# Improved Wildfire Spread Prediction with Time-Series Data and the WSTS+ Benchmark

Supplementary Material

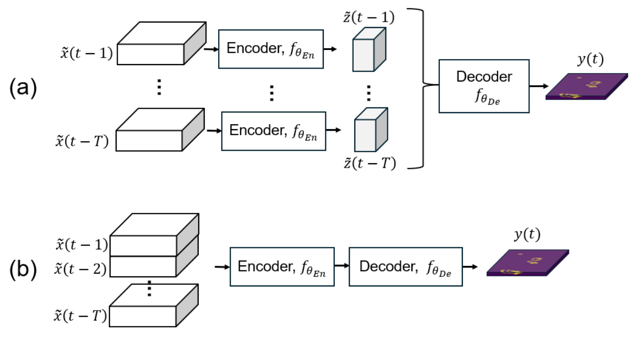

Figure 7. Illustration of (a) feature-level fusion, and (b) data-level fusion as we define it here. Further description is provided in the main text, and mathematical notation is described in Sec. 2

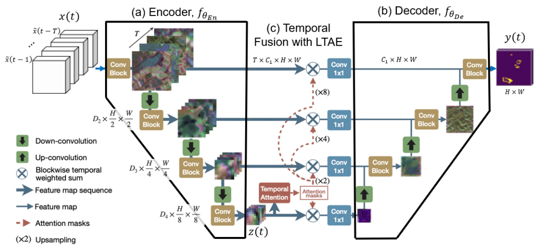
$x(t)$
$\tilde{x}(t-T)$
$\tilde{x}(t-1)$
(a) Encoder, $f_{\theta_{En}}$
$T$
Conv Block
$D_2 \times \frac{H}{2} \times \frac{W}{2}$
$D_3 \times \frac{H}{4} \times \frac{W}{4}$
$D_4 \times \frac{H}{8} \times \frac{W}{8}$
$z(t)$
(c) Temporal Fusion with LTAE
$T \times C_1 \times H \times W$
(×8)
(×4)
(×2)
Temporal Attention
Attention masks
Conv 1x1
(b) Decoder, $f_{\theta_{De}}$
$C_1 \times H \times W$
$y(t)$
$H \times W$
Down-convolution
Up-convolution
Blockwise temporal weighted sum
Feature map sequence
Feature map
Attention masks
(×2) Upsampling

Figure 8. Illustration of the U-Net and UTAE models, adapted from [16] to our wildfire problem: see description in main text.

## 8. Experimental Details

### 8.1. UTAE

The UTAE [16], originally developed for satellite imagery is essentially a U-Net that has been modified to process a time-series of imagery, and was recently found successful for modeling wildfire spread [17]. We propose a novel modification of the time-series positional encodings and therefore discuss the technical details of the UTAE here. The UTAE encodes each entry in the time-series independently using a shared encoder shown in Fig. 8(a), and then fuses the resulting embeddings from each day using a Lightweight Temporal Self-Attention (LTAE) block [15], shown in Fig. 8(c). Given a $T$-length time-series of input, the encoder produces a series of $T$ embeddings $z(t) = \{\tilde{z}(t-i)\}_{i=1}^{T}$ where $\tilde{z}(t) \in \mathbb{R}^{D_4 \times \frac{H}{8} \times \frac{W}{8}}$ at the output of the last layer of the encoder. Then the LTAE computes an attention mask, $a \in \mathbb{R}^{T \times \frac{H}{8} \times \frac{W}{8}}$, which is utilized to combine the $T$ embeddings. Before computing the temporal attention, LTAE adds a sinusoidal positional embedding, $p(\bar{t})$ to each input embedding, where $\bar{t} \in [1, 365]$ is an integer representing the day of the year, and $p(\bar{t})$ maps $\bar{t}$ to a unique sinusoidal representation. This positional embedding is motivated by the original application of UTAE to agricultural segmentation, where the appropriate segmentation depends heavily upon the day of the year. Once the attention mask is computed, it is then upsampled, and applied to the encoder embeddings output at each resolution to collapse the temporal dimension. After all temporal dimensions are collapsed, a conventional U-Net-like decoder is applied to the collapsed embeddings, as shown in Fig. 8(b).

### 8.2. SwinUnet

SwinUnet [7] is a pure transformer-based Unet-shaped model that was first proposed for medical imagery segmentation. The model replaces the convolution blocks of the Unet with Swin Transformer blocks [33], including them throughout the encoder, bottleneck, and decoder. They also rely on patch merging and patch expansion layers in the encoder and decoder, respectively, to downsample the input features and then upsample the extracted features and produce the segmentation mask. Finally, they preserve skip connections to concatenate shallow and deep features. The SwinUnet outperformed the Unet [42], ViT [8], Att-Unet [37], and TransUnet [8] on two medical benchmark datasets, and was shown to outperform the Unet on wildfire prediction [54]. Its state-of-the-art performance, ability to learn both global and long-range dependencies, and use of the more efficient Swin blocks make it a good candidate for our task. Since the model was developed for RGB images, we modify the `in_chans` parameter to take in the number of channels of our multi-modal inputs (Veg: 7, Multi: 33, All: 40) instead of 3.

### 8.3. SegFormer

SegFormer [52] is a recent, efficient, Transformer-based model developed for semantic segmentation. Whereas Swin focuses on improving the encoder using Transformers, SegFormer improves both the encoder, using a hierarchical Transformer that does not require positional encodings, and the decoder, using a lightweight MLP that makes the model efficient. SegFormer achieved excellent performance on the ADE20K and Cityscapes semantic segmentation benchmarks, surpassing state-of-the-art models like DeeplabV3+ and SETR. Several papers used SegFormer in wildfire-related tasks, including [6, 14, 38, 45, 47] and found it to outperform CNNs on burned area delineation. Since the model does not use positional encodings, it can be finetuned/tested on any resolution. Therefore, we finetune it on our dataset without padding our input images. To control for model complexity, we use the SegFormer-B2 model as it uses the closest number of model parameters (27.5M) to that of the SwinUnet (27.2M).

### 8.4. Model pre-training

To evaluate the effect of pre-training on the SwinUnet model, we load the `swin-tiny-patch4-window7-224` weights from HuggingFace onto each of our Swin blocks. These weights correspond to a Swin Transformer trained on ImageNet at 224x224 resolution. We zero-pad our input images (128x128) to match the expected input dimensions and benefit from the pre-trained weights. As for the Unet models, we follow [17] and use the `segmentation_models_pytorch` implementation, and set `encoder_weights` to `imagenet`, which loads a model with ImageNet pre-trained weights. The UTAE pre-training uses the PASTIS weights, released with the original paper [16]. We use the 4th fold checkpoint, as it was the one with the highest performance. Finally, we load the `mit-b2` weights from HuggingFace to use the SegFormer-B2 encoder fine-tuned on Imagenet-1k.

### 8.5. Training details

To train our models, we adopt the implementations shared by [17], which can be found in this GitHub repository. The implementation relies on PyTorch Lightning for model creation, training, and testing and Weights & Biases for model logging and metric visualization. All our models use a fixed batch size of 64, the AdamW optimizer, and a fixed optimized learning rate, as described in Sec. 5. Also, following [17], we train our models for 10,000 iterations. Increasing the number of iterations to 15,000 and 20,000 did not yield any notable increases in performance. For all runs in Tab. 2, we report the mean test AP averaged over the 12 folds, and the standard deviation. During the hyperparameter search, we only use a single data fold (id = 2), train for 50 epochs, and pick the combination that yields the highest validation AP.

## 9. Additional Analyses

### 9.1. Deployment Characteristics

In Tab. 7, we provide key deployment characteristics for each model used in our benchmark. Parameter count refers to the total number of trainable parameters in Millions. We compute inference time by doing 10 warmup runs to stabilize the GPUs, then 100 inference runs, and report the mean in milliseconds $\pm$ standard deviation. GPU Memory Usage tracks peak GPU memory consumption in MB. We use `torchprofile.profile_macs` to estimate total FLOPs (floating point operations). Training Time Estimation (in hours) simulates 20 forward and backward passes, then times a full training step, and extrapolates it to the full training regime (100 epochs using 1000 steps). Model size refers to the model weights file size in MB.

The results in Tab. 7 show that the Res18-UNet offers the best balance, being a small (14M parameters), fast (2.5 ms inference), low-memory (55 Mb) model, resulting in excellent test AP (0.455). The Res50-UNet offers slightly higher accuracy (0.457) at the cost of double the amount of parameters, inference time, training time, and size. Both UNet-based models are relatively cheap computationally (1.8 and 3.1G FLOPs, respectively) and use a manageable amount of GPU Memory (70 MB and 375 MB, respectively), making them easier to deploy on machines with resource-constrained GPUs.

The transformer-based models SwinUnet and SegFormer are slower (9-13 ms for inference and 1.8-2.0 h for training), and computationally heavier (3.7-6.1G FLOPs and 526-865 MB of GPU memory usage), yet without any AP gains. Finally, while the UTAE is compact in storage (4 MB only), it is very computationally expensive (10.6G FLOPs and 997 MB GPU memory usage) despite having the smallest number of parameters (1M). This is likely due to the expensive operations inside the temporal attention block (LTAE). Compared to the other models, it is rather slow in inference (9.5 ms) yet relatively fast in training (1 h). As such, it seems that the Res18-UNet is most optimal if deployment efficiency is the priority. Although SwinUnet and SegFormer don't outperform the UNets in this setup, they may generalize better in other domains. UTAE offers a mix of fast training and lightweight model size with heavy computation and GPU memory usage.

### 9.2. Why do simpler models outperform more complex ones?

In Tab. 2, we found that the simpler convolution-based Res18-Unet outperformed its more complex, Transformer-based counterparts (SwinUnet and SegFormer). We hypothesized that this may be due to the realistic 12-fold leave-one-year-out (LOYO) cross-validation scheme adopted by the WSTS benchmark, penalizing the complex models for overfitting to temporal shifts. To test this hypothesis, we retrain our Res18-Unet, SwinUnet, and SegFormer models using a random 4-fold cross-validation scheme across fire events (i.e., each fire event, and all associated training instances only appear in one fold), and we report the results in Tab. 8. We find that performance (in AP) increases significantly when using event-based cross-validation ("random" in Tab. 2) instead of LOYO validation, as expected. However, the rank-order of the models remains unchanged, with Res18-Unet still outperforming the transformer-based

Table 7. Model deployment characteristics and performance trade-offs

| Model | Params (M) | FLOPs (G) | Inference (ms) | GPU Mem (MB) | Size (MB) | Training (h) | Test AP |
|---|---|---|---|---|---|---|---|
| Res18-UNet | 14.3 | 1.8 | 2.5±0.0 | 70 | 55 | 0.4 | 0.455 |
| Res50-UNet | 32.6 | 3.1 | 5.1±0.1 | 375 | 125 | 1.1 | 0.457 |
| SwinUnet | 27.2 | 6.1 | 8.9±0.0 | 526 | 106 | 1.8 | 0.432 |
| SegFormer | 27.5 | 3.7 | 12.7±0.8 | 865 | 105 | 2.0 | 0.448 |
| UTAE | 1.1 | 10.6 | 9.5±1.0 | 997 | 4 | 1.0 | 0.452 |

SwinUnet and SegFormer, and by a similar margin. These results suggest that the cross-validation scheme is not responsible for the lower performance of more complex (e.g., transformer-based) models. Furthermore, we find no evidence of overfitting among the transformer-based models in either LOYO or event-based "Random" cross-validation. Therefore, it does not appear that overfitting is the cause of their inferiority compared to the ResUnet, and it (tentatively) appears that simpler convolutional models, such as the Res18-Unet may be generally superior for this task, although this is only a hypothesis and further study is needed to conclude.

Table 8. Mean test AP $\pm$ standard deviation using vegetation features only (Veg), vegetation, land cover, topography, and weather (Multi), and All features, when training with 1 input day using the original leave-one-year-out (LOYO) 12-fold cross-validation scheme versus a random 4-fold cross-validation scheme.

| Model | X-val | Veg | Multi | All | Params |
|---|---|---|---|---|---|
| Res18-Unet | LOYO | $0.455 \pm 0.090$ | $0.468 \pm 0.087$ | $0.460 \pm 0.084$ | 14.3M |
| | Random | $0.527 \pm 0.056$ | $0.540 \pm 0.058$ | $0.542 \pm 0.066$ | |
| SwinUnet | LOYO | $0.432 \pm 0.088$ | $0.437 \pm 0.082$ | $0.424 \pm 0.090$ | 27.2M |
| | Random | $0.493 \pm 0.127$ | $0.529 \pm 0.113$ | $0.511 \pm 0.090$ | |
| SegFormer | LOYO | $0.433 \pm 0.080$ | $0.436 \pm 0.083$ | $0.423 \pm 0.087$ | 27.5M |
| | Random | $0.503 \pm 0.053$ | $0.515 \pm 0.046$ | $0.511 \pm 0.069$ | |

## 9.3. Statistical Significance

To determine if the performance increase of our models was statistically significant with respect to the variance introduced by randomness in the training, validation, and testing data sets, we conducted a Wilcoxon signed-rank test on the twelve accuracy scores obtained from the 12 cross-validation folds of our best Res18-Unet, compared to the original Res18-Unet from [17]. Given the distribution of the models' performance values (given by mAP) is highly non-Gaussian, we use the Wilcoxon test which does not assume a Gaussian distribution of outcomes, and tests whether the median differences are zero. The results indicate that our Res18-Unet model's mAP was statistically significantly higher than the previous model (W = 7.0, p = 0.0093). The relatively small W score indicates that the ranking differences between the models are consistently in favor of our model. Moreover, the p-value is below the 0.05 threshold, which supports the rejection of the null hypothesis. Fig. 9 shows the per-fold accuracy improvement, with our model consistently achieving better average precision in 10 out of the 12 folds, further confirming the validity of our improvements.

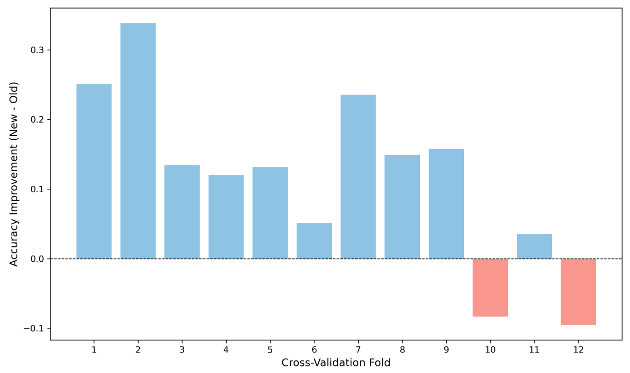

Figure 9. Per-Fold Accuracy Improvement (Res18-Unet (ours) vs. Original)

## 9.4. Failure Case Analysis

We present in Fig. 11, Fig. 13, Fig. 15, Fig. 17 the 10 best predictions (as determined by AP) made by our best model, the Res18-Unet, for each testing year. On the other hand, Fig. 12, Fig. 14, Fig. 16, Fig. 18 show the 10 worst predictions.

Looking at the predictions, we observe that the model consistently fails (AP around 0.001) when the fire to predict is either extremely small (a few pixels), nonexistent in the previous day (a new, ignited fire), or considerably displaced relative to the current day (a spotting event, which is known to be a modeling challenging for the fire spread community [50]). The model collapse is characterized by prediction maps being dominated almost entirely by false positives (shown in red). This is somewhat expected, as the model is not trained to predict the start of new fires. Moreover, we note that this behavior is consistent across all four test years, suggesting that these specific failures are due to the nature of the fire events, and not to data shift between years.

Conversely, the model does best when the fires are larger, have a more consolidated structure, and grow/shrink around

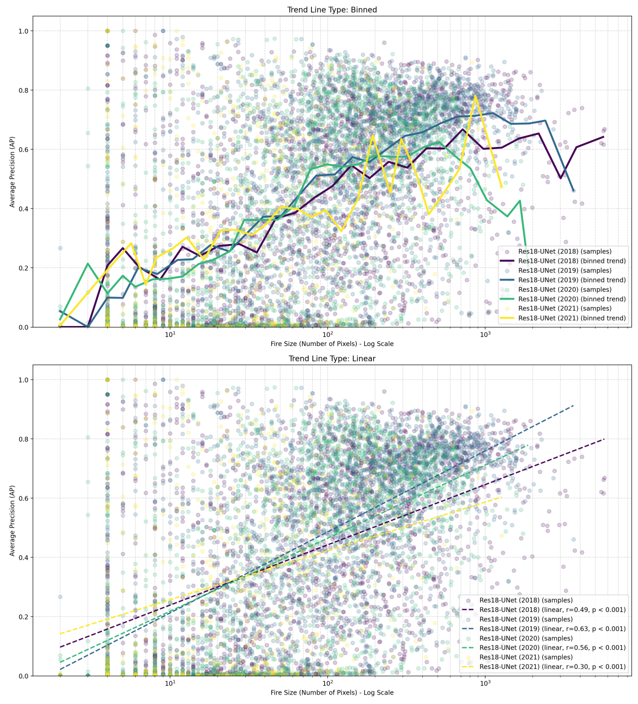

Figure 10. Scatters of test AP against fire size. Top: we overlay a line through binned averages. Bottom: we overlay a regression line.

the same vicinity. It learned to consistently predict the bulk of the fire spread correctly (shown in green), with a small amount of false positives (shown in red) or false negatives (shown in blue), mostly around the edges of the fire. Still, the model achieves high performance on these samples (AP $> 0.9$; F1 $> 0.8$), showing that it learned to accurately predict larger fires.

## 9.5. Fire Size Impact

To further investigate the impact of fire size on the difficulty of prediction, we visualize in Fig. 10 scatters of AP against fire mask size. We compute fire size as the total number of positive pixels in each ground truth mask, and compute the AP, per fire event, achieved by our best Res18-Unet, when tested on each year. Each dot represents a test instance, and in the top plot, we overlay a regression line between the fire size (on log scale) and the test AP. For the bottom plot, we first divide fire sizes into 30 bins and compute the mean AP for each bin, then plot a smoothed curve through the mean AP values.

Looking at the plots, we observe that fire sizes vary from very small ($< 10$ pixels) to extremely large ($> 1000$ pixels). We also note that the performance (measured by AP) is highly variable across scales. However, using smoothing allows us to confirm that our observations in Sec. 9.4 are not anecdotal but rather systematic: *there exists a positive correlation between fire size and model performance across years, with larger fires being generally easier to predict.*

In the top plot, we notice that the AP increases steadily with fire size up to a few hundred pixels, then plateaus. We also observe some differences between the years. For example, when the model is tested on 2019, it achieves the highest AP across medium-to-large fires. We also observe that the AP of the model tested on 2020 initially rises but reaches the lowest value of all models. The model tested on 2021 shows some fluctuation, with apparent instability in performance for large fires. Finally, the 2018 model seems to follow the smoothest curve.

In the bottom plot, we notice that the correlation strength varies throughout years, with 2019 (shown in blue) showing the strongest correlation (r = 0.63), and 2021 (shown in yellow) having the weakest correlation (r = 0.30). This means that the model is usually able to predict larger fires better, but this is inconsistent across years.

## 10. WSTS+ Details

### 10.1. Collection Details

To ensure our added wildfire events are most similar to the original ones, we follow the exact same collection procedure in [17]. Namely, we rely on the Google Earth Engine script found in this repository, to only collect wildfires that are larger than 10 km$^2$, and we use the GlobFire dataset [2] to identify wildfire events in the United States for 2016 and 2017. However, given GlobFire's temporal availability ends at 2021, we use the MTBS Burned Areas Boundaries Dataset [36] to identify wildfires in 2022 and 2023.

The main differences between the datasets used for fire event *identification* are that GlobFire relies on MODIS [19] as a data source, which has a resolution of 500 meters, while MTBS uses Landsat imagery, which has 30 meter resolution. Furthermore, GlobFire returns burned area maps with start and end dates, while MTBS returns fire perimeters with start dates only. Regardless, we only use the centroid coordinate for both area maps and perimeters to download the fire masks. To account for the lack of fire end dates in MTBS, we collect 30 days of samples after the start date, with an additionnal buffer of 4 days before and after the fire events, similar to [17]. We visualize the distribution of fire events in WSTS+ in Fig. 19.

### 10.2. Quality Assurance

The new data were processed in the exact same way as the original WSTS data. To verify that it was done properly, we first replicated the downloading and processing of the original WSTS data (2018-2021), and measured the differences between our reproduction and the original data. We found that both are quantitatively similar. Specifically, we computed the mean pixel values of each data band for two

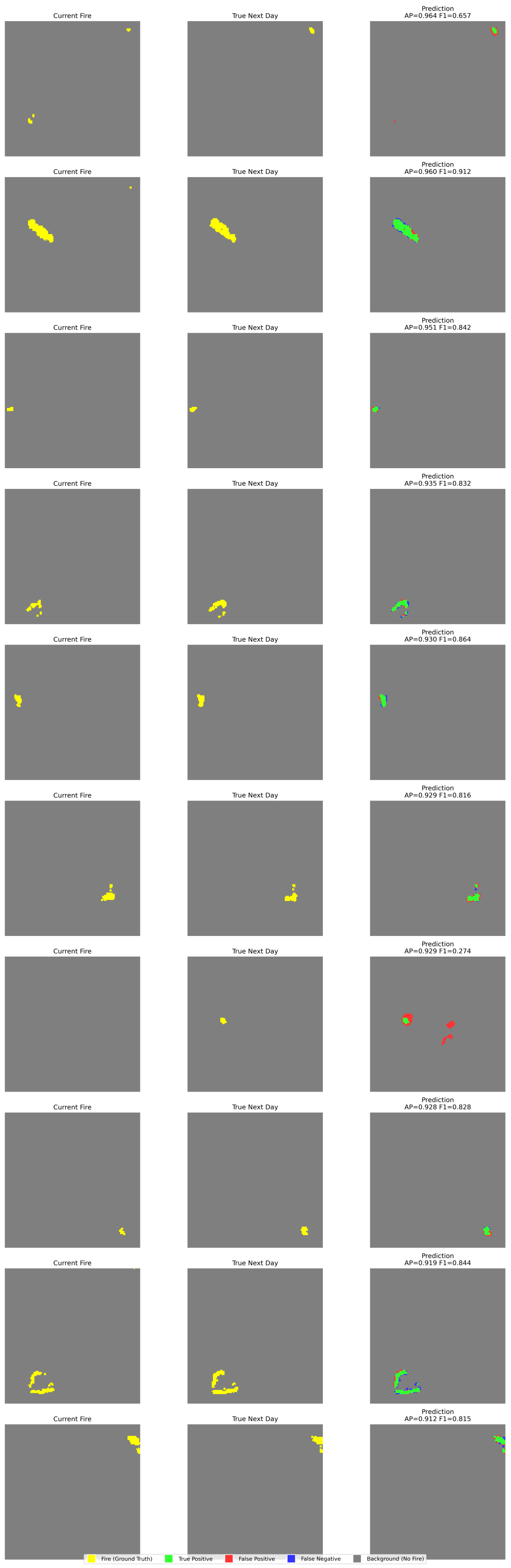

Figure 11. 10 best predictions made by the Res18-Unet on the 2018 test year.

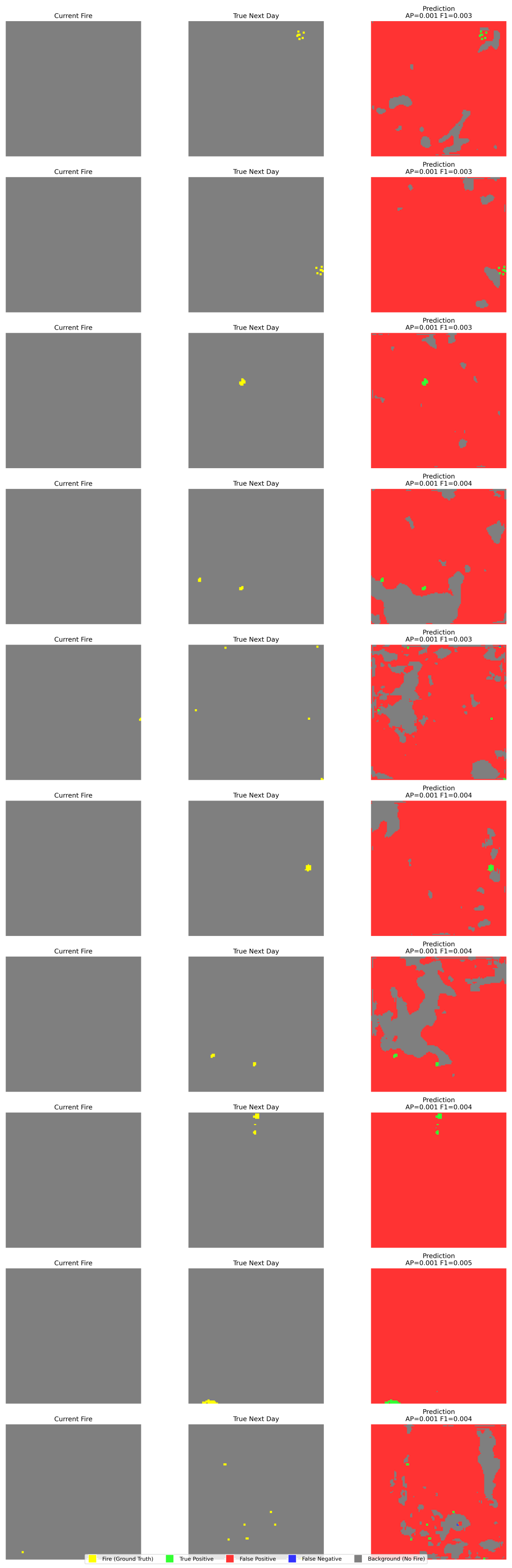

Figure 12. 10 worst predictions made by the Res18-Unet on the 2018 test year.

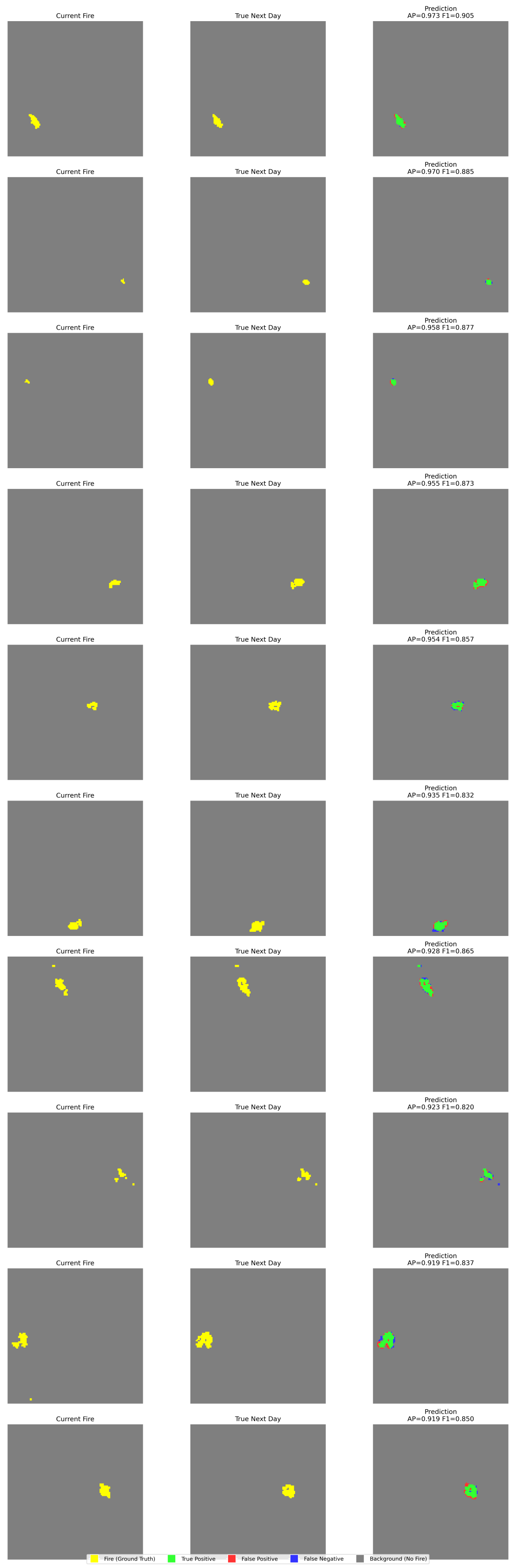

Figure 13. 10 best predictions made by the Res18-Unet on the 2019 test year.

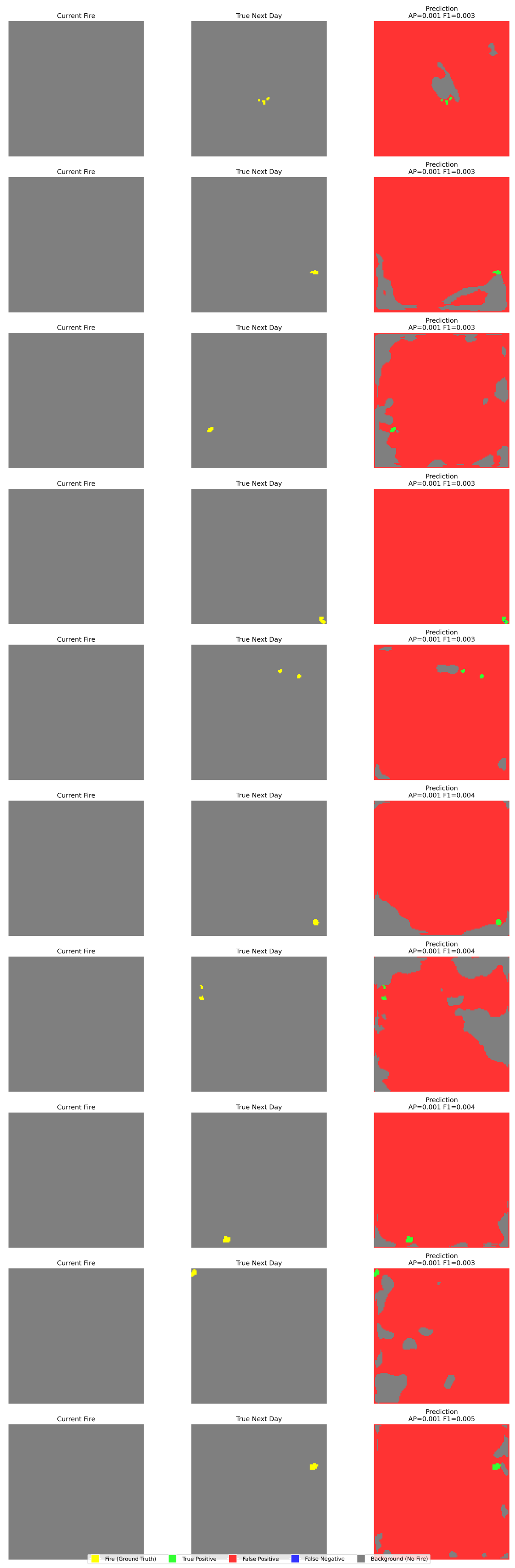

Figure 14. 10 worst predictions made by the Res18-Unet on the 2019 test year.

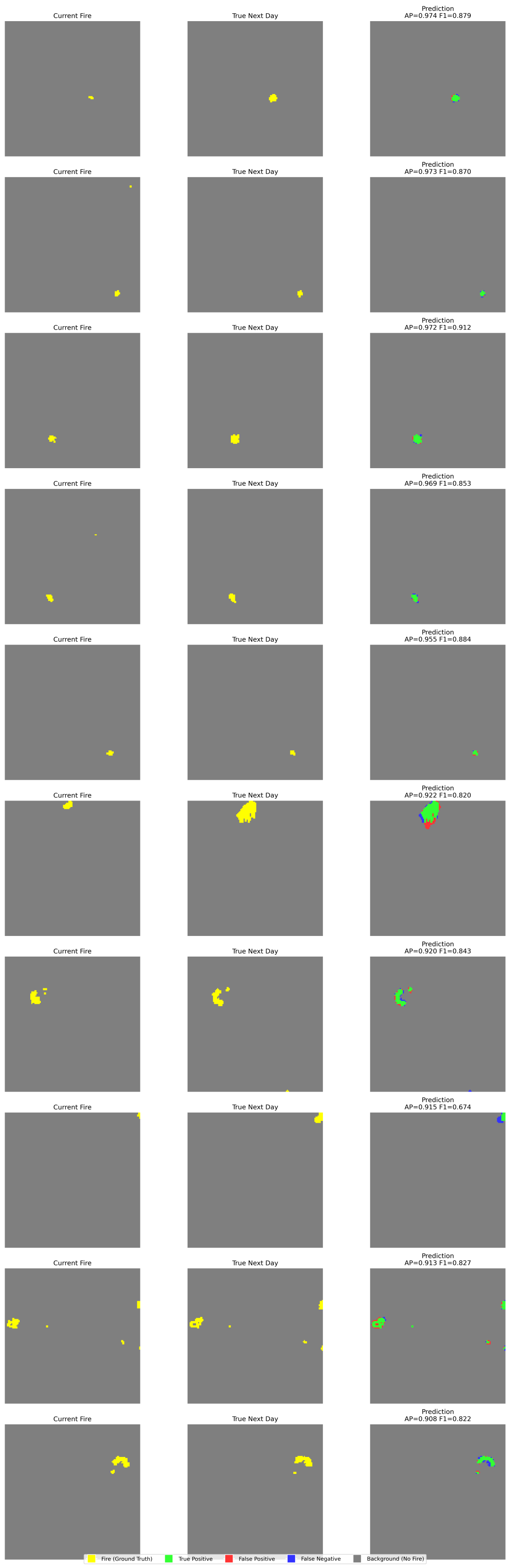

Figure 15. 10 best predictions made by the Res18-Unet on the 2020 test year.

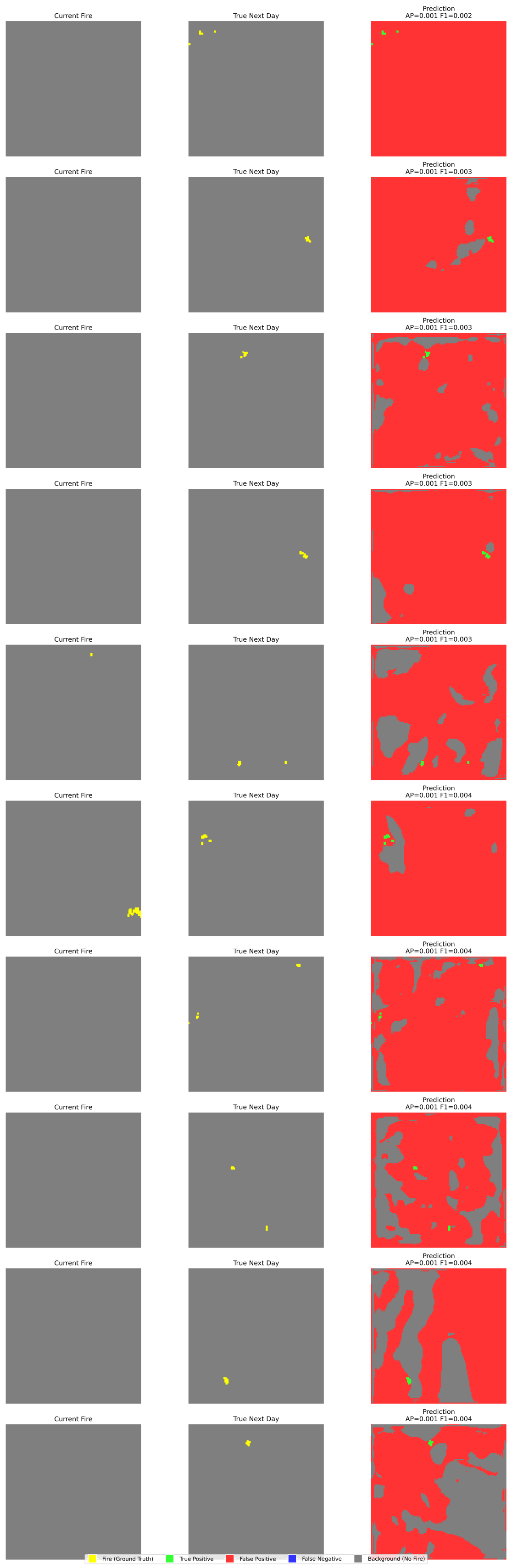

Figure 16. 10 worst predictions made by the Res18-Unet on the 2020 test year.

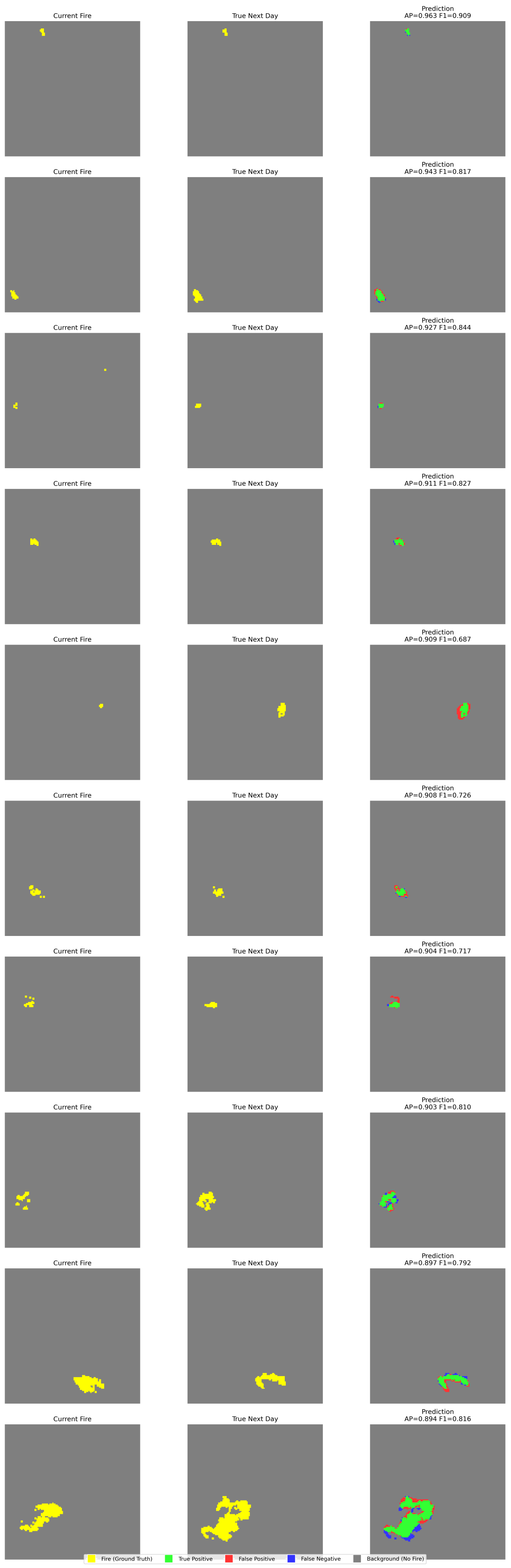

Figure 17. 10 best predictions made by the Res18-Unet on the 2021 test year.

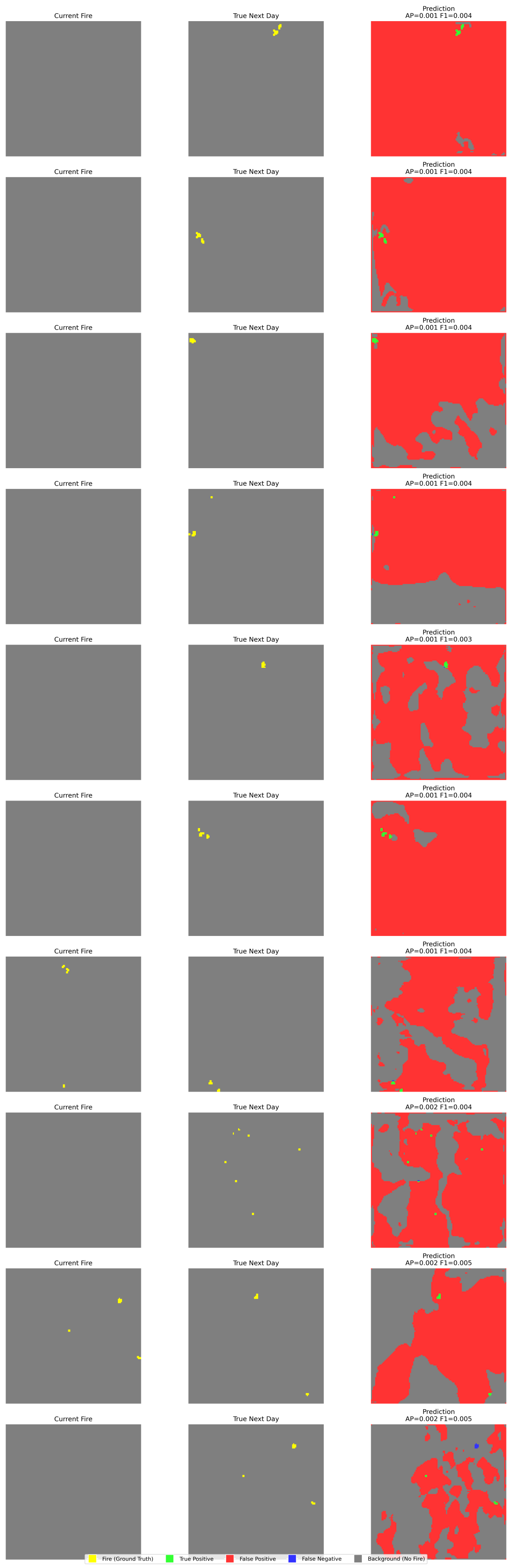

Figure 18. 10 worst predictions made by the Res18-Unet on the 2021 test year.

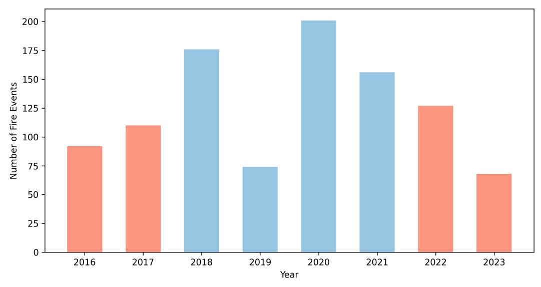

Figure 19. Distribution of fire events in WSTS+ per year

folds (2018, 2019; and 2020, 2021) and found virtually no difference (max difference was 7.11e-04% of each other). Further, to ensure these differences were not meaningful, we trained a Res18-Unet with T=1 on the Multi feature set (the best performing one from Tab. 2) using our replicated WSTS and the original one. To verify that the results are similar, we show in Tab. 9 the test performance on each individual year and found that they are within a small numerical error of each other.

Upon collecting the additional data in WSTS+, we computed the means and variances of each explanatory feature (e.g., wind speed, humidity, NDVI, EVI2, ERC) as well as the active fire feature across both original years (2018-2021) and newly added ones (2016, 2017, 2022, and 2023), and found that the distributions suggest some distribution shift. Fig. 24 shows kernel density estimates of the yearly distributions of multiple explanatory features in the dataset, highlighting varying degrees of cross-year domain shift across years. To validate this hypothesis, we conduct the cross-year experiments described in Sec. 10.4.

Table 9. Comparison of model performance on WSTS original data versus our replicated WSTS data.

| Test Year | Original | Replicated |
|---|---|---|
| 2018 | 0.49533 | 0.48594 |
| 2019 | 0.31190 | 0.32115 |
| 2020 | 0.42248 | 0.41793 |
| 2021 | 0.56742 | 0.56031 |
| Average | 0.44928 | 0.44633 |

## 10.3. Inter-Year Domain Shift: Additional Analysis

In this section, we perform additional analysis of domain shift. It is well-known within the fire science community that fire spread is impacted by a diverse set of environmental factors such as weather (e.g., wind, temperature, precipitation), topography, and fuel quantity and type [22, 44]. These factors also vary substantially across locations and time. For example, fire behavior experts have long established that annual weather patterns strongly influence both fire prevalence and extent [48].

Most of these important environmental factors are all encoded by one or multiple, input features in the WSTS dataset. Furthermore, the creators of the WSTS dataset presented evidence that most of these features influence the likelihood of fire spread, to varying degrees (see Table 6 in [17]). Here, we build on those findings and report various evidence (e.g., visualizations, histograms, or statistics) that these features exhibit substantial inter-year variability as well, providing further evidence of cross-year domain shifts. We focus our analyses on the features that were found to be (statistically) most influential of fire spread within the WSTS dataset, from the analysis in [17].

**Landcover Classes** According to Table 6 in [17], the categorical variables with the highest importance (by absolute mean coefficient) were dominated by the different land cover classes (absolute coefficients between 5-28). Looking at our plots, we notice in Fig. 20 that the proportions of landcover types vary substantially between years. For instance, LC 10 (Grasslands) was highly represented in 2016 (56.2%) and 2022 (50.0%) but comprised a much smaller proportion in 2021 (23.8%) and 2023 (20.6%). Additionally, LC 8 (Woody Savannas), shifts from being a minor component in 2016 (1.5%) to a major landscape feature in 2019 and 2023 ( 20% of the area). We also notice a considerable decline for LC 11 (Permanent Wetlands), where it represented a significant portion of the landcover in 2016-2017 (+22%), but shrunk to less than 6% in more recent years like 2021 and 2023.

Furthermore, many landcover classes were nearly or totally absent in some years. For example, LC 2 (Evergreen Broadleaf Forests) is not represented in 2016, 2019, or 2022, and LC 14 (Cropland/Natural Vegetation Mosaics) is only found in 2016 (2.5%) and 2020 (0.4%). We also observe that the proportion of LC 5 (Mixed Forests) jumped to 13.3% in 2023, after being absent in all prior years, suggesting a recent shift in the fire's environment.

**Active Fire** Aside from landcover classes, active fire masks were -unsurprisingly- the features with the highest importance. As such, we visualize in Fig. 21 the proportion of events where the fire mask size is zero for each year, alongside KDE plots showing fire size distribution after filtering out zero-sized events. To better visualize the skewed data, we log-transformed the x-axis, and to ensure a fair comparison, we balanced the number of fire events by randomly sampling events to match the smallest amount available for any given year.

Similar to the landcover plots, Fig. 21 reveals significant year-to-year variation, both for zero- and non-zero-

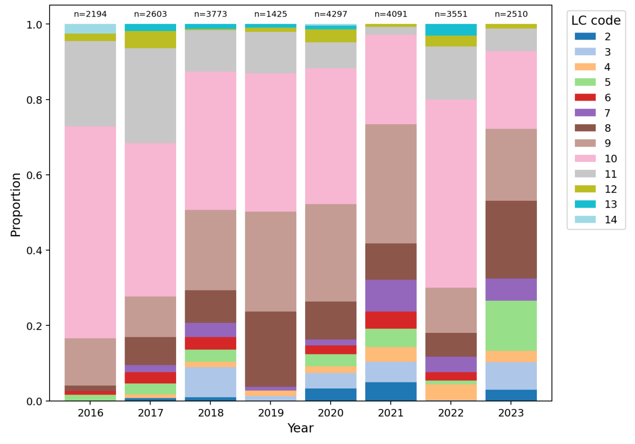

Figure 20. Histogram of landcover proportions across different years.

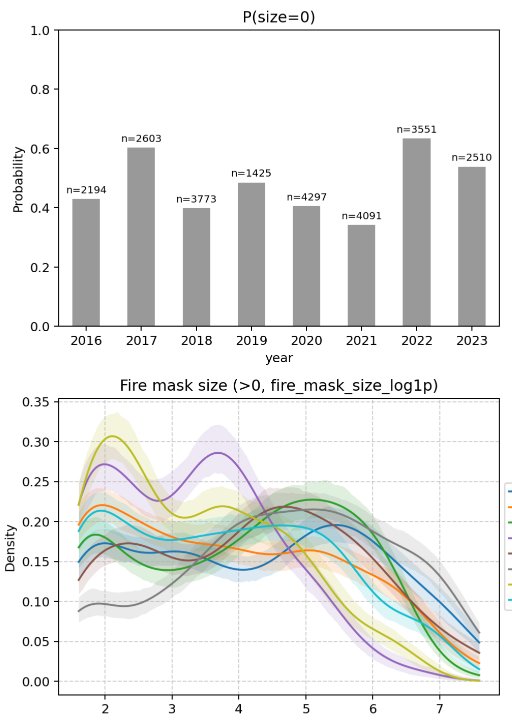

Figure 21. Top: Probability of a fire event having size zero for each year. Bottom: KDE showing distribution of fire sizes for fires that were larger than zero. The x-axis is log-transformed for better visualization, and the fire events balanced to ensure a fair comparison across years.

sized events. For example, 2021 had the lowest probability of zero-sized events ( 35%), i.e., a significantly higher proportion of observations in 2021 had active fires compared to other years. Looking back at the results in Tab. 6, we observe that when 2021 is used as a testing year, we achieve the highest average AP, regardless of training year (0.567). On the other hand, 2017 and 2022 had much higher probabilities (+60%), pointing to much higher fire inactivity, which exacerbates the class imabalance, and can also be seen in the relatively low results in Tab. 6, where the average AP of models tested on these years was 0.282 and 0.322, respectively.

Looking at the KDE plot, we further observe significant variability in the shape and typical size of fire events across years. For instance, the 2022 distribution (shown in olive) has a single, sharp peak at a smaller fire size, while the 2021 distribution (shown in purple) is much broader and has two distinct peaks, suggesting two common modes of fire size in that year. Both distributions are shifted to the left, indicating smaller fires on average. On the other hand, the gray curve for 2021 is shifted furthest to the right, meaning that larger fires were more common that year. Finally, 2016, 2017, 2018, and 2023 are quite mixed, with broad spread that indicates highly variable fire sizes.

**Continuous Features** As for the continuous features, we find from [17] that the ones with the highest importance were Total precipitation (-22.014), Forecast: Total precipitation (-9.865), NDVI (+4.178), Elevation (+2.933), Energy release component (+2.637), Slope (+2.406), VIIRS band M11 (+2.156), Maximum temperature (+0.948), Specific humidity (+0.857), Minimum temperature (+0.690). As such, we visualize KDE plots for each of them in Fig. 24. Similar to Fig. 21, we each line represents a KDE curve (with confidence intervals), showing the probability density across feature values in a given year. We also applied balanced subsampling, so that all years have the same number of samples, making comparisons fair.

The precipitation plots are both similarly skewed toward very low values, indicating that the precipitation values do not vary as much from year to year. However, looking at the distribution of forecast zero-precipitation events in Fig. 22 reveals a different insight: 2021 ( 0.8) and 2022/2023 ( 0.7) have many more zero-precipitation samples than 2016–2020, indicating that for some years, forecast no-rain events dominate, while in others forecasts predict more wet conditions. On the other hand, distributions of observed no-rain are nearly identical across years. Therefore, we can attribute the domain shift to the frequency of precipitation forecasts, but not to the differences in the magnitude of forecast or observed precipitation.

The remaining features show different levels of interannual variability, with each year's curve having a different shape (i.e., some are unimodal, some are bimodal), and peaking at different values. For instance, looking at the NDVI plot, we observe that 2016 is unimodal and peaks around an NDVI of 4000, while 2023 is bimodal and peaks much higher, around 7000, suggesting a greener year. This means that the type and condition of vegetation fueling the

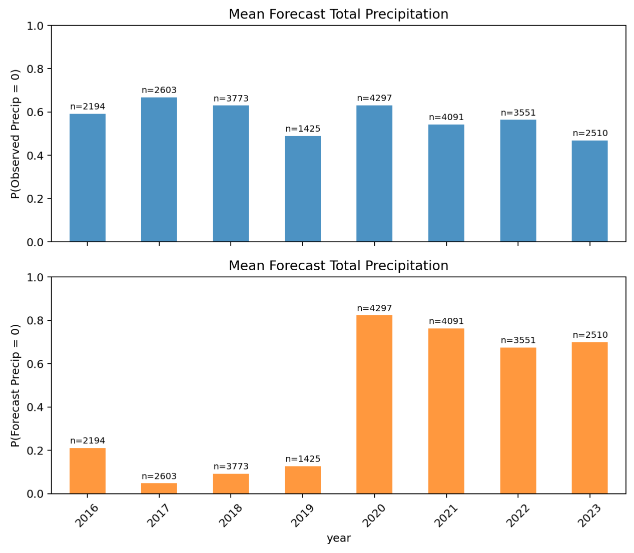

Figure 22. Distribution of forecast (top) and observed (bottom) no-rain events across years

fires varied significantly from year to year.

The elevation plot also shows significant variability of average elevation of fire locations between years, with the curves having completely different shapes and peaks. For instance, 2016 and 2022 peak around 400, while 2021 peaks at 1000, and 2018 at both 500 and 2000, suggesting that fires occurred in separate geographies with distinct elevations that year.

Looking at the Energy Release Component (ERC) distributions reveals that curves shifted to the right, like 2018 (in green), experienced more severe drought conditions, therefore higher potential for intense fires. Overall, the significant spread across years highlights major differences in drought, a key driver of fire season severity.

The slope plot shows that the 2022 fires (in olive) occurred on flatter terrain relative to the other years (low peaks). The M11, min/max temperatures, and specific humidity plots show significant overlap, with some outlier years (e.g., 2016 M11 peaking at 2000, correlating with hotter, intense fires; 2017 max temperature peaking at 305, indicating fire occurring in hotter conditions).

**Fire Growth** To quantify the average daily fire growth pattern, we examine the number of active fire pixels on each day after ignition. In Fig. 23, we plot the average progression of active fire pixels over the first 35 days following ignition, with a separate line for each year of data (blue lines represent WSTS years, while red one represent WSTS+ new years), and 95% confidence intervals. As evidenced by the contrast in recorded fires between 2019 (1,422 fires) and 2020 (4,297 fires) shown in the legend, weather con-

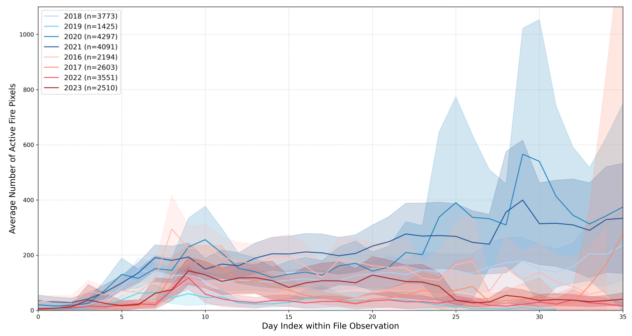

Figure 23. Comparing fire growth behavior for the different years in WSTS+ reveals significant interannual variability

ditions drive substantial interannual fire variability. Fig. 23 highlights this variability between years, with many annual patterns falling entirely outside the confidence intervals of other years. Notably, years with greater fire activity, such as 2020 and 2021, exhibit more explosive growth during the first five days after ignition compared to other years.

## 10.4. Cross-Year Experimental Design

We discuss here the experimental design of the results shown in Tab. 6 in the main manuscript. We trained a Res18Unet model on each training dataset listed in Tab. 10. Each year contributed a fixed quantity (and importance) of data samples (338 per year) to a shared validation set. We reached that number by reserving 20% of the data of the year with the least amount of samples (2019 had 1351 total samples) as validation and used that number for all other years, resulting in 2704 validation samples across 8 years, which represented between 8.25% and 16% of the total samples of the remaining 7 years. The training sets contain $min(2000, |N|)$ where $N$ is the total data available for that year, after removing the validation samples. This ensured the training sets across years had roughly the same amount of data to train on (all years ended up having 2000 samples, except for 2016 and 2019, with 1751 and 1002 samples, respectively). We then evaluated each model's average precision (AP) on each test set in Tab. 10. Notably, we ensured the training and validation sets for each year contain disjoint sets of fire events.

Table 10. Training, validation, and testing set sizes for each year, used for the cross-year train/testing.

| **Year** | **2016** | **2017** | **2018** | **2019** | **2020** | **2021** | **2022** | **2023** |
|---|---|---|---|---|---|---|---|---|
| Training Set Size | 1385 | 1697 | 2000 | 692 | 2000 | 2000 | 2000 | 1818 |
| Validation Set Size | 2704 | | | | | | | |
| Testing Set Size | 338 | 338 | 338 | 338 | 338 | 338 | 338 | 338 |

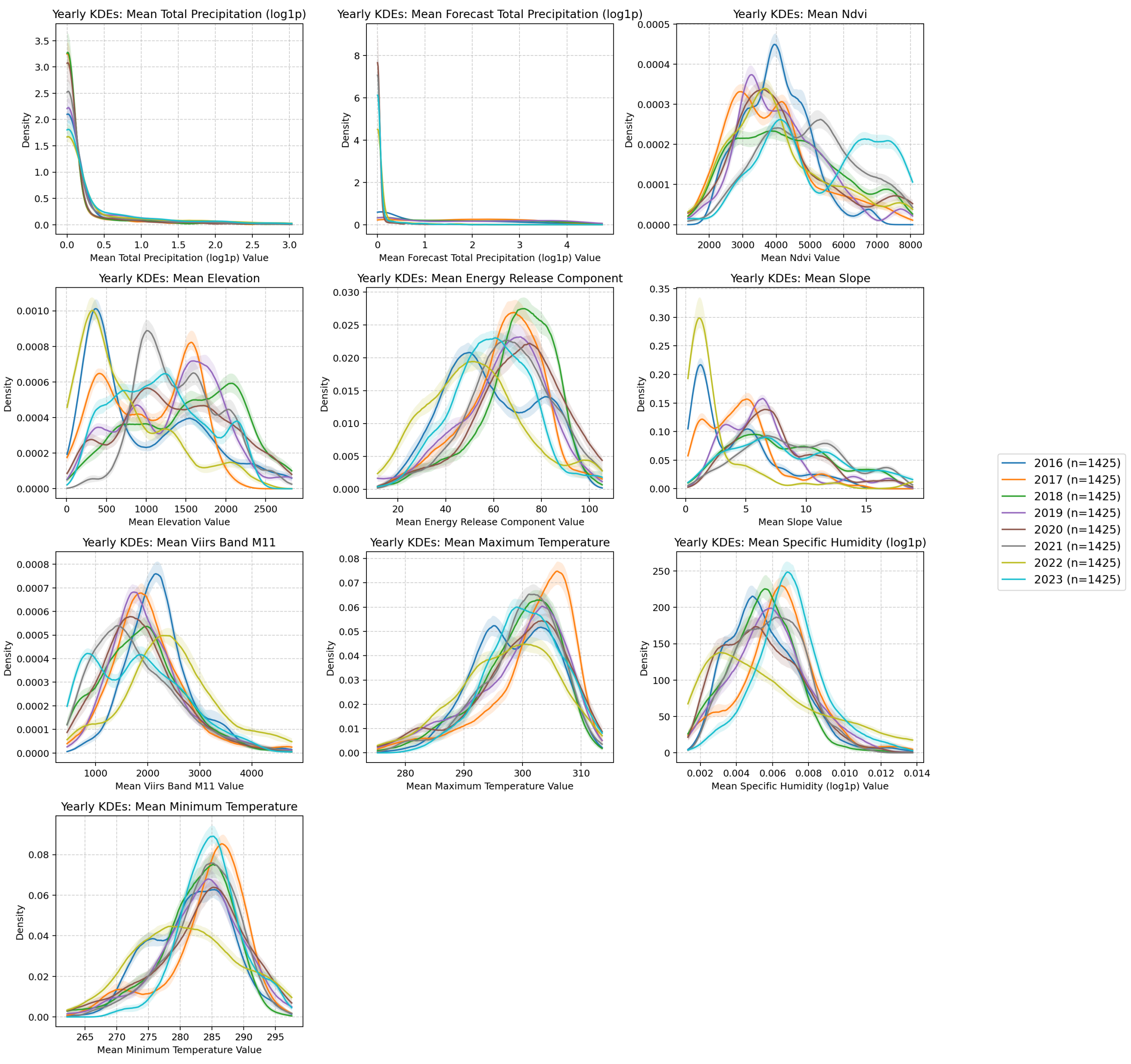

Figure 24. KDE facets of continuous features, plotted separately for each year.