L}(\mathcal{P}) = \log_{10}\left(\frac{\mathrm{MAE}(y_p, y_t) + \tau}{\mathrm{MAE}(y_i, y_t) + \tau}\right). \quad (1)$$

Small fire growth results in only a small set of pixels indicating burns, which in turn leads to very small $\mathrm{MAE}(y_p, y_t)$. To address this, $\mathrm{MAE}(y_i, y_t)$ is used as a normalisation factor. Furthermore, $\tau = 10^{-12}$ is used to remove singularities that arise if either MAE values approach zero.

C. Evaluation

We implemented the emulator in TensorFlow [1], using the *Adam* optimiser [16] with a batch size of 16. The autoencoder was trained for 20 epochs and returned an MAE loss of 2.1×10^{-3}. The emulator was trained for 50 epochs using the loss function in Eqn. (1).

We train the emulator under several configurations of hyper-parameters. Specifically, we test variously sized cropping windows (c), U-Net depths (d), and padding values (p). The number of interpolation slices per interval is 4.

Table I shows the average loss and Jaccard (IOU) scores for the training and validation sets, as well as for the prediction set. The prediction results are split into two parts. The first part is when the emulator is run over the entire duration of the prediction set (intervals 0-22). The second part is when the emulator begins with the fire in progress (intervals 5-22). Furthermore, two Jaccard scores are shown: the top value represents an unweighted score averaged over all samples, while the lower bracketed value shows the score weighted by burned area over all samples.

Direct comparison between configurations of loss values is not possible due to differences in how padding and cropping sizes affect the loss function defined in Eqn. (1). We note that the differences in loss and Jaccard scores between training and evaluation sets are small. This indicates that the model is generalising well to the entire dataset.

The prediction set shows how well the model performs across the full spatial and temporal extents of each sample. The best performance is found for larger cropping sizes (512 pixels). The prediction Jaccard scores of the smaller cropped samples are significantly lower than that found during training and evaluation. There does not appear to be a significant difference between depth 1 and 2 U-Net configurations. Finally, the model performs better when using a 32 pixel padding distance. A number of trials were also performed using 6 interpolation slices per interval rather than 4; only a very marginal improvement in Jaccard scores was observed.

TABLE I
MODEL LOSS AND EVALUATION METRICS. EACH CONFIGURATION COMPRISES c, d, p COMPONENTS, WHERE c IS SIZE OF THE CROPPING WINDOW, d IS THE DEPTH OF THE U-NET COMPONENT, AND p IS THE AMOUNT OF PADDED PIXELS.

Configuration	Train. loss	Val. loss	Train. Jacc.	Val. Jacc.	Pred. Jacc. (0-22)	Pred. Jacc. (5-22)
512c, 1d, 32p	-1.43	-1.43	0.78	0.79	**0.76 (0.80)**	**0.79 (0.83)**
512c, 1d, 64p	-1.47	-1.48	0.79	0.79	0.71 (0.74)	0.77 (0.74)
512c, 2d, 32p	-1.43	-1.45	0.78	0.79	0.71 (0.74)	0.77 (0.81)
512c, 2d, 64p	-1.55	-1.51	0.81	0.80	0.69 (0.70)	0.71 (0.76)
256c, 1d, 32p	-1.61	-1.73	0.81	0.83	0.64 (0.67)	0.69 (0.72)
256c, 1d, 64p	-1.88	-1.92	0.86	0.86	0.71 (0.72)	0.72 (0.77)
256c, 2d, 32p	-1.64	-1.67	0.82	0.82	0.67 (0.74)	0.71 (0.77)
256c, 2d, 64p	-1.98	-1.93	0.87	0.86	0.65 (0.68)	0.68 (0.74)
128c, 1d, 32p	-1.91	-1.92	0.86	0.86	0.42 (0.43)	0.45 (0.45)
128c, 2d, 32p	-1.77	-1.82	0.84	0.85	0.52 (0.52)	0.53 (0.55)

Figs. 2 and 3 illustrate the emulator performance on a sample fire. There is close agreement between simulated and emulated fires for much of the duration. The largest disagreement occurs as the fire passes through a region containing small bodies of water. The emulator fails to find a similar pathway to the original Spark simulation.

Figs. 4 and 5 show the emulator performance on another sample fire. In this example there is also good agreement between simulated and emulated fires. To the west an early under-estimation by the emulator leads to a large under-estimation as the wind changes and pushes the fire north.

These types of behaviour are typical of many samples that have been manually inspected. Small differences between emulator and simulation are often exaggerated over time. Nonetheless, the overall dynamics of the emulator appear to be in line with expected fire behaviour.

V. CONCLUSION

In this paper we have shown how convolutional networks can be constructed in order to closely emulate wildfire spread from the Spark simulator, resulting in an average Jaccard (IOU) score of 0.76 for up to 11.5 hour fire duration. Qualitatively, the emulator makes predictions that exhibit very similar behaviour to that of the targeted simulations. The stochastic nature of wildfires means that large discrepancies are often the result of small differences being exaggerated, rather than a fundamental problem in the emulation estimate.

The proposed approach has several features that make it versatile. It is able to work using variable spatial extents and resolutions, variable temporal extents and resolutions, as well as being able to incorporate various types of spatial, temporal and scalar features.

We use a novel approach to model training. This approach incorporates transfer learning as well as data augmentation in the form of targeted cropping and the use of single intervals for training (rather than full duration trials). Additionally, we use a custom loss function that is designed to operate well for this specific class of problem.

A promising future area of exploration is using ensemble simulations as training data for emulators. These ensembles would be constructed to generate likelihood estimates of fire locations. Using this data, an emulator could be trained to estimate ensemble predictions using only a single run.

A further area of interest is to take data from real world samples and use these to 'fine tune' the model parameters. In this way it may be possible to use relatively sparse real world data to improve model performance, and potentially infer fire dynamics directly.

The flexibility of the modelling approach also means that it should be possible to incorporate new features into the model without needing to fully retrain the model. For example, if a new land class is added, then it is possible that only the first few downsampling convolutional layers would need to be retrained.

As a final remark we note that the proposed approach is not inherently specific to wildfire prediction. Similar geo-spatial modelling problems such as pollutant spread, pest spread, or disease spread may also be well represented by a similar emulation approach.

References

[1] M. Abadi, P. Barham, J. Chen, et al. TensorFlow: A system for large-scale machine learning. In *USENIX Symp. Operating Systems Design and Implementation*, pages 265–283, 2016.

[2] F. Abid. A survey of machine learning algorithms based forest fires prediction and detection systems. *Fire Technology*, 57(2):559–590, 2021.

[3] R. Alizadeh, J. K. Allen, and F. Mistree. Managing computational complexity using surrogate models: a critical review. *Research in Engineering Design*, 31(3):275–298, 2020.

[4] F. Allaire, V. Mallet, and J.-B. Filippi. Emulation of wildland fire spread simulation using deep learning. *Neural Networks*, 141:184–198, 2021.

[5] K. Bot and J. G. Borges. A systematic review of applications of machine learning techniques for wildfire management decision support. *Inventions*, 7(1):15, 2022.

[6] D. Bowman, C. Kolden, J. Abatzoglou, F. Johnston, et al. Vegetation fires in the anthropocene. *Nature Reviews Earth & Environment*, 1(10):500–515, 2020.

[7] J. Burge, M. Bonanni, M. Ihme, and L. Hu. Convolutional LSTM neural networks for modeling wildland fire dynamics. *arXiv:2012.06679*, 2020.

[8] M. A. Finney. FARSITE: Fire area simulator–model development and evaluation. Research Paper RMRS-RP-4, United States Department of Agriculture, 1998.

[9] D. Gladish, D. Pagendam, L. Peeters, P. Kuhnert, and J. Vaze. Emulation engines: choice and quantification of uncertainty for complex hydrological models. *Journal of Agricultural, Biological, and Environmental Statistics*, 23(1):39–62, 2018.

[10] J. L. Hodges and B. Y. Lattimer. Wildland fire spread modeling using convolutional neural networks. *Fire Technology*, 55(6):2115–2142, 2019.

[11] C. Huston, C. Miller, J. Hilton, and E. Campbell. Thoughts on spatio-temporal uncertainty metrics motivated by input sensitivity in the spark bushfire spread model. In *International Congress on Modelling and Simulation*, pages 1296–1302, 2015.

[12] P. Jain, S. C. Coogan, S. G. Subramanian, M. Crowley, S. Taylor, and M. D. Flannigan. A review of machine learning applications in wildfire science and management. *Environmental Reviews*, 28(4):478–505, 2020.

[13] K. Kashinath, M. Mustafa, A. Albert, J.-L. Wu, C. Jiang, et al. Physics-informed machine learning: case studies for weather and climate modelling. *Philosophical Transactions of the Royal Society A*, 379(2194):20200093, 2021.

[14] M. F. Kasim, D. Watson-Parris, L. Deaconu, S. Oliver, et al. Building high accuracy emulators for scientific simulations with deep neural architecture search. *Machine Learning: Science and Technology*, 3(1):015013, 2021.

[15] M. Kennedy and A. O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society, Series B: Statistical Methodology*, 63(3):425–464, 2001.

[16] D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. In *Int. Conf. Learning Representations*, 2015.

[17] W. B. Leeds, C. K. Wikle, et al. Modeling 3-D spatio-temporal biogeochemical processes with a forest of 1-D statistical emulators. *Environmetrics*, 24(1):1–12, 2013.

[18] C. Miller, J. Hilton, A. Sullivan, and M. Prakash. SPARK – a bushfire spread prediction tool. In *Environmental Software Systems, Infrastructures, Services and Applications*, pages 262–271, 2015.

[19] D. Radke, A. Hessler, and D. Ellsworth. FireCast: Leveraging deep learning to predict wildfire spread. In *International Joint Conference on Artificial Intelligence*, pages 4575–4581, 2019.

[20] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. Lecture Notes in Computer Science, vol. 9351, pp. 234–241, 2015.

[21] C. Sanderson, D. Pagendam, B. Power, F. Bennett, and R. Darnell. Opportunistic emulation of computationally expensive simulations via deep learning. In *International Congress on Modelling and Simulation*, pages 673–679, 2021.

[22] A. Scott, D. Bowman, W. Bond, S. Pyne, and M. Alexander. *Fire on Earth: An Introduction*. Wiley, 2013.

[23] M. Sit, B. Z. Demiray, et al. A comprehensive review of deep learning applications in hydrology and water resources. *Water Science & Technology*, 82(12):2635–2670, 2020.

[24] S. Sung, Y. Li, and L. Ortolano. WildfireNet: Predicting wildfire profiles (student abstract). *AAAI Conference on Artificial Intelligence*, 35(18):15905–15906, 2021.

[25] J. J. Thiagarajan, B. Venkatesh, R. Anirudh, et al. Designing accurate emulators for scientific processes using calibration-driven deep models. *Nature Communications*, 11(1):5622, 2020.

[26] K. Tolhurst, B. Shields, and D. Chong. Phoenix: development and application of a bushfire risk management tool. *Australian Journal of Emergency Management*, 23(4):47–54, 2008.

[27] C. Wang, Z. Zhao, Q. Ren, Y. Xu, and Y. Yu. Dense U-net based on patch-based learning for retinal vessel segmentation. *Entropy*, 21:168, 2019.

[28] N. Zekri, O. Harrouz, A. Kaiss, J.-P. Clerc, and X. D. Viegas. Generalized Byram's formula for arbitrary fire front geometries. *International Journal of Thermal Sciences*, 110:222–228, 2016.