Location-Aware Adaptive Normalization: A Deep Learning Approach for Wildfire Danger Forecasting

Mohamad Hakam Shams Eddin, *Member, IEEE*, Ribana Roscher, *Member, IEEE*, and Juergen Gall, *Member, IEEE*

***Abstract*— Climate change is expected to intensify and increase extreme events in the weather cycle. Since this has a significant impact on various sectors of our life, recent works are concerned with identifying and predicting such extreme events from Earth observations. With respect to wildfire danger forecasting, previous deep learning approaches duplicate static variables along the time dimension and neglect the intrinsic differences between static and dynamic variables. Furthermore, most existing multibranch architectures lose the interconnections between the branches during the feature learning stage. To address these issues, this article proposes a 2-D/3-D two-branch convolutional neural network (CNN) with a location-aware adaptive normalization (LOAN) layer. Using LOAN as a building block, we can modulate the dynamic features conditional on their geographical locations. Thus, our approach considers feature properties as a unified yet compound 2-D/3-D model. Besides, we propose using the sinusoidal-based encoding of the day of the year to provide the model with explicit temporal information about the target day within the year. Our experimental results show a better performance of our approach than other baselines on the challenging FireCube dataset. The results show that location-aware adaptive feature normalization is a promising technique to learn the relation between dynamic variables and their geographic locations, which is highly relevant for areas where remote sensing data build the basis for analysis. The source code is available at https://github.com/HakamShams/LOAN.**

***Index Terms*— Adaptive normalization, climate science, convolutional neural network (CNN), machine learning, remote sensing, time encoding, wildfire.**

I. Introduction

THERE is a general expectation that weather and climate extremes will change their patterns and frequencies in the future [1], [2], [3], [4]. This is particularly the case for the Mediterranean region, which has been identified as a hot spot for climatic changes [5], [6], [7]. Because extreme weather events can impose short- and long-term risks in our Earth system, predicting these risks, such as droughts, windstorms, and wildfires, has become recently more relevant. In particular, wildfire forecasting constitutes one of the open challenges for risk assessment and emergency response [8], [9], [10]. Wildfire forecasting refers to the task of fire-susceptibility mapping using key remote sensing, meteorological, and anthropogenic variables [11]. Building an integrated modeling system of the Earth should also consider wildfire events to comprehend the origin of past patterns better and predict future ones [12]. Unlike typical prediction tasks, understanding when weather conditions have a high tendency to cause fire events addresses more complexities; among these are the stochastic nature of fire events [13] and fire drivers, which are time-dependent and intercorrelated across variables [14]. Moreover, the prediction model should consider difficulties like a high false positive (FP) error rate, uncertainty, and class imbalance.

In recent years, many works leveraged classical machine learning approaches to solve the task [11]. More recently, deep learning methods [15] have become popular since they can handle large multivariate datasets more efficiently and are able to learn highly complex relationships between observations and the predicted outcome. In the context of wildfire danger forecasting, Kondylatos et al. [13] and Prapas et al. [16] proposed to use recurrent neural networks in combination with 2-D convolutions to exploit both temporal and spatial contexts. These approaches, however, do not distinguish between the different input variables. Static variables, such as elevation, which barely change over time, are simply copied and concatenated with dynamic variables, such as surface temperature. This results not only in a highly redundant input to the network, but it also neglects strong causal effects between static and dynamic variables. For instance, the surface temperature strongly depends on the geographical location, which is described by static variables.

In this work, we, thus, propose a convolutional neural network (CNN) for wildfire danger forecasting that handles static and dynamic variables differently. Since the static variables do not change over time, they are processed by a branch consisting of 2-D convolutions, while the dynamic variables are processed by the second branch with 3-D convolutions, as illustrated in Fig. 1. To address the causal effect of static variables on dynamic variables, we introduce *feature modulation* for the dynamic variables where the modulation parameters are generated dynamically and conditionally on the geographical location. We, thus, name this method location-aware adaptive normalization (LOAN). In addition, we encode the date of the forecasting during a year by an absolute time encoding based on the sinusoidal encoding [17]. Both LOAN and time encoding can be implemented as plugin

Manuscript received 15 December 2022; revised 7 April 2023 and 26 May 2023; accepted 5 June 2023. Date of publication 12 June 2023; date of current version 27 June 2023. This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Grant SFB 1502/1–2022 and Project 450058266. *(Corresponding author: Mohamad Hakam Shams Eddin.)*

Mohamad Hakam Shams Eddin is with the Department of Information Systems and Artificial Intelligence, Institute of Computer Science III, University of Bonn, 53115 Bonn, Germany (e-mail: shams@iai.uni-bonn.de).

Juergen Gall is with the Department of Information Systems and Artificial Intelligence, Institute of Computer Science III, University of Bonn, 53115 Bonn, Germany, and also with the Lamarr Institute for Machine Learning and Artificial Intelligence, 53115 Bonn, Germany (e-mail: gall@iai.uni-bonn.de).

Ribana Roscher is with the Research Center Jülich, 52428 Jülich, Germany (e-mail: ribana.roscher@uni-bonn.de).

Digital Object Identifier 10.1109/TGRS.2023.3285401

layers in different deep learning architectures. We view our model as a generic architecture that can be used for other time-dependent forecasting tasks with static and dynamic variables. We conduct extensive experiments on the FireCube dataset [18] where our approach outperforms previous works. We achieve an overall improvement of up to **5.72%** in precision, **3.24%** in F1-score, **0.63%** in AUROC, and **1.15%** in OA on the test set.

The rest of this article is organized as follows. Section II reviews the related literature. Section III provides information about the used dataset. The proposed method is described in detail in Section IV. The experimental results and ablation study are provided in Sections V and VI, respectively. Finally, conclusions and outlook are given in Section VII.

II. Related Works

A. Wildfire Danger Forecasting

Wildfire forecasting or wildfire-susceptibility mapping from remote sensing and Earth observations data is a very important topic for wildfire management [11]. We briefly review some prior related works in this direction. Iban and Sekertekin [19], Pham et al. [20], and Gholami et al. [21] relied on traditional machine learning approaches to generate susceptibility maps. Shang et al. [22] and Mitsopoulos and Mallinis [23] utilized random forest (RF) classifiers. In their works, they studied the importance of biotic and abiotic predictors for wildfire forecasting. Jiang et al. [24] proposed a deep learning approach based on a multilayer perceptron (MLP) and included a comparison with traditional machine learning algorithms. In [25], a similar MLP-based approach was presented to generate a forest fire danger map. Zhang et al. [26] used a CNN and extended their work later to predict fire susceptibility at the global level [27]. Other works with CNN were conducted by Bjånes et al. [28] and Bergado et al. [29]. Furthermore, Huot et al. [30] approached the problem as a scene classification task using U-Net models to predict wildfire spreading. Their approach operates directly on the whole scene. A similar approach based on a U-Net++ model for global wildfire forecasting was proposed by Prapas et al. [31]. Yoon and Voulgaris [32] presented an approach that relies on a recurrent network with gated recurrent units (GRUs) to model past observations and on a CNN to predict wildfire probability maps for multiple time steps. More recently, Kondylatos et al. [13] and Prapas et al. [16] proposed to use long short-term memory (LSTM)-based approaches. They exploited both temporal and spatiotemporal contexts by applying recurrent LSTM and ConvLSTM models. They did not consider the whole scene at once but rather the classification of one pixel at a time (pixel-level).

Unlike these works, we do not treat all observation variables in the same way, but we propose a deep learning model that handles different types of variables in separated 2-D and 3-D CNN branches. In contrast to a ConvLSTM, which models spatial and temporal relations separately, a 3-D CNN models spatiotemporal relations jointly. We assume that the dataset contains static and dynamic variables, which we argue is the case for most datasets. Similar to Kondylatos et al. [13] and Prapas et al. [16], we also formulate the problem as pixel-level classification taking into account the spatiotemporal local context around the target pixel.

B. Multibranch Neural Networks

When deep learning is applied to potentially multisource remote sensing-based Earth observation data, multibranch neural networks are a commonly used framework. This is mainly because such networks enrich representation learning and provide discriminative learning perspectives of the input variables [33]. In addition, an important aspect of the multibranch design is the capability to adapt some parts of the model to a specific type of input. The general framework generates features from each branch and fuses these features in the network to obtain a unified feature vector. This fused representation is used as input to the subsequent layers. In [34], a two-branch 2-D CNN network was proposed to handle panchromatic information along with a multispectral one for image classification. Tan et al. [35] reduced the depth of a semantic segmentation classifier by applying consecutive blocks, each containing three CNN branches. A similar objective can be found in [33], where the network complexity was reduced via weight sharing and self-distillation (SD) embedding. In this way, only the main branch is used during inference, which inherits the knowledge of trained subbranches and has a close performance to an ensemble model. For hyperspectral image classification, Xu et al. [36] introduced a model called spectral–spatial unified network (SSUN). In their model, spectral features are learned by a grouping-based LSTM, and spatial features are learned by a 2-D CNN. Shen et al. [37] used separated spectral and spatial convolutional branches for hyperspectral input (S^2CDELM). They based their framework on the extreme learning machine (ELM). Unlike common backpropagation algorithms, they used a single hidden layer feedforward model. A multibranch architecture was also explored for image fusion. Liu et al. [38] proposed a two-stream CNN called StfNet. They investigated the task of spatiotemporal image fusion. Their network takes a coarse image input along with its neighboring images to predict the reconstructed fine image. Some works adapted a multibranch architecture to construct a multiscale feature vector. In [39], a dual-branch CNN with different filter kernel sizes was used as an autoencoder. Thus, the input image could be processed at different scales. Tang et al. [40] proposed a multiscale Gaussian pyramid to handle hyperspectral input. They used the Gaussian pyramid to obtain multiscale images that are then processed by ResNet modules [41]. In this way, spatial features can be learned at different scales. They further used a second branch that performs a discrete wavelet transform on the spectral input followed by an LSTM module [42]. The spatial and spectral features are then fused and processed by an MLP to obtain the final classification result. For short-term multitemporal image classification, Zheng et al. [43] addressed the task through a multitemporal deep fusion network (MDFN). In their framework, the LSTM-based branch is used to learn temporal–spectral features. At the same time, temporal–spatial

and spectral–spatial information is learned via a joint 3-D–2-D CNN with two branches. Furthermore, some works employed attention mechanisms with multibranch architectures [44], [45], [46]. When attention is applied, it drives the model to focus more on regions of interest. The former described methods [34], [35], [36], [37], [38], [39], [40], [43], [44] except those of Zhao et al. [33], Zhu et al. [45], and Deng et al. [46] did not consider linking information between branches during the feature learning stage. This limits the gradient flow and disentangles the correlations between the learned features.

This article proposes an architecture composed of two CNN branches for a forecasting task. A 2-D branch is used to learn spatial features from static variables. At the same time, a 3-D branch is used to learn spatiotemporal features from dynamic variables, which vary along the input temporal dimension. The branches are further linked via adaptive modulation layers to model the causal effects of static variables on dynamic variables.

C. Conditional Normalization

Since the introduction of normalization techniques in deep learning, they became a basic building block in many state-of-the-art models. Common normalization methods include batch normalization [47], group normalization [48], instance normalization [49], and layer normalization [50]. It has been shown empirically that normalization layers help with model optimization and regularization. Through normalization layers, the activation maps inside the model are normalized to follow a normal distribution with zero means. After that, the normalized activation maps are modulated or denormalized by learnable affine transformation parameters. These parameters vary across channels and are learned based on the running training statistics together with the model parameters. Therefore, such a normalization method is called unconditional. Compared to popular unconditional normalizations, there exist conditional normalization techniques that aim to learn affine parameters conditionally on external input. In the field of computer vision, conditional normalization is often used for image synthesis and style transformation [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [62], [63]. More recently, Marín and Escalera [64] adapted the conditional normalization from [51] and [52] to generate high-resolution satellite images. We use conditional normalization in a very different way than [52], [64], and [51]. While these works focus on synthesizing an image using the segmentation map as conditional input, we aim to modulate dynamic features conditioned on static features.

D. Temporal Positional Encoding

A plethora of studies exist about temporal modeling in remote sensing [65], [66], [67], [68]. Recently, the self-attention model, also known as transformer and first presented by Vaswani et al. [17] for natural language processing, has become a natural choice to handle sequential data, which includes a positional encoding. In the field of remote sensing, many works showed the benefits of adapting positional encoding for time-dependent image classification (Garnot et al. [69] and Garnot and Landrieu [70]), panoptic segmentation (Garnot and Landrieu [71]), and image generation (Dress et al. [72]). In their work, each image was given an encoded time vector according to its position with respect to a reference point, i.e., the first acquisition time step. Nyborg et al. [73] used the calendar time (day of the year) to provide positional information within the sequence. They also proposed to learn or estimate time shifts between geographically distant regions to enhance the generalization further. In another work, Nyborg et al. [74] used the thermal time, which is obtained by accumulating daily average temperatures over the growing season, for crop classification. In general, transformer-based approaches require positional encoding since the temporal information is otherwise lost within a transformer model. While 3-D CNNs consider the temporal order of the input such that a positional encoding is not necessary, we show in this work that adding an absolute temporal encoding (TE) is also useful for time-dependent forecasting.

III. Dataset

There are only very few publically available datasets for wildfire forecasting, and they differ significantly in the observational variables, the spatial and temporal resolutions, and the task that needs to be addressed. A related dataset is the Next Day Wildfire Spread dataset [30] where the task is to predict wildfire spread. It is formulated as a scene classification task and not as a pixelwise wildfire forecasting task as it is proposed in the FireCube dataset [18] and addressed in this work. The FireCube dataset was first published in [16] and extended later in [13]. It includes multivariate spatiotemporal data streams with 90 variables from the years 2009–2021 with a resolution of 1 km $\times$ 1 km $\times$ 1 day. The area is 1253 km $\times$ 983 km, covering parts of the Eastern Mediterranean. The observational variables include meteorological data [75], satellite-derived products [76], [77], topographic features [78], human-related activities [79], and historical fire records [80], [81]. In addition, Copernicus Corine Land Cover (CLC) [82] and Fire Weather Index (FWI) [83] are provided. The target is to predict for each pixel if a wildfire will ignite and become large (>0.3 km^2) in the next day. The task is equivalent to binary classification, where the positive class represents a wildfire event.

Since wildfire forecasting is essentially an imbalanced classification task, Kondylatos et al. [13] extracted samples as follows. For a target day $T+1$, the static variables form a patch of 25 km $\times$ 25 km centered around the target pixel at day T. In contrast, the dynamic variables consist of 25 km $\times$ 25 km $\times$ 10 days time series of observations from days $T-9$ until T. For each positive sample, a few negative samples from different locations are sampled. Although the negatives are from different locations, they are sampled from regions that have a similar land cover distribution as the positive samples to make the task more difficult.

Overall, the dataset includes 71 471 samples for training (13 518 positives and 57 953 negatives for the years 2009–2018), 6430 samples for validation (1300 positives and

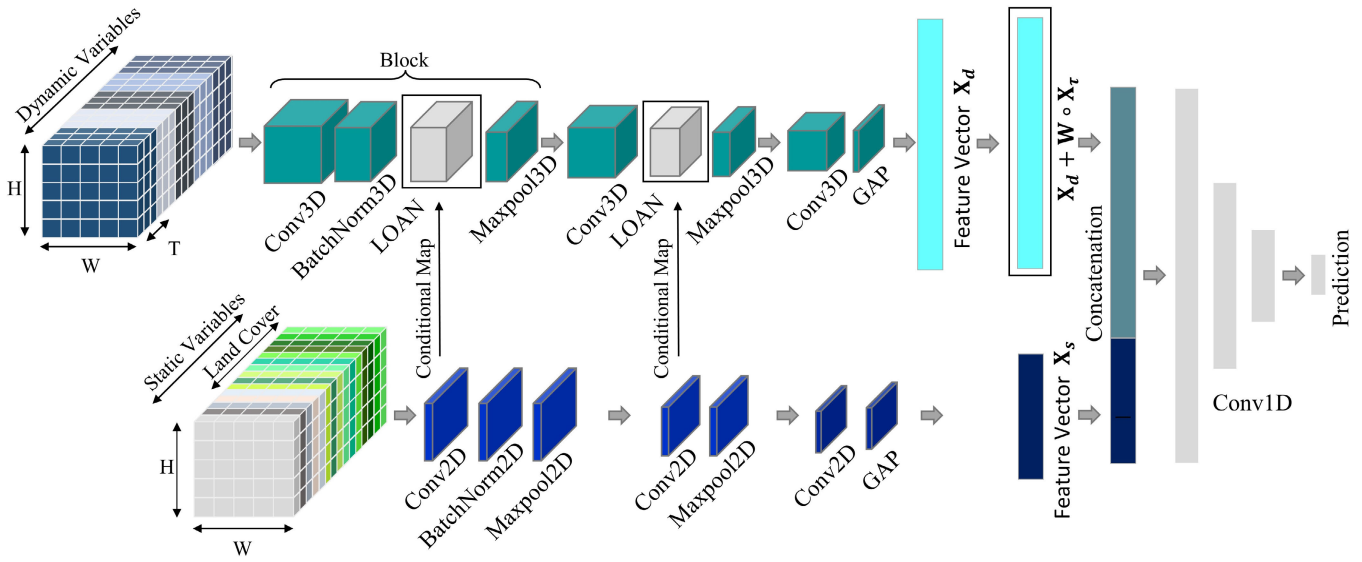


Fig. 1. Overview of the proposed approach. Our network handles static and dynamic variables by two branches. For the static branch, we use 2-D convolutions, whereas we use 3-D convolutions for the dynamic branch since the dynamic variables change over time. Since dynamic variables, such as surface temperature strongly, depend on static variables, such as elevation, we modulate the dynamic features conditioned on the static features at two blocks (LOAN). This makes the model location-aware since the static variables contain geographic data like a land cover. The feature vector $\mathbf{X}_d$ from the dynamic branch is combined with a weighted TE vector $\mathbf{X}_\tau$ before it is concatenated with the static feature vector $\mathbf{X}_s$. The concatenated vector serves as input to fully connected layers that predict the probability of a wildfire.

5130 negatives for the year 2019) and 42 820 samples for testing (1228 positives and 4860 negatives for the year 2020 and 4407 positives and 32 325 negatives for the year 2021). The year 2021 in the test set contains an extreme wildfire season in Greece [13], [84]. The extracted samples are available in [85].

In this article, we use from the described dataset the same variables as in [13]. This includes the following.

1) *15 Static Variables:*
   a) Digital elevation model (DEM) and slope [78].
   b) Distance to roads, distance to the waterway, and population density [79].
   c) Copernicus CLC variables representing the fractions of classes for each pixel. This gives ten variables per pixel [82].
2) *Ten Dynamic Variables:*
   a) Day and night land surface temperatures [77].
   b) Normalized difference vegetation index (NDVI) [76].
   c) Soil moisture index [86].
   d) Maximum 2 m temperature, maximum wind speed, minimum relative humidity, total precipitation, maximum 2 m dew point temperature, and maximum surface pressure [75].

IV. Methodology

Problem Formulation: Given a multivariate spatiotemporal data cube $\mathcal{C}(\{V, T, W, H\})$, where H and W are the spatial extensions of the cube, T is the temporal extension in the past for the time series $1, 2, \ldots, T$, and V is the number of variables (static and dynamic), our aim is to learn a mapping function f approximated by a neural network that can predict the probability $Y_{T+1} \in [0, 1]$ of a wildfire event to start at the center of $W \times H$ for the target day $T+1$

$$f : \mathcal{C}(\{V, T, W, H\}) \rightarrow \ Y_{T+1}. \tag{1}$$

To achieve this, we propose a spatiotemporal 2-D/3-D CNN with two branches, as illustrated in Fig. 1. First, the network design is introduced in Section IV-A. Then, the LOAN layer, which is the core of our work, is explained in detail in Section IV-B. Finally, Section IV-C describes the integration of the absolute TE into the model.

A. 2-D/3-D Two-Branch CNN

As shown in Fig. 1, our network consists of two branches that process dynamic and static variables, respectively. We denote the data cube with dynamic variables by $\mathcal{C}(\{V_d, T, W, H\})$ and the data cube with static variables by $\mathcal{C}(\{V_s, W, H\})$. As in previous works, we normalize the input channelwise to the range $[0, 1]$. Since the static variables do not have a time component, we use 2-D convolutions for the static branch and 3-D convolutions for the dynamic branch.

1) Dynamic Branch: More in detail, the dynamic branch takes the variables V_d that vary over time as input. It consists of three blocks; each block has a 3-D convolution with a 3×3 kernel size followed by a rectified linear unit (ReLU) activation function and a 3-D max pooling layer. To reduce overfitting, we use global average pooling (GAP) [87] at the end of the last block. We denote the feature vector learned from this branch as $\mathbf{X}_d \in \mathbb{R}^{256}$.

2) Static Branch: In parallel to the dynamic branch, the static branch has a similar architecture. However, 2-D convolutions are used instead of 3-D ones. We denote the feature vector learned from this branch as $\mathbf{X}_s \in \mathbb{R}^{128}$. Note that the dimensionality of the static feature vector is lower than the dimensionality of the dynamic feature vector since the input data cube is smaller.

In a nutshell, the dynamic and static branches are two functions f_d and f_s, respectively,

$$f_d : \mathcal{C}(\{V_d, T, W, H\}) \rightarrow \mathbf{X}_d \tag{2}$$

$$f_s : \mathcal{C}(\{V_s, W, H\}) \rightarrow \mathbf{X}_s. \tag{3}$$

For the dynamic feature vector, we add an absolute TE $\mathbf{X}_\tau$ that will be described in Section IV-C. The two feature vectors are then concatenated and fed into four classification layers with 1-D convolutions of kernel size 1. The layers reduce the dimensionality from 384 to 256, 128, 32, and 2. To reduce overfitting, we use dropout with a dropout probability $p = 0.5$ for the 1-D convolutional layers except for the last two layers. Finally, a softmax activation is used after the last classification layer to predict the probability of a wildfire. In addition, we use a batch normalization layer [47] for the first block of each branch. More implementation details are given in Section V.

For training, we use the binary cross-entropy as loss function

$$-\frac{1}{N}\sum_{n=1}^{N}\left[\hat{Y}_{T+1}^{(n)}\log\left(Y_{T+1}^{(n)}\right)+\left(1-\hat{Y}_{T+1}^{(n)}\right)\log\left(1-Y_{T+1}^{(n)}\right)\right] \tag{4}$$

where N denotes the batch size and $\hat{Y}_{T+1}^{(n)} \in \{0, 1\}$ is the true label for sample n.

In the following, we discuss the LOAN that modulates the dynamic features based on the static features and the already mentioned absolute TE.

B. Location-Aware Adaptive Normalization

In general, dynamic variables are correlated with the geographic location, i.e., temperature and pressure change with elevation, soil moisture, and NDVI vary with land cover, and humidity is correlated with some static variables, such as the waterway distance. Since the dynamic variables depend on the static variables and not vice versa, we aim to exploit this knowledge in our approach. This is done by learning to normalize the dynamic features based on the location-dependent static features. To this end, we propose a conditional normalization technique for remote sensing data called LOAN.

We first describe a batch normalization [47] where the activation map is normalized before it is modulated by scale γ and bias β. Let ${z_d}^i \in \mathbb{R}^{N\times K^i\times D^i\times W^i\times H^i}$ be an activation map in the ith block of the dynamic branch and ${z_s}^i \in \mathbb{R}^{N\times K^i\times W^i\times H^i}$ be an activation map of the corresponding ith block in the static branch, where K^i denotes the number of channels and D^i, W^i, and H^i denote the depth, width, and height of the activation map z^i, respectively. Using the indices $n \in \{1, \ldots, N\}$, $k \in \{1, \ldots, K^i\}$, $t \in \{1, \ldots, D^i\}$, $w \in \{1, \ldots, W^i\}$, and $h \in \{1, \ldots, H^i\}$, the normalization of the dynamic branch is performed by the following equation:

$$\left(\frac{{z_d}^i_{(n,k,t,w,h)} - \mu_k}{\sigma_k}\right)\cdot\gamma_k + \beta_k \tag{5}$$

where ${z_d}^i_{(n,k,t,w,h)}$ is the activation before the normalization, μ_k and σ_k are the computed mean and standard deviation of channel k, i.e., computed over the tensor $D^i \times W^i \times H^i$ and all samples n in the batch, and γ_k and β_k are the learnable modulation parameters.

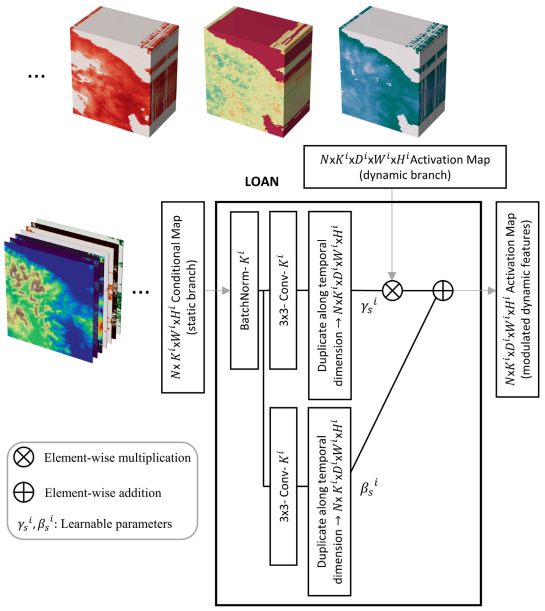


Fig. 2. Illustration of the LOAN layer when using the conditional map from the static branch. The conditional map of the static branch and the activation map of the dynamic branch have the same N, K^i, W^i, and H^i dimensions. BatchNorm denotes a batch normalization layer. 3×3-Conv-K^i denotes a convolution layer with a kernel size of 3×3 and K^i output channels.

In our case, we aim to learn a modulation of the dynamic features ${z_d}^i_{(n,k,t,w,h)}$ at the ith block where the modulation parameters ${\gamma_s}^i_{(n,k,w,h)}$ and ${\beta_s}^i_{(n,k,w,h)}$ depend on the corresponding static features ${z_s}^i_{(n,k,w,h)}$

$${z_d}^i_{(n,k,t,w,h)} \cdot {\gamma_s}^i_{(n,k,w,h)} + {\beta_s}^i_{(n,k,w,h)}. \tag{6}$$

In contrast to (5), the modulation parameters ${\gamma_s}^i_{(n,k,w,h)}$ and ${\beta_s}^i_{(n,k,w,h)}$ vary with respect to sample n in the batch, the location (w, h), and across channel k, but they are constant over time t. Furthermore, they are conditioned on the static features ${z_s}^i$. We, thus, call ${z_s}^i$ the conditional map for the modulation.

The way how ${\gamma_s}^i_{(n,k,w,h)}$ and ${\beta_s}^i_{(n,k,w,h)}$ are computed is illustrated in Fig. 2. First, the conditional map ${z_s}^i$ is normalized channelwise as follows:

$$\hat{z}_s{}^i_{(n,k,w,h)} = \frac{{z_s}^i_{(n,k,w,h)} - {\mu_k}^i}{{\sigma_k}^i} \tag{7}$$

where

$${\mu_k}^i = \frac{1}{NW^iH^i}\sum_{n=1}^{N}\sum_{w=1}^{W^i}\sum_{h=1}^{H^i}{z_s}^i_{(n,k,w,h)} \tag{8}$$

$${\sigma_k}^i = \sqrt{\frac{1}{NW^iH^i}\sum_{n=1}^{N}\sum_{w=1}^{W^i}\sum_{h=1}^{H^i}\left({z_s}^i_{(n,k,w,h)} - {\mu_k}^i\right)^2}. \tag{9}$$

Afterward, $\hat{z}_s{}^i_{(n,k,w,h)}$ is projected by two convolutional layers with K^i filters to compute ${\gamma_s}^i_{(n,k,w,h)}$ and ${\beta_s}^i_{(n,k,w,h)}$. In our implementation, these modulation parameters are then duplicated along the temporal dimension to match the depth D^i of ${z_d}^i$ such that (6) can be computed.

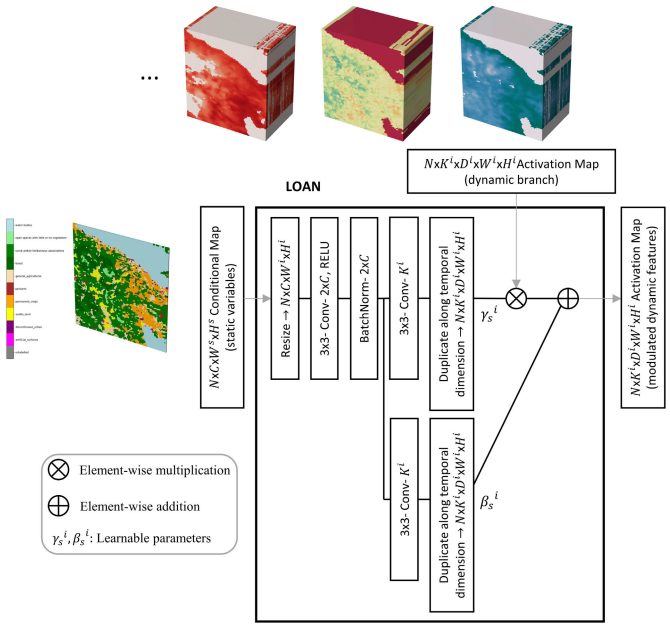


Fig. 3. Illustration of the LOAN layer when using C static variables directly for modulation. In this case, there is a mismatch between the dimensions of the conditional map and the activation maps of the dynamic branch at the ith block. To adjust for the spatial resolution, we resize the conditional map using nearest-neighbor downsampling to match the resolution of the activation map from the dynamic branch. The convolution layer takes C channels as input and generates an output with K^i channels.

We add the LOAN layer in the first two blocks, as shown in Fig. 1. The activation maps of the dynamic branch are normalized only in the first block and modulated in both the first and second blocks. The impact of the blocks where the LOAN layer is added is evaluated in Section VI-A.

In the experimental section, we also evaluate a variant of LOAN that is not conditioned on the intermediate features of the static branch, as shown in Figs. 1 and 2, but on the static variables directly, as shown in Fig. 3. In this case, the conditional map has different spatial dimensions and number of channels compared to the features in the dynamic branch, i.e., the conditional maps consist of C variables and have $W^s \times H^s$ spatial dimensions. In this respect, the conditional map ($W^s \times H^s$) is first resized to match the spatial dimensions ($W^i \times H^i$) of the activation map from the dynamic branch. We use the nearest-neighbor method for the downsampling. The resized conditional map is then fed into a convolutional layer with 3×3 kernel size to double the number of channels, i.e., $2 \times C$. Finally, as in the previous version of LOAN, the conditional map is normalized, projected by two convolutional layers, and duplicated along the temporal dimension to compute $\gamma_s{}^i_{(n,k,w,h)}$ and $\beta_s{}^i_{(n,k,w,h)}$. The impact of the blocks where the LOAN layer is added is evaluated in Section V-C.

C. Absolute Temporal Encoding

Some extreme events in the climate model have a dependent relation on time [14], [88]. This is also the case for the FireCube dataset [18] where wildfire events vary from month to month and occur more frequently in the summer time, as shown in Fig. 4. So far, the network does not consider

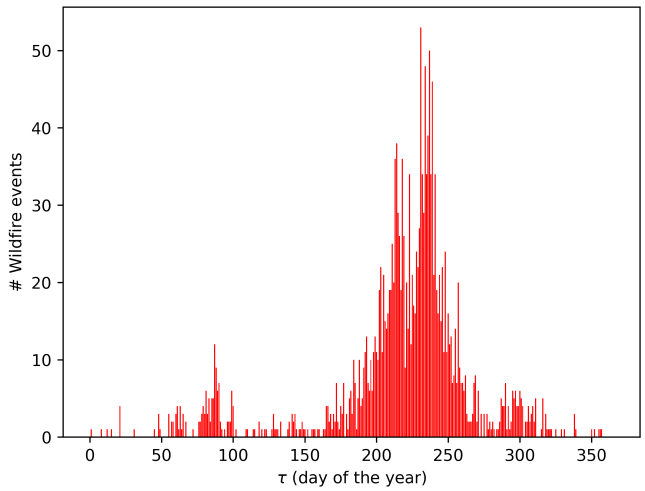


Fig. 4. Distribution of the wildfire events per day between March 6, 2009, and August 29, 2021 (FireCube dataset).

an absolute time like the day of the year. Instead, for any forecast day $T+1$, the last ten days are used as observations, but the network does not have the information on what day during the year T is.

As shown in Fig. 1, we add this information to the dynamic branch before we concatenate the static and dynamic features. To encode the day of the year, we use the standard fixed sinusoidal-based encoding by Vaswani et al. [17]. We precompute for each day of the year $\tau \in [0, 365]$,[1] which is extracted from T, the absolute TE vector $\mathbf{X}_\tau \in \mathbb{R}^{256}$

$$\mathbf{X}_\tau(2j) = \sin(\tau/10^{2j/256}) \tag{10}$$

$$\mathbf{X}_\tau(2j+1) = \cos(\tau/10^{2j/256}) \tag{11}$$

where j is the embedding dimension. Each even dimension results from a sine function, while each odd dimension results from a cosine function. This allows τ to have a smooth and yet unique encoding for every time step, i.e., each day of a year. Note that the vector has the same size as $\mathbf{X}_d$.

In order to add the absolute time encoding vector $\mathbf{X}_\tau$ to the dynamic feature vector $\mathbf{X}_d$, we weight each element of the vector by a learnable weight vector $\mathbf{W} \in \mathbb{R}^{256}$

$$\mathbf{X}_d + \mathbf{W} \circ \mathbf{X}_\tau \tag{12}$$

where $\circ$ denotes the Hadamard product, i.e., elementwise multiplication. Fig. 5 illustrates how the temporal embedding is added to the model.

V. Experimental Results and Analysis

Implementation Details: The network is trained with the binary cross-entropy loss (4) using the Pytorch framework [89] with a learning rate of 0.00003 and the Adam optimizer ($\beta_1 = 0.9$ and $\beta_2 = 0.999$) [90] with a weight decay of 0.02. We use a batch size of $N = 256$ and train the network for 40 epochs. All models were trained on a single NVIDIA GeForce RTX 3090 GPU.

Performance Metrics: As described in Section III, we use the FireCube dataset [18]. We follow the same protocol for

[1] We consider February 29 for the encoding.

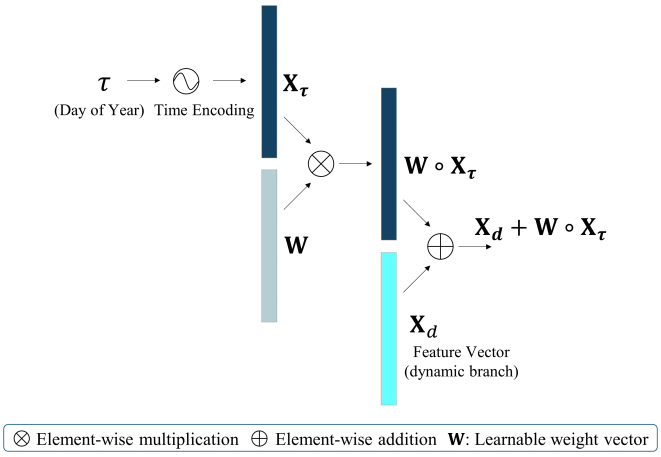


Fig. 5. Illustration of how the absolute TE is added in the model. The day of year τ is encoded into a vector $\mathbf{X}_\tau$, and each element is weighted by the learned weight vector $\mathbf{W}$. The weighted vector is then added to the dynamic feature vector $\mathbf{X}_d$.

quantitative comparison as in [13] and [16]. The evaluation metrics are precision, recall, and F1-score, calculated for the positive class that represents a wildfire event. In addition, we report true positives (TPs), FPs, true negatives (TNs), and false negatives (FNs). Moreover, we provide the overall accuracy (OA) and the area under the receiver operating characteristic curve (AUROC) as evaluation metrics. OA is the accuracy obtained on all negative and positive samples in the test set. The AUROC describes the TP rate (TPR) against the FP rate (FPR) within multiple thresholds in one value.

A. Comparison With Baselines

We compare our approach to the approaches that have been evaluated on the described dataset in [13]. This includes two deep learning models, namely, LSTM [42] and ConvLSTM [65], and two classical machine learning classifiers, namely, RFs [93] and XGBoost [94]. For further details regarding the architectures and hyper-parameters of the models, we refer to the work of Kondylatos et al. [13]. In order to demonstrate the benefit of treating static and dynamic variables differently, we also compare with a one-branch 3-D CNN where we duplicate the static variables along the temporal dimension and concatenate them together with the dynamic variables to form a single data cube. In addition, we compare to the recent transformer models TimeSformer [91] and Video SwinTransformer [92], which use space–time attention. As mentioned in Section II-D, vision transformers are based on a self-attention mechanism to model spatiotemporal dependencies. Compared to CNNs, transformers have less inductive bias and need much more computational resources for training. To the best of our knowledge, no prior work has done a systematic study about the performance of video vision transformers for wildfire forecasting.

To ensure a fair comparison, all baselines were reimplemented and trained on the same samples with a fixed random seed. We do not use any augmentation technique. The quantitative results of our experiments are provided in Table I. The results of the proposed 2-D/3-D CNN are shown with and without absolute TE.

We can observe that the proposed 2-D/3-D CNN outperforms the other methods for most metrics, particularly FP, TN, precision, F1-score, and OA, on the validation and testing sets. SwinTransformer and ConvLSTM achieve slightly higher AUROC for 2019 and 2020, respectively. LSTM and SwinTransformer achieve a higher recall but at the cost of very low precision. In comparison with other deep learning methods, LSTM and SwinTransformer have even the highest number of FPs for all years. The main weakness of LSTM lies in the fact that it does not consider the spatial context, while large models, such as SwinTransformer, are prone to overfitting, which results in a relatively poor performance for 2020. RF and XGBoost have the same disadvantage as LSTM but even a weaker temporal model and, thus, perform worse than LSTM. Most interesting is the comparison to 3-D CNN since it uses the same 3-D CNN structure but only one branch, i.e., it treats static variables, such as dynamic variables. The results show that the proposed approach with two branches outperforms the single-branch architecture for all metrics and all years. This demonstrates the importance of treating static variables differently than dynamic variables. Adding the absolute TE to the model substantially reduces FP at the cost of decreasing TP. This is also reflected in the precision and recall.

In Table II, we present additional experimental results alongside the memory footprint as the number of parameters (# Params), the estimated multiply-and-accumulate operations (multiply-adds) (MMACs), and the expected inference time, which is estimated as samples per millisecond (# SPmS). The performance metrics are calculated on both testing years 2020 and 2021 as one set. Since the LOAN layers increase the number of parameters of the 2-D/3-D CNN, we report the results of the one-branch 3-D CNN using 323k and 499k parameters. The smaller 3-D CNN has about the same amount of parameters as the 2-D/3-D CNN without LOAN, whereas the larger 3-D CNN has more parameters than the proposed model. Due to the lack of spatial modeling, LSTM has the fewest parameters and is the fastest, but the precision is very low. ConvLSTM, the small one-branch 3-D CNN, and 2-D/3-D CNN without LOAN and TE, which consider the spatial context, perform similarly, but 2-D/3-D CNN is the fastest approach, and ConvLSTM is the slowest approach among them. Compared to the other approaches, transformer models have considerably more parameters and require more operations, which makes them computationally expensive. Nevertheless, our approach outperforms both transformer models while requiring far fewer parameters and computational operations. Normalizing the dynamic features conditioned on the static features (LOAN) increases all metrics. It also outperforms the large 3-D CNN in all metrics and inference time. Adding TE increases all metrics when LOAN is not used while increasing the computational cost only very little. When LOAN is used, adding TE decreases the recall but increases all other metrics. Since LOAN and TE change the dynamic features, we observe a different tradeoff between recall and precision if both are used. This change is consistent over the years, as shown in Table I. Nevertheless, the F1-score, AUROC, and OA are highest if both are used.

TABLE I

COMPARISON WITH BASELINES. THE CLASSIFICATION METRICS ARE SHOWN FOR THE YEARS 2019–2021. THE VALUES OF PRECISION, RECALL, F1-SCORE, OA, AND AUROC ARE GIVEN IN PERCENT (%). TE DENOTES THE ABSOLUTE TEMPORAL ENCODING

Year 2019 (val) - 6430 samples									
Algorithm	TP(↑)	FP(↓)	TN(↑)	FN(↓)	Precision(↑)	Recall(↑)	F1-score(↑)	AUROC(↑)	OA(↑)
RF [13]	575	372	4758	725	60.72	44.23	51.18	88.54	82.94
XGBoost [13]	928	448	4682	372	67.44	71.38	69.36	92.33	87.25
LSTM [13]	968	431	4699	332	69.19	74.46	71.73	93.63	88.13
ConvLSTM [13]	867	276	4854	433	75.85	66.69	70.98	94.69	88.97
TimeSformer [91]	967	248	4882	333	79.59	74.38	76.90	**95.71**	90.96
SwinTransformer [92]	**979**	324	4806	**321**	75.13	**75.31**	75.22	94.54	89.97
3D CNN	918	265	4865	382	77.60	70.62	73.94	94.17	89.94
2D/3D CNN	970	248	4882	330	79.64	74.62	**77.05**	94.52	91.01
2D/3D CNN w/ TE	905	**182**	**4948**	395	**83.26**	69.62	75.83	95.08	**91.03**

Year 2020 (test) - 6088 samples									
Algorithm	TP(↑)	FP(↓)	TN(↑)	FN(↓)	Precision(↑)	Recall(↑)	F1-score(↑)	AUROC(↑)	OA(↑)
RF [13]	750	245	4615	478	75.38	61.07	67.48	91.17	88.12
XGBoost [13]	**891**	322	4538	**337**	73.45	**72.56**	73.00	91.12	89.18
LSTM [13]	**891**	290	4570	**337**	75.44	**72.56**	73.97	93.60	89.70
ConvLSTM [13]	811	155	4705	417	83.95	66.04	73.93	**94.31**	90.60
TimeSformer [91]	751	140	4720	477	84.29	61.16	70.88	92.41	89.87
SwinTransformer [92]	794	202	4658	434	79.72	64.66	71.40	92.79	89.55
3D CNN	797	160	4700	431	83.28	64.90	72.95	93.10	90.29
2D/3D CNN	841	160	4700	387	84.02	68.49	**75.46**	93.98	**91.02**
2D/3D CNN w/ TE	776	**117**	**4743**	452	**86.90**	63.19	73.17	94.20	90.65

Year 2021 (test) - 36732 samples									
Algorithm	TP(↑)	FP(↓)	TN(↑)	FN(↓)	Precision(↑)	Recall(↑)	F1-score(↑)	AUROC(↑)	OA(↑)
RF [13]	3264	1157	31168	1143	73.83	74.06	73.95	96.82	93.74
XGBoost [13]	3016	1345	30980	1391	69.16	68.44	68.80	95.88	92.55
LSTM [13]	3739	1359	30966	668	73.34	84.84	78.67	97.13	94.48
ConvLSTM [13]	3514	769	31556	893	82.05	79.74	80.87	97.76	95.48
TimeSformer [91]	3578	867	31458	829	80.49	81.19	80.84	97.67	95.38
SwinTransformer [92]	**3962**	954	31371	**445**	80.59	**89.90**	84.99	98.09	96.19
3D CNN	3766	810	31515	641	82.30	85.45	83.85	98.02	96.05
2D/3D CNN	3870	757	31568	537	83.64	87.81	85.68	98.19	96.48
2D/3D CNN w/ TE	3841	**416**	**31909**	566	**90.23**	87.16	**88.67**	**98.54**	**97.33**

TABLE II

QUANTITATIVE RESULTS OF DIFFERENT DEEP LEARNING MODEL DESIGNS. THE CLASSIFICATION METRICS ARE GIVEN IN PERCENT (%). IN ADDITION, THE TOTAL NUMBER OF PARAMETERS (# PARAMS), ESTIMATED MULTIPLY-AND-ACCUMULATE OPERATIONS GIVEN IN MEGA (MMACS), AND THE INFERENCE TIME AS SAMPLES PER MILLISECOND (# SPMS) ARE PROVIDED. TE DENOTES THE ABSOLUTE TEMPORAL ENCODING AND THE LOAN LAYER

Year 2020-2021 (test)										
Algorithm	LOAN	TE	# Params(↓)	MMACs (↓)	# SPmS(↑)	Precision(↑)	Recall(↑)	F1-score(↑)	AUROC(↑)	OA(↑)
LSTM [13]	×	×	30k	0.27	955±43	73.74	82.17	77.72	96.53	93.80
ConvLSTM [13]	×	×	372k	417.08	7±0	82.40	76.75	79.47	97.12	94.78
TimeSformer [91]	×	×	1.16m	831,667.58	2±0	81.13	76.82	78.92	96.57	94.60
SwinTransformer [92]	×	×	1.78m	122,346.56	1±0	80.45	**84.40**	82.38	97.06	95.25
3D CNN	×	×	323k	476.56	18±1	83.93	77.48	80.58	97.15	95.08
	×	×	499k	585.11	17±1	82.47	80.98	81.72	97.15	95.23
2D/3D CNN	×	×	321k	137.24	47±0	83.35	79.98	81.63	97.11	95.26
	√	×	413k	168.23	33±1	83.71	83.60	83.65	97.41	95.70
	×	√	321k	137.24	47±1	83.90	84.22	84.06	97.64	95.80
	√	√	414k	168.23	33±2	**89.65**	81.93	**85.62**	**97.78**	**96.38**

B. Variable Importance

To assess the importance of different static variables, we present the results obtained with different combinations of static variables in Table III. For this experiment, we use 2-D/3-D CNN with LOAN but without TE. All dynamic variables are used in this experiment, and the results are reported for the year 2019 and for the years 2020 and 2021 as one set. The static variables are grouped into three

TABLE III
PERFORMANCE METRICS FOR DIFFERENT CATEGORIES OF STATIC VARIABLES. THE VALUES ARE GIVEN IN PERCENT (%)

	Year 2019 (val)					Year 2020-2021 (test)				
Static Variables	Precision(↑)	Recall(↑)	F1-score(↑)	AUROC(↑)	OA(↑)	Precision(↑)	Recall(↑)	F1-score(↑)	AUROC(↑)	OA(↑)
DEM + slope	75.61	73.69	74.64	93.82	89.88	80.88	81.67	81.27	97.06	95.05
Distance to roads Distance to waterway Population density	77.27	76.08	76.67	94.16	90.63	81.91	74.48	78.02	96.11	94.48
Land cover	77.27	76.62	76.94	94.62	90.72	81.85	79.93	80.88	96.92	95.03
DEM + slope Distance to roads Distance to waterway Population density	75.09	**77.00**	76.03	94.09	90.19	79.64	76.89	78.24	96.69	94.37
DEM + slope Land cover	74.61	73.46	74.03	94.24	89.58	80.05	81.44	80.74	97.20	94.89
Land cover Distance to roads Distance to waterway Population density	77.52	73.23	75.32	**94.52**	90.30	82.63	81.12	81.87	96.98	95.27
All static variables	**79.64**	74.62	**77.05**	**94.52**	**91.01**	**83.71**	**83.60**	**83.65**	**97.41**	**95.70**

TABLE IV
IMPACT OF DIFFERENT CONDITIONAL MAPS ON THE FEATURE MODULATION. THE EVALUATION METRICS ARE GIVEN IN PERCENT (%)

	Year 2019 (val)					Year 2020-2021 (test)				
Conditional Map	Precision(↑)	Recall(↑)	F1-score(↑)	AUROC(↑)	OA(↑)	Precision(↑)	Recall(↑)	F1-score(↑)	AUROC(↑)	OA(↑)
–	75.68	62.23	68.30	93.38	88.32	83.35	79.98	81.63	97.11	95.26
DEM + slope	70.84	71.00	70.92	92.77	88.23	76.90	79.45	78.15	96.16	94.15
Distance to roads Distance to waterway Population density	73.83	**76.15**	74.97	93.74	89.72	77.98	76.63	77.30	96.22	94.08
Land cover	79.05	72.00	75.36	94.58	90.48	81.53	75.19	78.23	96.80	94.49
All static variables	77.12	74.92	76.00	**94.95**	90.44	81.05	**84.08**	82.54	**97.55**	95.32
Activation maps (Static branch)	**79.64**	74.62	**77.05**	94.52	**91.01**	**83.71**	83.60	**83.65**	97.41	**95.70**

main categories: topographic variables consisting of DEM and slope, anthropogenic-related variables consisting of distance to roads or waterways and population density, and land cover variables.

From the results in Table III, we can conclude that, among the three categories, topographic variables give the best results for the years 2020 and 2021 when they are used without other variables, while the land cover and anthropogenic-related variables provide the best results for the year 2019. Overall, all static variables are relevant, and the best results are obtained when all static variables are used (last row). This is also better than using the static variables of two out of the three categories, which is reported in rows 4–6 of Table III.

C. Comparing Different Conditional Maps

While Table III shows the importance of different static variables as input to the 2-D/3-D CNN, we also analyze the impact of different ways to modulate the dynamic features in Table IV. For the experiments, we use the 2-D/3-D CNN without TE and all dynamic and static variables as input. While we use all variables, the different settings differ in the input that is fed to the LOAN layer, i.e., the static features that the modulation of the dynamic features is conditioned on.

The results of the proposed conditioning, where we use the features from the corresponding block of the static branch, are shown in the last row. In the first row, we show the results if we do not use LOAN at all, i.e., we do not use any modulation of the dynamic features. For the other rows of Table IV, we modify LOAN such that it is not conditioned on the intermediate features of the static branch, as shown in Figs. 1 and 2. Instead, we condition LOAN directly on static variables. Note that we need to slightly adapt LOAN, as shown in Fig. 3, since the number of static input variables C differs from the number of feature channels K at the block where LOAN is added.

As seen from Table IV, the best result is obtained when we use the activation maps, i.e., the intermediate features, from the static branch for conditioning the modulation. However, comparable results are obtained when all static variables are directly used by the variant of LOAN that is shown in Fig. 5. While this variant achieves slightly higher AUROC, the variant shown in Fig. 3 achieves higher F1-score and OA. Using the static variables directly, we can analyze how the three categories of static variables impact the modulation of the dynamic features and, thus, the results. We can conclude that the modulation of the dynamic features is very sensitive to its

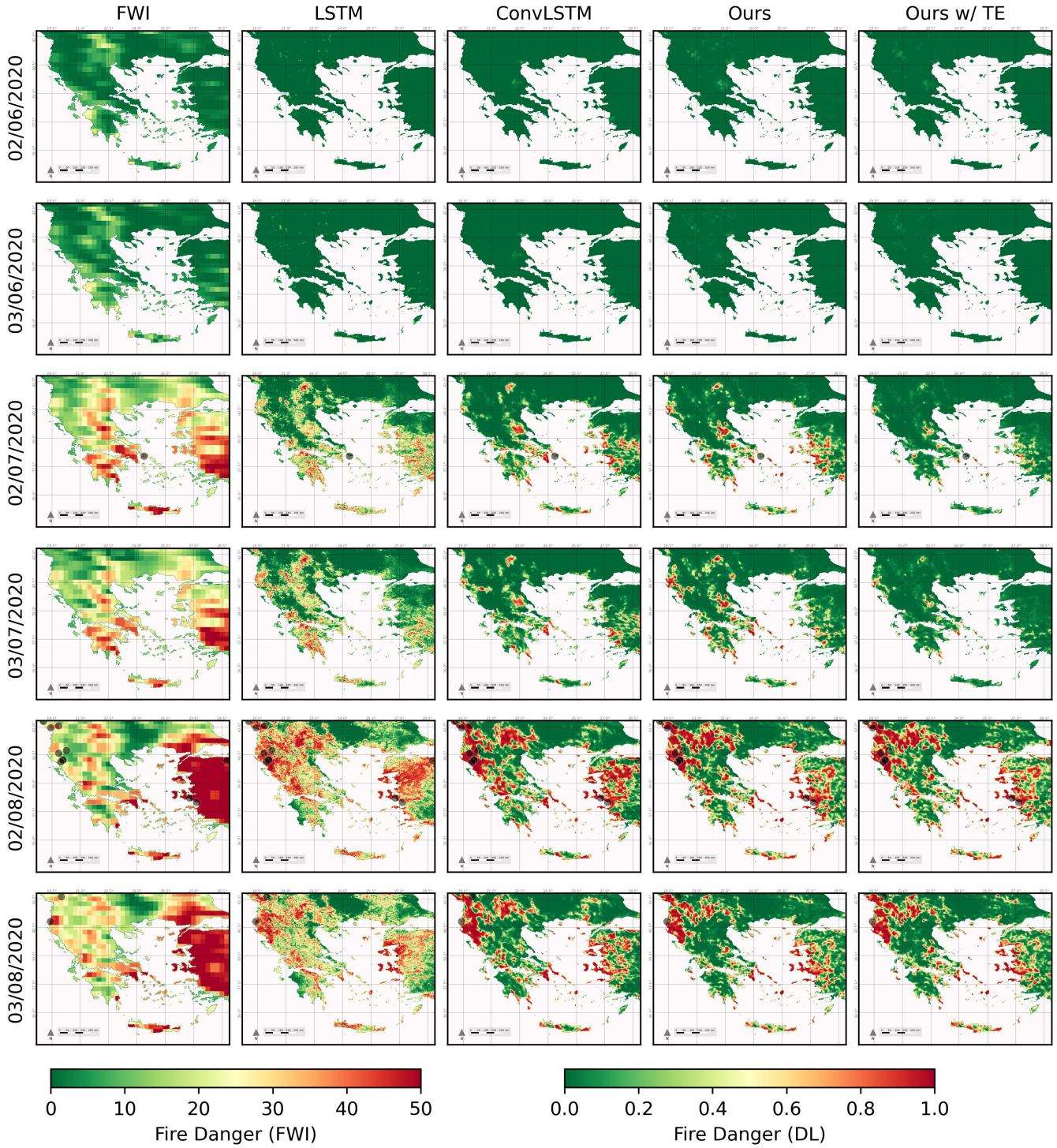


Fig. 6. Qualitative results for six days during the wildfire season in year 2020. The black circles represent an ignition of a large wildfire on that day. TE denotes the absolute temporal encoding.

condition. For the year 2019, all three categories (rows 2–4) improve the results compared to the setup without feature modulation (first row). For the years 2020 and 2021, this is not the case, and only the combination of all static variables (row 5) leads to an improvement. The reason is the mismatch between C and K^i. Compared to the other categories, land cover has the highest number of variables per pixel (C=10) and shows the best performance. This indicates that conditioning the modulation of the dynamic features on the intermediate features of the static branch is a more practical approach than conditioning the modulation on the static variables directly, which seems to be sensitive to the number of variables.

D. Qualitative Results

Predicted wildfire danger-susceptibility maps are depicted in Figs. 6–10. We take input from the first and second days of three consecutive months in summer (June, July, and

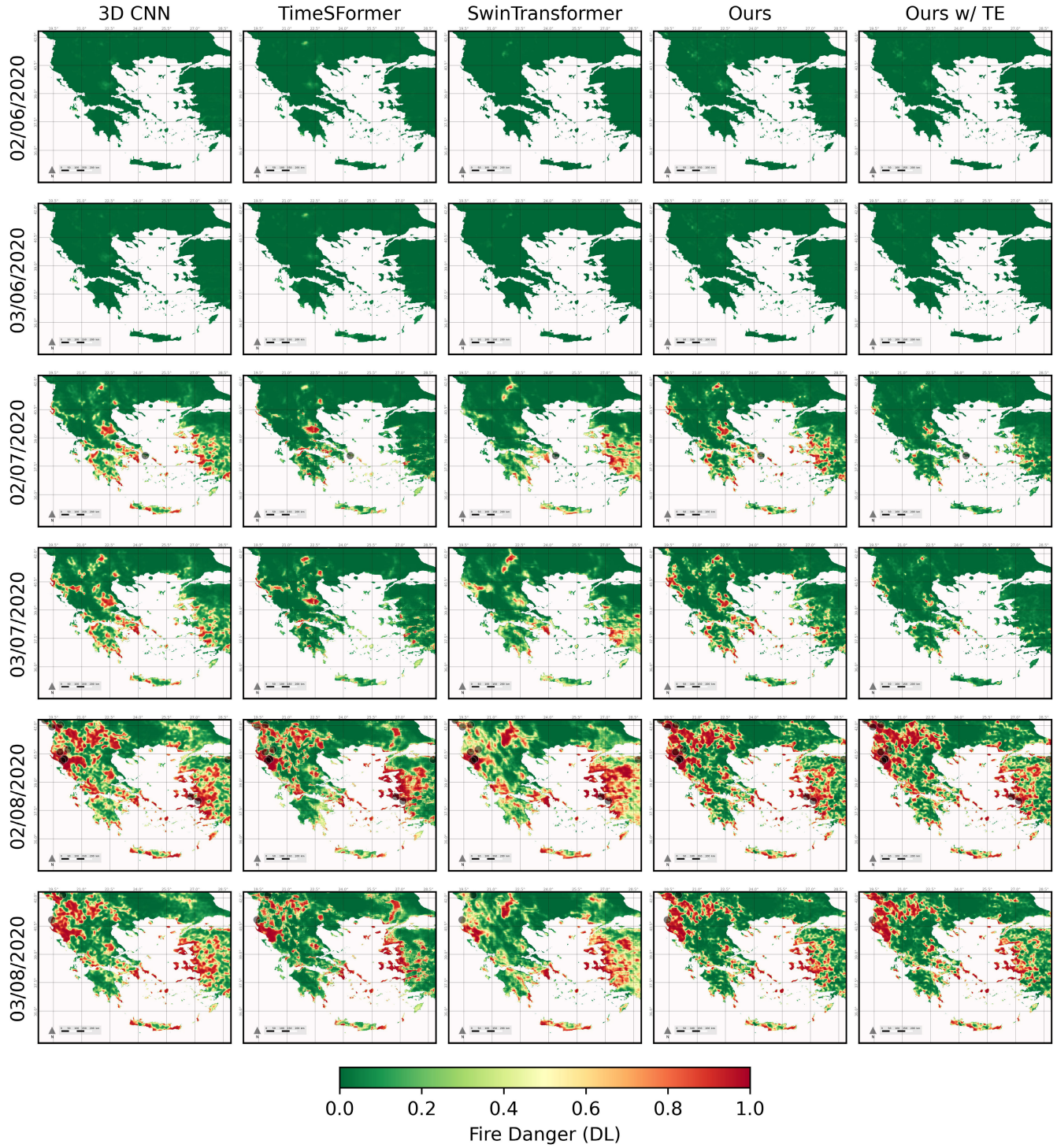


Fig. 7. Qualitative results for six days during the wildfire season in year 2020. The black circles represent an ignition of a large wildfire on that day. TE denotes the absolute temporal encoding.

August) and predict the second and third days of each month. We end up with around 500k pixels (samples) per day. The output from the deep learning models (LSTM, ConvLSTM, 3-D CNN, TimeSFormer, SwinTransformer, and Ours) is a probability $Y \in [0, 1]$. In addition, we visualize the predicted maps produced by FWI with the provided spatial resolution 8 km $\times$ 8 km. The output of FWI is clipped to the range $[0, 50]$ [13]. The ignition points of large wildfires on those days are represented as black circles on the map. The first observation is that, regardless of the coarse resolution of FWI, the predicted maps produced by the deep learning models are more reliable. While FWI relies on meteorological observations and models functional relationships, the results show that the modeled functional relationships are insufficient and do not reflect the complexity of the problem of forecasting wildfire. We also find that our proposed model with TE

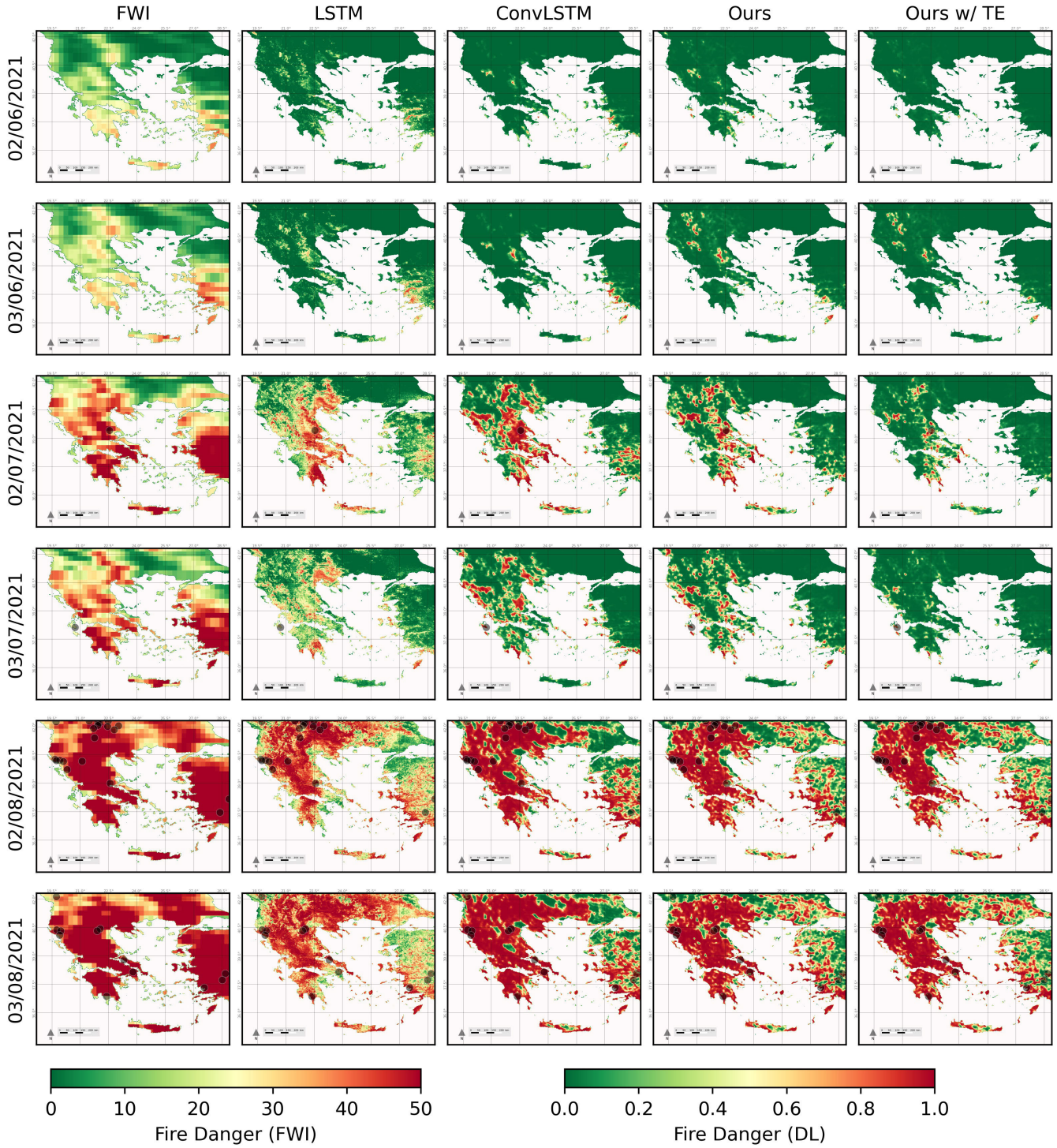


Fig. 8. Qualitative results for six days during the wildfire season in the extreme year 2021. The black circles represent an ignition of a large wildfire on that day. TE denotes the absolute temporal encoding.

discards spots that result in a high FP error rate, while it keeps extreme ones (cf. the results for July 2020 and 2021). Another important observation is that the LSTM model, which does not account for the spatial context, tends to produce heterogeneous predictions where neighboring pixels often have very different wildfire danger probabilities. Consequently, it generates many FPs. In contrast, our proposed model produces more homogeneous and clustered predictions. The respective performance metrics for Figs. 6–9 are provided in Table V. As shown in Table V, all models perform very well in June 2020 and 2021. In these months, there are no positive samples, and our model generates the least number of FPs. In July 2020 and 2021, only 2-D/3-D CNN predicts all positive samples, whereas 2-D/3-D CNN with absolute TE generates the least FPs. The results are mixed in August. While TimeSformer performs very well in August 2020, our approach performs the best in August 2021.

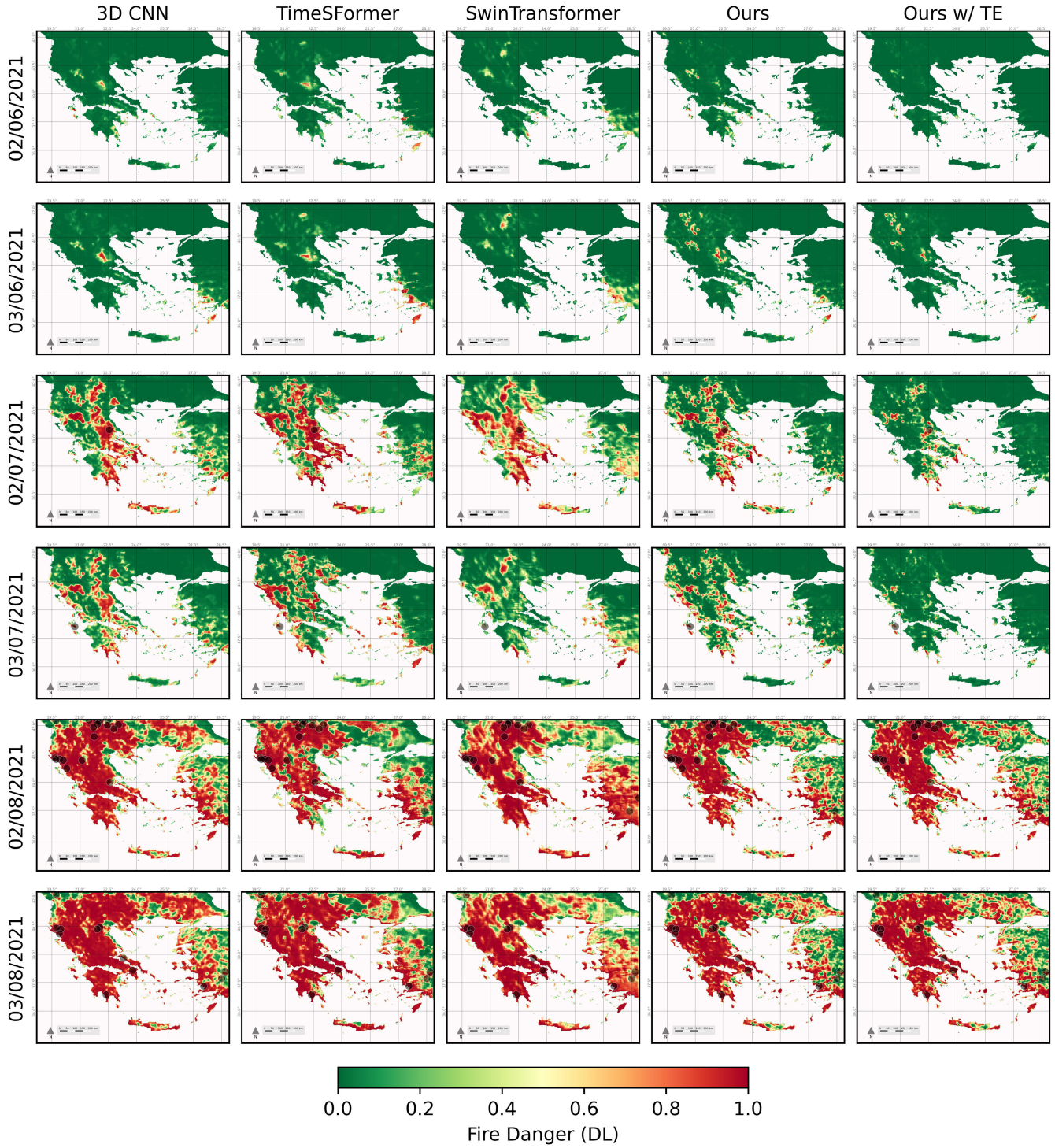


Fig. 9. Qualitative results for six days during the wildfire season in the extreme year 2021. The black circles represent an ignition of a large wildfire on that day. TE denotes the absolute temporal encoding.

VI. Ablation Study

We finally evaluate two additional aspects. In Section VI-A, we analyze at which blocks LOAN layer is best added for the proposed 2-D/3-D CNN. In Section VI-B, we investigate the impact of the absolute TE with respect to the number of negative samples.

A. LOAN Position in the Model

As shown in Fig. 1, the proposed network has three blocks, and we add LOAN to the first and second blocks. We evaluate in Table VI different configurations where we add LOAN only to the first or all three blocks. The results are reported without TE. If we add LOAN only to the first block, the performance increases for the year 2019 but not for the years 2020 and 2021 compared to our model without LOAN (first row). When adding LOAN to the first two blocks, we observe a consistent improvement for all years. For the year 2019, precision, recall, F1-score, and AUROC are improved by +3.96%, +12.39%, +8.75%, and +1.14%, respectively, and +0.36%, +3.62%, +2.02%, and +0.30% for the years

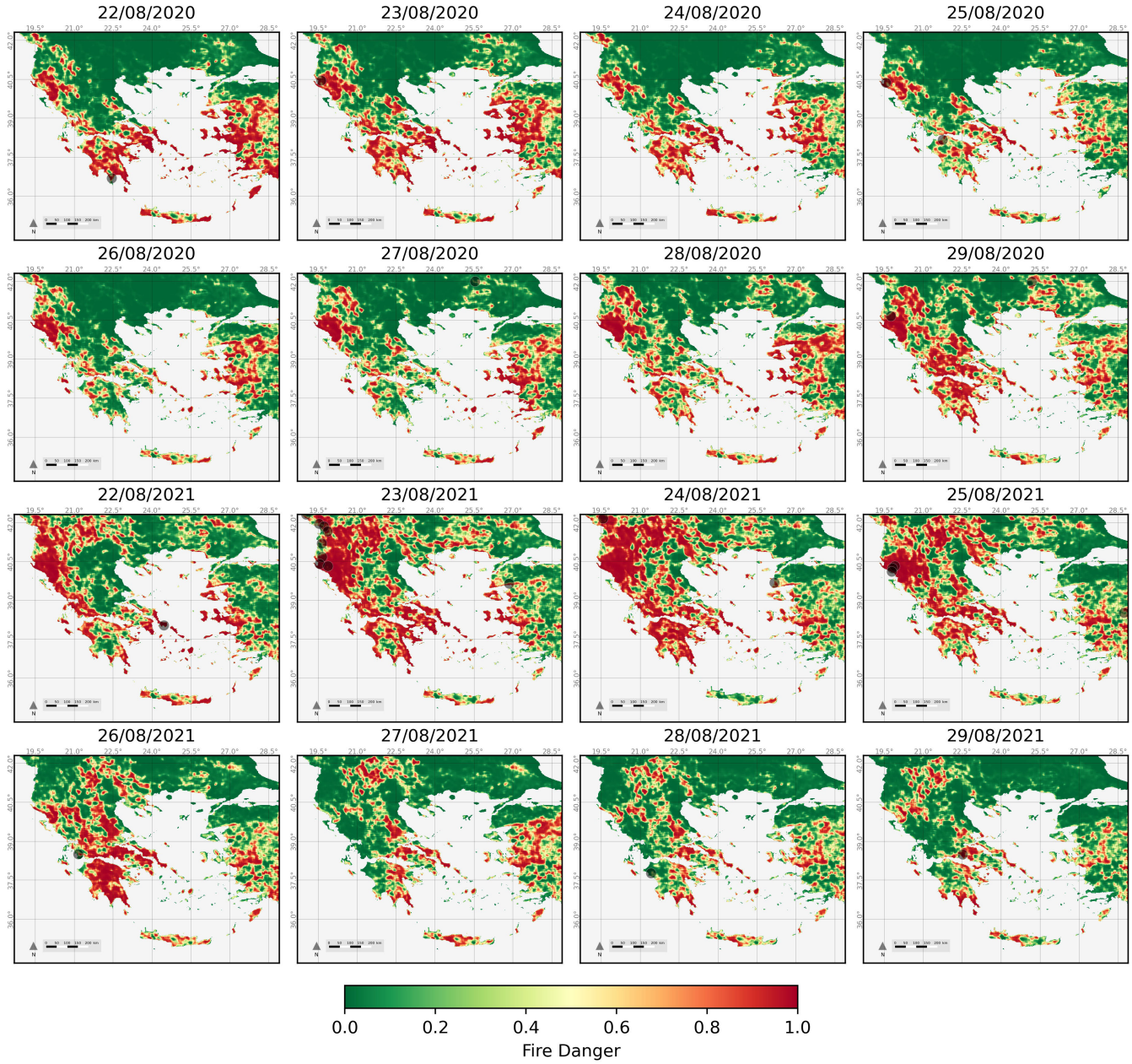


Fig. 10. Predictions for eight consecutive days in August. The upper two rows show the results for the year 2020, and the last two rows show the results for the extreme year 2021. The maps are produced by the proposed 2-D/3-D CNN with absolute TE. The black circles represent an ignition of a large wildfire on that day.

2020 and 2021, respectively. Adding LOAN to all three blocks performs worse than adding LOAN only to the first two blocks. This is due to the decrease in spatial resolution after each block by the pooling layers. In the third block, the spatial resolution is too coarse for a location-specific modulation of the dynamic features.

B. Absolute Temporal Encoding

As we have seen in Tables I and II, absolute TE increases precision at the cost of a lower recall. Depending on the use of wildfire forecasting, recall or precision is more important. The precision also depends on the number of negative samples. In order to show that TE gives consistently a higher precision, we varied the number of negative samples. In this experiment, we test all positive samples in the test set (years 2020 and 2021) and gradually increase the number of negative ones. As shown in Fig. 11, we start with a setup where the number of negative samples is equal to the number of positive samples, i.e., 5635 positive and 5635 negative samples. We then increase the number of negative samples. Since the number of negative samples increases, the precision decreases for all methods. Note that the recall does not change since the number of positive samples remains the same. As already observed in Table II, LSTM has very low precision. 2-D/3-D CNN with LOAN has in all settings a higher precision than ConvLSTM and a much higher recall, as shown in Table II. Transformer models on the other hand have less precision than ConvLSTM

TABLE V

RECALL (%) METRIC PER CLASS FOR FIGS. 6–9. P DENOTES THE POSITIVE CLASS, AND N DENOTES THE NEGATIVE ONE. THE NUMBER OF POSITIVE SAMPLES IS SHOWN BELOW THE DATE. THIS IS EQUIVALENT TO THE AREA (KM^2) THAT WAS BURNED BY LARGE WILDFIRES AT THAT DAY

# Positive samples	02/06/2020 0		03/06/2020 0		02/07/2020 2		03/07/2020 0		02/08/2020 71		03/08/2020 25	
Algorithm	P	N	P	N	P	N	P	N	P	N	P	N
LSTM [13]	-	99.98	-	99.96	00.00	81.41	-	82.37	87.32	48.93	**88.00**	56.23
ConvLSTM [13]	-	**100.00**	-	**100.00**	00.00	89.72	-	92.46	**92.96**	65.09	80.00	72.52
TimeSformer [91]	-	**100.00**	-	**100.00**	00.00	95.60	-	95.41	83.10	**71.02**	80.00	**78.24**
SwinTransformer [92]	-	**100.00**	-	**100.00**	00.00	91.19	-	92.64	88.73	65.57	86.61	75.11
3D CNN	-	**100.00**	-	**100.00**	00.00	88.28	-	87.44	80.28	63.83	80.00	71.06
2D/3D CNN	-	**100.00**	-	99.99	**100.00**	89.51	-	90.33	73.24	66.74	80.00	76.17
2D/3D CNN w/ TE	-	**100.00**	-	**100.00**	00.00	**96.81**	-	**97.92**	78.87	65.68	80.00	74.32
# Positive samples	02/06/2021 0		03/06/2021 0		02/07/2021 36		03/07/2021 22		02/08/2021 679		03/08/2021 1417	
Algorithm	P	N	P	N	P	N	P	N	P	N	P	N
LSTM [13]	-	98.23	-	96.44	**100.00**	70.58	68.18	83.14	82.03	32.54	93.30	23.88
ConvLSTM [13]	-	99.11	-	98.25	**100.00**	75.53	**100.00**	87.42	71.87	35.66	87.01	28.28
TimeSformer [91]	-	98.92	-	97.38	**100.00**	73.46	95.45	81.84	47.72	38.86	97.04	28.99
SwinTransformer [92]	-	99.29	-	98.26	**100.00**	72.14	77.27	91.25	**97.05**	30.13	94.71	26.23
3D CNN	-	99.52	-	98.53	**100.00**	78.39	**100.00**	85.05	83.51	29.80	92.59	22.45
2D/3D CNN	-	**99.38**	-	98.08	**100.00**	84.36	**100.00**	88.97	83.80	**39.17**	99.29	**33.75**
2D/3D CNN w/ TE	-	**99.91**	-	**98.95**	**100.00**	**94.77**	**100.00**	**98.61**	85.13	36.67	**99.36**	31.67

TABLE VI

ABLATION STUDY OF DIFFERENT POSITION CHOICES FOR THE LOAN LAYER. ALL CLASSIFICATION METRICS ARE GIVEN IN PERCENT (%)

Year 2019 (val)						
Block			Precision(↑)	Recall(↑)	F1-score(↑)	AUROC(↑)
1st	2nd	3rd				
×	×	×	75.68	62.23	68.30	93.38
√	×	×	76.90	**75.54**	76.21	94.40
√	√	×	**79.64**	74.62	**77.05**	**94.52**
√	√	√	73.92	70.85	72.35	93.68
Year 2020-2021 (test)						
Block			Precision(↑)	Recall(↑)	F1-score(↑)	AUROC(↑)
1st	2nd	3rd				
×	×	×	83.35	79.98	81.63	97.11
√	×	×	80.90	78.72	79.80	97.22
√	√	×	**83.71**	**83.60**	**83.65**	**97.41**
√	√	√	79.77	81.30	80.52	96.92

but provide an overall better recall, as shown in Table II. While adding TE decreases recall, it improves the precision substantially, and the improvement increases when the number of negative samples increases. While other metrics such as F1-score or AUROC combine precision and recall in a single measure, depending on the application a higher recall or a higher precision might be more important. If precision is more important, TE is very useful. If recall is more important, TE should not be used. We also point out that TE encodes only the day of the year since the dataset has a relatively small spatial extension (10.2°Lon × 8°Lat). In the case of larger datasets at the continental scale, a consideration of the spatial location for the encoding would also become relevant as biogeographical regions occur [73], which are characterized by

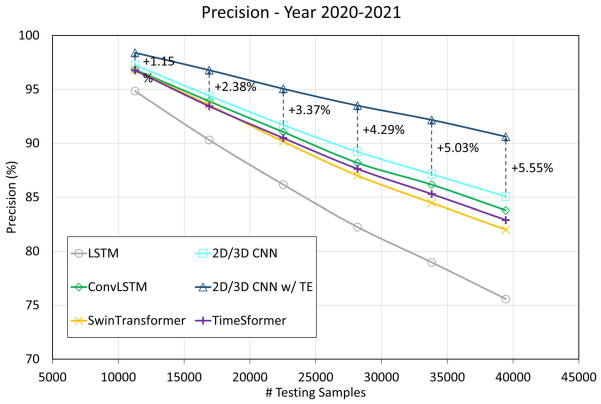


Fig. 11. Impact of the number of testing samples on the precision. TE denotes the absolute temporal encoding.

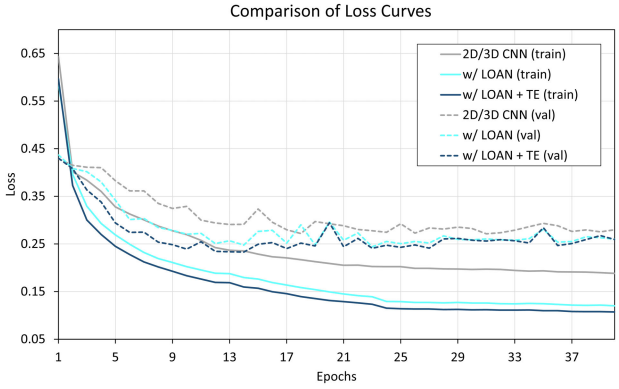


Fig. 12. BCE loss during training for the training (solid) and validation (dashed) sets. LOAN denotes the location-aware adaptive normalization layer, and TE denotes the absolute temporal encoding.

different climate variabilities and anthropogenic drivers over time. Finally, we plot in Fig. 12 the loss (4) curve during training.

VII. Conclusion

In this work, we proposed a new deep learning approach for wildfire danger forecasting. In contrast to previous works, we handle spatial (static) and spatiotemporal (dynamic) variables differently. Our model processes the spatial and spatiotemporal variables in two separated 2-D/3-D CNN branches to learn static and dynamic feature vectors. Moreover, we have introduced the LOAN layer, which modulates the activation maps in the dynamic branch conditionally on their respective static features to address the causal effect of static features on dynamic features. We furthermore integrated an absolute time encoding into the model. By encoding the calendar time, we make the model explicitly aware of the forecasting day. While the time encoding reduces the recall, it substantially increases the precision. We conducted our experiments on the FireCube dataset and demonstrated the effectiveness of our approach compared to several baselines in terms of precision, F1-score, AUROC, and OA. Although our approach demonstrated a substantial improvement compared to previous works for wildfire forecasting, it still has some limitations. Despite the fact that our framework includes domain knowledge through the normalization layer and absolute time encoding, it does not incorporate physical knowledge about the Earth system. Furthermore, the FireCube dataset covers only parts of the Eastern Mediterranean and the years 2009–2021. There is a need for more standardized datasets for wildfire forecasting at a continental scale and longer time periods. Finally, there may be hidden events that are correlated with climate variability and extreme weather conditions. It is an open question how these impact the forecast quality and if additional input variables will be needed to improve the forecast accuracy.

We believe that the proposed approach to dealing with spatial and spatiotemporal variables is also highly relevant for other remote sensing applications.

Acknowledgment

The authors would like to thank Ioannis Prapas and Spyros Kondylatos for providing the FireCube dataset.

References

[1] L. Ren, P. Arkin, T. M. Smith, and S. S. P. Shen, "Global precipitation trends in 1900-2005 from a reconstruction and coupled model simulations," *J. Geophys. Res., Atmos.*, vol. 118, no. 4, pp. 1679–1689, Feb. 2013. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/jgrd.50212

[2] A. M. Lausier and S. Jain, "Overlooked trends in observed global annual precipitation reveal underestimated risks," *Sci. Rep.*, vol. 8, no. 1, pp. 1–7, Nov. 2018.

[3] S. E. Perkins-Kirkpatrick and S. C. Lewis, "Increasing trends in regional heatwaves," *Nature Commun.*, vol. 11, no. 1, pp. 1–8, Jul. 2020.

[4] R. Samuels et al., "Evaluation and projection of extreme precipitation indices in the eastern Mediterranean based on CMIP5 multi-model ensemble," *Int. J. Climatol.*, vol. 38, no. 5, pp. 2280–2297, Apr. 2018.

[5] G. Zittis, P. Hadjinicolaou, M. Klangidou, Y. Proestos, and J. Lelieveld, "A multi-model, multi-scenario, and multi-domain analysis of regional climate projections for the Mediterranean," *Regional Environ. Change*, vol. 19, no. 8, pp. 2621–2635, Dec. 2019.

[6] M. J. Barcikowska, S. B. Kapnick, L. Krishnamurty, S. Russo, A. Cherchi, and C. K. Folland, "Changes in the future summer Mediterranean climate: Contribution of teleconnections and local factors," *Earth Syst. Dyn.*, vol. 11, no. 1, pp. 161–181, Feb. 2020. [Online]. Available: https://esd.copernicus.org/articles/11/161/2020/

[7] A. Hochman et al., "Extreme weather and societal impacts in the eastern Mediterranean," *Earth Syst. Dyn.*, vol. 13, no. 2, pp. 749–777, Apr. 2022. [Online]. Available: https://esd.copernicus.org/articles/13/749/2022/

[8] M. P. Thompson et al., "Risk management and analytics in wildfire response," *Current Forestry Rep.*, vol. 5, no. 4, pp. 226–239, Dec. 2019.

[9] S. C. P. Coogan, F.-N. Robinne, P. Jain, and M. D. Flannigan, "Scientists' warning on wildfire—A Canadian perspective," *Can. J. Forest Res.*, vol. 49, no. 9, pp. 1015–1023, Sep. 2019.

[10] F. Moreira et al., "Wildfire management in Mediterranean-type regions: Paradigm change needed," *Environ. Res. Lett.*, vol. 15, no. 1, Jan. 2020, Art. no. 011001, doi: 10.1088/1748-9326/ab541e.

[11] P. Jain, S. C. P. Coogan, S. G. Subramanian, M. Crowley, S. Taylor, and M. D. Flannigan, "A review of machine learning applications in wildfire science and management," *Environ. Rev.*, vol. 28, no. 4, pp. 478–505, Dec. 2020.

[12] D. Fornacca, G. Ren, and W. Xiao, "Performance of three MODIS fire products (MCD45A1, MCD64A1, MCD14ML), and ESA fire_CCI in a mountainous area of northwest Yunnan, China, characterized by frequent small fires," *Remote Sens.*, vol. 9, no. 11, p. 1131, Nov. 2017.

[13] S. Kondylatos et al., "Wildfire danger prediction and understanding with deep learning," *Geophys. Res. Lett.*, vol. 49, no. 17, Sep. 2022, Art. no. e2022GL099368. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2022GL099368

[14] S. Hantson et al., "The status and challenge of global fire modelling," *Biogeosciences*, vol. 13, no. 11, pp. 3359–3375, 2016.

[15] M. Reichstein et al., "Deep learning and process understanding for data-driven Earth system science," *Nature*, vol. 566, no. 7743, pp. 195–204, Feb. 2019.

[16] I. Prapas et al., "Deep learning methods for daily wildfire danger forecasting," 2021, *arXiv:2111.02736*.

[17] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[18] I. Prapas, S. Kondylatos, and I. Papoutsis, "FireCube: A daily datacube for the modeling and analysis of wildfires in Greece," Zenodo, Tech. Rep., May 2022, doi: 10.5281/zenodo.6475592.

[19] M. C. Iban and A. Sekertekin, "Machine learning based wildfire susceptibility mapping using remotely sensed fire data and GIS: A case study of Adana and Mersin provinces, Turkey," *Ecological Informat.*, vol. 69, Jul. 2022, Art. no. 101647. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574954122000966

[20] B. T. Pham et al., "Performance evaluation of machine learning methods for forest fire modeling and prediction," *Symmetry*, vol. 12, no. 6, p. 1022, Jun. 2020. [Online]. Available: https://www.mdpi.com/2073-8994/12/6/1022

[21] S. Gholami, N. Kodandapani, J. Wang, and J. M. L. Ferres, "Where there's smoke, there's fire: Wildfire risk predictive modeling via historical climate data," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 15309–15315.

[22] C. Shang, M. A. Wulder, N. C. Coops, J. C. White, and T. Hermosilla, "Spatially-explicit prediction of wildfire burn probability using remotely-sensed and ancillary data," *Can. J. Remote Sens.*, vol. 46, no. 3, pp. 313–329, May 2020.

[23] I. Mitsopoulos and G. Mallinis, "A data-driven approach to assess large fire size generation in Greece," *Natural Hazards*, vol. 88, no. 3, pp. 1591–1607, Sep. 2017.

[24] T. Jiang, S. K. Bendre, H. Lyu, and J. Luo, "From static to dynamic prediction: Wildfire risk assessment based on multiple environmental factors," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2021, pp. 4877–4886.

[25] H. V. Le et al., "A new approach of deep neural computing for spatial prediction of wildfire danger at tropical climate areas," *Ecol. Informat.*, vol. 63, Jul. 2021, Art. no. 101300. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574954121000911

[26] G. Zhang, M. Wang, and K. Liu, "Forest fire susceptibility modeling using a convolutional neural network for Yunnan province of China," *Int. J. Disaster Risk Sci.*, vol. 10, no. 3, pp. 386–403, Sep. 2019.

[27] G. Zhang, M. Wang, and K. Liu, "Deep neural networks for global wildfire susceptibility modelling," *Ecol. Indicators*, vol. 127, Aug. 2021, Art. no. 107735. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1470160X21004003

[28] A. Bjånes, R. De La Fuente, and P. Mena, "A deep learning ensemble model for wildfire susceptibility mapping," *Ecol. Informat.*, vol. 65, Nov. 2021, Art. no. 101397. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574954121001886

[29] J. R. Bergado, C. Persello, K. Reinke, and A. Stein, "Predicting wildfire burns from big geodata using deep learning," *Saf. Sci.*, vol. 140, Aug. 2021, Art. no. 105276.

[30] F. Huot, R. L. Hu, N. Goyal, T. Sankar, M. Ihme, and Y.-F. Chen, "Next day wildfire spread: A machine learning dataset to predict wildfire spreading from remote-sensing data," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 4412513.

[31] I. Prapas et al., "Deep learning for global wildfire forecasting," in *Proc. NeurIPS Workshop Tackling Climate Change Mach. Learn.*, 2022, pp. 1–9. [Online]. Available: https://www.climatechange.ai/papers/neurips2022/52, doi: 10.48550/arXiv.2211.00534.

[32] H. Yoon and P. Voulgaris, "Multi-time predictions of wildfire grid map using remote sensing local data," in *Proc. IEEE Int. Conf. Knowl. Graph (ICKG)*, Nov. 2022, pp. 365–372.

[33] Q. Zhao, Y. Ma, S. Lyu, and L. Chen, "Embedded self-distillation in compact multibranch ensemble network for remote sensing scene classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 4506415.

[34] R. Gaetano, D. Ienco, K. Ose, and R. Cresson, "A two-branch CNN architecture for land cover classification of PAN and MS imagery," *Remote Sens.*, vol. 10, no. 11, p. 1746, Nov. 2018. [Online]. Available: https://www.mdpi.com/2072-4292/10/11/1746

[35] Y. Tan, S. Xiong, and P. Yan, "Multi-branch convolutional neural network for built-up area extraction from remote sensing image," *Neurocomputing*, vol. 396, pp. 358–374, Jul. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231219309208

[36] Y. Xu, L. Zhang, B. Du, and F. Zhang, "Spectral–spatial unified networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 10, pp. 5893–5909, Oct. 2018.

[37] Y. Shen, L. Xiao, J. Chen, and D. Pan, "A spectral–spatial domain-specific convolutional deep extreme learning machine for supervised hyperspectral image classification," *IEEE Access*, vol. 7, pp. 132240–132252, 2019.

[38] X. Liu, C. Deng, J. Chanussot, D. Hong, and B. Zhao, "StfNet: A two-stream convolutional neural network for spatiotemporal image fusion," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 9, pp. 6552–6564, Sep. 2019.

[39] C. Gan, X. Yan, Y. Wu, and Z. Zhang, "A two-branch convolution residual network for image compressive sensing," *IEEE Access*, vol. 8, pp. 1705–1714, 2020.

[40] Y. Tang, X. Xie, and Y. Yu, "Hyperspectral classification of two-branch joint networks based on Gaussian pyramid multiscale and wavelet transform," *IEEE Access*, vol. 10, pp. 56876–56887, 2022.

[41] Z. Zhong, J. Li, L. Ma, H. Jiang, and H. Zhao, "Deep residual networks for hyperspectral image classification," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Jul. 2017, pp. 1824–1827.

[42] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[43] Y. Zheng, S. Liu, Q. Du, H. Zhao, X. Tong, and M. Dalponte, "A novel multitemporal deep fusion network (MDFN) for short-term multitemporal HR images classification," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 14, pp. 10691–10704, 2021.

[44] R. Li, S. Zheng, C. Duan, Y. Yang, and X. Wang, "Classification of hyperspectral image based on double-branch dual-attention mechanism network," *Remote Sens.*, vol. 12, no. 3, p. 582, Feb. 2020. [Online]. Available: https://www.mdpi.com/2072-4292/12/3/582

[45] Z. Zhu, Y. Tao, and X. Luo, "HCNNet: A hybrid convolutional neural network for spatiotemporal image fusion," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 2005716.

[46] Z. Deng et al., "A triple-path spectral–spatial network with interleave-attention for hyperspectral image classification," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 15, pp. 5906–5923, 2022.

[47] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[48] Y. Wu and K. He, "Group normalization," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 3–19.

[49] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," 2016, *arXiv:1607.08022*.

[50] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.

[51] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu, "Semantic image synthesis with spatially-adaptive normalization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2332–2341.

[52] X. Wang, K. Yu, C. Dong, and C. Change Loy, "Recovering realistic texture in image super-resolution by deep spatial feature transform," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 606–615.

[53] X. Huang and S. J. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1501–1510.

[54] J. Ling, H. Xue, L. Song, R. Xie, and X. Gu, "Region-aware adaptive instance normalization for image harmonization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 9357–9366.

[55] N. Van Noord and E. Postma, "A learned representation of artist-specific colourisation," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2017, pp. 2907–2915.

[56] P. Zhu, R. Abdal, Y. Qin, and P. Wonka, "SEAN: Image synthesis with semantic region-adaptive normalization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 5103–5112.

[57] Z. Tan et al., "Efficient semantic image synthesis via class-adaptive normalization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 9, pp. 4852–4866, Sep. 2022.

[58] Z. Lv et al., "Learning semantic person image generation by region-adaptive normalization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 10801–10810.

[59] Y. Lyu, P. Chen, J. Sun, X. Wang, J. Dong, and T. Tan, "DRAN: Detailed region-adaptive normalization for conditional image synthesis," 2021, *arXiv:2109.14525*.

[60] K. Jakoel, L. Efraim, and T. R. Shaham, "GANs spatial control via inference-time adaptive normalization," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2022, pp. 31–40.

[61] T. Chen, M. Lucic, N. Houlsby, and S. Gelly. (2019). *On Self Modulation for Generative Adversarial Networks*. [Online]. Available: https://openreview.net/forum?id=Hkl5aoR5tm

[62] V. Sushko, E. Schönfeld, D. Zhang, J. Gall, B. Schiele, and A. Khoreva, "OASIS: Only adversarial supervision for semantic image synthesis," *Int. J. Comput. Vis.*, vol. 130, no. 12, pp. 2903–2923, Dec. 2022.

[63] S. Li, M.-M. Cheng, and J. Gall, "Dual pyramid generative adversarial networks for semantic image synthesis," in *Proc. 33rd Brit. Mach. Vis. Conf. (BMVC)*, London, U.K.: BMVA Press, 2022, pp. 1–13. [Online]. Available: https://bmvc2022.mpi-inf.mpg.de/0285.pdf, doi: 10.48550/arXiv.2210.04085.

[64] J. Marín and S. Escalera, "SSSGAN: Satellite style and structure generative adversarial networks," *Remote Sens.*, vol. 13, no. 19, p. 3984, Oct. 2021. [Online]. Available: https://www.mdpi.com/2072-4292/13/19/3984

[65] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 1–9.

[66] M. Rußwurm and M. Körner, "Multi-temporal land cover classification with sequential recurrent encoders," *ISPRS Int. J. Geo-Inf.*, vol. 7, no. 4, p. 129, Mar. 2018. [Online]. Available: https://www.mdpi.com/2220-9964/7/4/129

[67] C. Pelletier, G. Webb, and F. Petitjean, "Temporal convolutional neural network for the classification of satellite image time series," *Remote Sens.*, vol. 11, no. 5, p. 523, Mar. 2019. [Online]. Available: https://www.mdpi.com/2072-4292/11/5/523

[68] W. R. Moskolaï, W. Abdou, A. Dipanda, and Kolyang, "Application of deep learning architectures for satellite image time series prediction: A review," *Remote Sens.*, vol. 13, no. 23, p. 4822, Nov. 2021. [Online]. Available: https://www.mdpi.com/2072-4292/13/23/4822

[69] V. S. F. Garnot, L. Landrieu, S. Giordano, and N. Chehata, "Satellite image time series classification with pixel-set encoders and temporal self-attention," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 12322–12331.

[70] V. S. F. Garnot and L. Landrieu, "Lightweight temporal self-attention for classifying satellite images time series," in *Advanced Analytics and Learning on Temporal Data*, V. Lemaire, S. Malinowski, A. Bagnall, T. Guyet, R. Tavenard, and G. Ifrim, Eds. Cham, Switzerland: Springer, 2020, pp. 171–181.

[71] V. S. Fare Garnot and L. Landrieu, "Panoptic segmentation of satellite image time series with convolutional temporal attention networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 4852–4861.

[72] L. Drees, I. Weber, M. Rußwurm, and R. Roscher, "Time dependent image generation of plants from incomplete sequences with CNN-transformer," in *Proc. DAGM German Conf. Pattern Recognit.* Springer, 2022, pp. 495–510.

[73] J. Nyborg, C. Pelletier, S. Lefèvre, and I. Assent, "TimeMatch: Unsupervised cross-region adaptation by temporal shift estimation," *ISPRS J. Photogramm. Remote Sens.*, vol. 188, pp. 301–313, Jun. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924271622001216

[74] J. Nyborg, C. Pelletier, and I. Assent, "Generalized classification of satellite image time series with thermal positional encoding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2022, pp. 1391–1401.

[75] J. Muñoz-Sabater et al., "ERA5-land: A state-of-the-art global reanalysis dataset for land applications," *Earth Syst. Sci. Data*, vol. 13, no. 9, pp. 4349–4383, Sep. 2021. [Online]. Available: https://essd.copernicus.org/articles/13/4349/2021/

[76] K. Didan, "MOD13A2 MODIS/Terra vegetation indices 16-day L3 global 1 km SIN grid v006 [data set]. NASA EOSDIS land processes DAAC," NASA EOSDIS Land Processes DAAC, Tech. Rep., 2015, doi: 10.5067/MODIS/MOD13A2.006.

[77] Z. Wan, S. Hook, and G. Hulley, "MOD11A2 MODIS/Terra land surface temperature/emissivity 8-day L3 global 1 km SIN grid v006," NASA EOSDIS Land Processes DAAC, USA, Tech. Rep., 2015, doi: 10.5067/MODIS/MOD11A1.006.

[78] A. Bashfield and A. Keim, "Continent-wide DEM creation for the European Union," in *Proc. 34th Int. Symp. Remote Sens. Environ. GEOSS Era, Towards Oper. Environ. Monit.*, Sydney, NSW, Australia, 2011, pp. 10–15.

[79] A. J. Tatem, "WorldPop, open data for spatial demography," *Sci. Data*, vol. 4, no. 1, pp. 1–4, Jan. 2017.

[80] J. San-Miguel-Ayanz, E. Schulte, G. Schmuck, and A. Camia, "The European forest fire information system in the context of environmental policies of the European Union," *Forest Policy Econ.*, vol. 29, pp. 19–25, Apr. 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S138993411200127X

[81] L. Giglio, W. Schroeder, and C. O. Justice, "The collection 6 MODIS active fire detection algorithm and fire products," *Remote Sens. Environ.*, vol. 178, pp. 31–41, Jun. 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0034425716300827

[82] G. Büttner, "CORINE land cover and land cover change products," in *Land Use and Land Cover Mapping in Europe*. Dordrecht, The Netherlands: Springer, 2014, pp. 55–74, doi: 10.1007/978-94-007-7969-3_5.

[83] D. Steinfeld. (2022). *Calculation of Indices for Forest Fire Risk Assessment in Weather and Climate Data*. [Online]. Available: https://github.com/steidani/FireDanger

[84] T. M. Giannaros et al., "Meteorological analysis of the 2021 extreme wildfires in Greece: Lessons learned and implications for early warning of the potential for pyroconvection," *Atmosphere*, vol. 13, no. 3, p. 475, Mar. 2022. [Online]. Available: https://www.mdpi.com/2073-4433/13/3/475

[85] I. Prapas, S. Kondylatos, and I. Papoutsis, "Training data for submitted paper 'wildfire danger prediction and understanding with deep learning,"' Zenodo, Tech. Rep., May 2022, doi: 10.5281/zenodo.6528394.

[86] C. Cammalleri, J. V. Vogt, B. Bisselink, and A. de Roo, "Comparing soil moisture anomalies from multiple independent sources over different regions across the globe," *Hydrol. Earth Syst. Sci.*, vol. 21, no. 12, pp. 6329–6343, Dec. 2017.

[87] M. Lin, Q. Chen, and S. Yan, "Network in network," 2013, *arXiv:1312.4400*.

[88] Y. Martín, M. Zúñiga-Antón, and M. R. Mimbrero, "Modelling temporal variation of fire-occurrence towards the dynamic prediction of human wildfire ignition danger in northeast Spain," *Geomatics, Natural Hazards Risk*, vol. 10, no. 1, pp. 385–411, Jan. 2019, doi: 10.1080/19475705.2018.1526219.

[89] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32. Red Hook, NY, USA: Curran Associates, 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

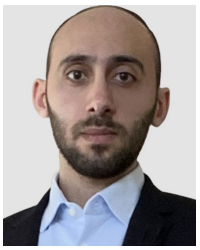
[90] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, pp. 1–15, Dec. 2015.

[91] G. Bertasius, H. Wang, and L. Torresani, "Is space-time attention all you need for video understanding?" in *Proc. Int. Conf. Mach. Learn. (ICML)*, Jul. 2021, p. 4.

[92] Z. Liu et al., "Video swin transformer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 3202–3211.

[93] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001. [Online]. Available: https://link.springer.com/article/10.1023/A:1010933404324

[94] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. New York, NY, USA: Association for Computing Machinery, Aug. 2016, pp. 785–794, doi: 10.1145/2939672.2939785.



Mohamad Hakam Shams Eddin (Member, IEEE) received the Dipl.Ing. degree in topographic engineering from the University of Aleppo, Aleppo, Syria, in 2015, and the M.Sc. degree in geomatics engineering from the University of Stuttgart, Stuttgart, Germany, in 2019. He is currently pursuing the Ph.D. degree with the University of Bonn, Bonn, Germany.

His research interests include deep learning, remote sensing, and anomaly detection.

Ribana Roscher (Member, IEEE) received the Dipl.Ing. and Ph.D. degrees in geodesy from the University of Bonn, Bonn, Germany, in 2008 and 2012, respectively.

Until 2022, she was a Junior Professor of remote sensing with the University of Bonn. She was a Post-Doctoral Researcher with the University of Bonn, the Julius Kuehn Institute, Siebeldingen, Germany, Freie Universität Berlin, Berlin, Germany, and Humboldt Innovation, Berlin. In 2015, she was a Visiting Researcher with the Fields Institute, Toronto, ON, Canada. Since 2022, she has been a Professor of data science for crop systems with the University of Bonn. She currently leads the data science research area at Institute of Bio- and Geosciences (IBG)-2, Forschungszentrum Jülich, Jülich, Germany.

Juergen Gall (Member, IEEE) received the B.Sc. degree in mathematics from the University of Wales, Swansea, U.K., in 2004, the M.Sc. degree in mathematics from the University of Mannheim, Mannheim, Germany, in 2005, and the Ph.D. degree in computer science from Saarland University, Saarbrücken, Germany, and the Max-Planck-Institut für Informatik, Saarbrücken, in 2009.

He was a Post-Doctoral Researcher with the Computer Vision Laboratory, ETH Zürich, Zürich, Switzerland, from 2009 to 2012, and a Senior Research Scientist with the Max Planck Institute for Intelligent Systems, Tübingen, Germany, from 2012 to 2013. Since 2013, he has been a Professor with the University of Bonn, Bonn, Germany, where he is currently the Head of the Computer Vision Group. He is also a member of the Lamarr Institute for Machine Learning and Artificial Intelligence, Dortmund, Germany.