UNIVERSITY OF CALIFORNIA SAN DIEGO

Where's the Smoke and Fire?
Exploiting Spatial and Temporal Context for Improved Wildfire Detection

A Thesis submitted in partial satisfaction of the requirements
for the degree Master of Science

in

Computer Science

by

Yash Pande

Committee in charge:

Professor Garrison Cottrell, Chair
Professor Manmohan Chandrekar
Professor Hao Su

2021

The Thesis of Yash Pande is approved, and is acceptable in quality and form for publication on microfilm and electronically.

University of California, San Diego

2021

DEDICATION

To Gary Cottrell, whose freshman seminar on neural networks led me down a path culminating in this thesis. Thank you for the great mentorship and underappreciated wit; I can't imagine what I would be doing right now if I hadn't met you. Thanks also to Wally Cottrell, may he rest in peace, for keeping me awake through Gary's early morning lectures and lab meetings.

EPIGRAPH

To deal with hyper-planes in a 14-dimensional space, visualize a 3-D space and say "fourteen" to yourself very loudly. Everyone does it.

—Geoffrey Hinton

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Thank you to Mai Nguyen for letting me work with her on so many interesting projects. Thanks also to Olga Liakhovich for introducing me to the world of applied machine learning and object detection.

Thank you to Hassler, Austin, and all the wonderful people I met at Microsoft, Google and Deepmind for such rewarding internships. Thanks to Sanjay, Dylan, and the many other researchers I've had the pleasure of working with in GURU.

Thank you to both Professor Manmohan Chandrekar and Professor Hao Su for agreeing to be on my thesis committee. Thank you also to Professor Hao Su for letting me join his lab on a project when I was just a second-year student and inspiring me to pursue more computer vision research.

Thank you to the many wonderful professors I have had at UCSD for shaping me into the learner I am today. Special thanks to Professors Paturi, Bejenaru, Kane and Rhoades.

ABSTRACT OF THE THESIS

Where's the Smoke and Fire?
Exploiting Spatial and Temporal Context for Improved Wildfire Detection

by

Yash Pande
Master of Science in Computer Science
University of California San Diego, 2021
Professor Garrison Cottrell, Chair

This thesis covers techniques to improve wildfire detection accuracy through the use of spatial and temporal context. Our dataset of wildfires consists of high-resolution images with smoke plumes that need to be detected early, when they are smallest, to allow firefighters enough time to react. We propose two novel architectures - the first combines a region proposal network on a lower-resolution image with a sequence network on the full-resolution image for efficient, accurate predictions; and the second combines a ResNet feature extractor with a vision transformer for high resolution tiled image classification. We also propose novel loss functions, combining a precise per-grid-element loss with a coarser per-image loss to achieve better precision in our detected fires without increasing the amount of false positives. With these techniques, we achieve state of the art results on this dataset with an accuracy of 89% and an average time to detection of 12 seconds.

Chapter 1

Introduction

As the result of unchecked climate change the number of wildfires in the US has doubled in the last 30 years. This has led to some states, especially California, seeing large portions of their forests burned in the past few years in what have been some of the largest wildfires in their history. A technique crucial to preventing large wildfires is detecting and controlling them in their early stages, as their rate of growth increases with size.

Our primary task is to use computer vision to detect wildfires visible in cameras that are part of the HPWREN camera network in an effort to speed up emergency response times. Our dataset comes from videos of past fires that have occurred in areas visible from the HPWREN camera network, whose frames have been manually labeled with a timestamp corresponding to how long before or after the fire each frame was and with segmentation masks and bounding boxes on all the frames containing smoke.

The dataset consists of high resolution color images that are either 2048 by 1536 or 3072 by 2048 pixels in size. The size of the smoke varies, and can range anywhere from 100 square pixels to over 100,000. The hardest smoke to classify is the early-stage smoke, which is darker and smaller in size than later-stage smoke. Unfortunately, this is also the most important smoke to classify as early detection and response is integral to stopping wildfires early.

We will focus on detecting a wildfire as early as possible - our metric will be minimizing the number of seconds between when a fire starts and when we detect it. We will also attempt to improve the overall accuracy, precision and recall of our detections, although we are willing to have a higher number of false positives if it decreases the number

of false negatives. While a false positive can easily be rejected by a human verifier, a false negative may lead to costly delays in responding to a fire.

We will focus on using deep-learning-based approaches to this problem. Deep learning is machine learning using deep neural networks, which use interconnected processing nodes (neurons) inspired by the human brain. They are parametrized by the weights between interconnected neurons, and trained to minimize a loss function on a dataset, where in our case, the loss function will be based on how well they classify smoke vs non-smoke.

In particular, we will be making use of convolutional neural networks (CNNs), which are neural networks designed specifically for computer vision tasks. These have seen increased use in computer vision since Krizhevsky et. al. [14] used a CNN to achieve state of the art results on the Imagenet image classification competition. The greatest benefit of these networks is their ability to learn features themselves based on a labeled dataset, which replaces the laborious process of manually designing features. In the next section, we will discuss the neural network architectures primarily used in this thesis, namely convolutional neural networks, recurrent neural networks, transformer networks, and the Mask-RCNN image segmentation architecture.

Chapter 2

Background

Modern neural network architectures build upon basic neural networks to more efficiently account for known relational inductive biases and to provide a desired invariance [4]. We will use these architectures to create a network that learns more efficiently and requires less data to train, ultimately generalizing better to unseen data. Specifically, we will use convolutional neural networks for image processing. These are designed to account for locality - the idea that nearby pixels will be highly correlated, and translational invariance - the idea that the same cluster of pixels in different parts of the image should elicit similar outputs. We will also use recurrent neural networks, which are designed for sequence processing - they account for sequential locality, the idea that nearby objects in a sequence will be highly correlated, and for time translational invariance - the same smaller sequence should elicit a similar output regardless of where it occurs in the larger sequence. We will be using recurrent neural networks because we will be analyzing sequences of images over time, trying to detect when the first instance of a fire occurs. Finally, we will be using the transformer architecture, which uses a mechanism known as self-attention to automatically determine non-local dependencies among its input and integrate information over those dependencies. We will use this architecture in two forms - a spatial transformer that operates on a single image broken into smaller tiles, and a temporal transformer that operates on a sequence of images. In both case, we intend that the additional contextual information gained through the transformer network will improve our model's learning ability.

2.1 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a form of artificial neural networks that learn convolutional filters to apply over an image [16]. We will be applying them to our input images in all our networks, due to their effectiveness at analyzing images which stems from the aforementioned inductive biases and invariances. Each filter itself is typically small in size, ranging from 3 by 3 to 7 by 7, and these filters are convolved over the entire image to create a filter map. The use of filters is how CNNs account for spatial locality - each filter operates on a local neighborhood of its center, taking advantage of the fact that nearby pixels in images are highly correlated to the center pixel while further away ones are conditionally independent of the center pixel given its neighbors. By repeatedly applying these filters on the input image, we can generate higher-level features that encode relevant salient information about the image. As mentioned earlier, an important feature of CNNs for our purposes is that they automatically learn what filters are best at identifying the target objects, and in our case where smoke can have a variety of shapes and sizes this is a more effective technique than trying to manually create features.

Another feature of CNNs is the use of pooling operations, which replace each patch (typically 2 by 2) in a feature map with either the average or max of the patch. This reduces the size of subsequent layers, allowing for fewer parameters in the model. Importantly, in the case of max pooling, it also adds translational invariance to the network. Since max pooling takes a max over 2 by 2 patches, and since each CNN architecture typically includes multiple max pooling layers, we can shift the input to the CNN by multiple pixels and the output will not change, allowing for a more robust network. The networks themselves are biologically inspired and are designed to mimic the local receptive field of a neuron in the visual cortex. An important feature of convolutional neural networks is that, as opposed to fully-connected neural networks, they use the same feature extractors at different parts of the image. For our use case, where smoke may appear in any part of the image, this is incredibly helpful as it allows us to learn general filters regardless of the location of the smoke.

2.2 Mask-RCNN Object Detection

While CNNs are generally helpful for image-related tasks, our specific goal is to detect smoke in images, and there are certain CNN architectures designed specifically for the task of object detection and segmentation. Since our labels include a precise segmentation of exactly which pixels have smoke in them, if we use an architecture designed for segmentation we can make use of this information as additional signal. One such architecture is the Mask-RCNN architecture [10], a two-stage image segmentation network.

The first stage of the Mask-RCNN architecture is a backbone feature extraction network, typically a Resnet-50 or Resnet-101. This takes an image and outputs a feature map with size proportional to the input image. These features are then fed into the region proposal network (RPN) [24], which convolves over the feature map and at each spatial window maps the feature map to a lower-dimensional embedding (e.g. a 512-dimensional vector). For each spatial window, it has a fixed number of "anchors", which are potential box shapes at that window. For each anchor, it uses the lower-dimensional embedding to predict whether or not the anchor contains an object, as well as how the anchor's shape should change to match the object. For k anchors, this corresponds to a size 2k class output (object or not object) and a size 4k regression output (corresponding to the change in height, width, x-value of center, and y-value of center). It then uses a process called ROI Align to align the features extracted by the backbone with the regions of interest proposed by the RPN. This is then fed into the second-stage classifier which in parallel predicts the class, bounding box and mask of each input box. This classifier predicts the segmentation mask for each possible class, without any competition between the classes. The class output is instead used to determine the correct class of the object, and the mask corresponding to that class is used as the output mask.

We train the Mask-RCNN with only two classes - smoke and background, with segmentation labels for the smoke. One thing to note is that Mask-RCNN networks are designed for a single image input, not for seqeuences of images. Since our input is a stream of images from a single camera, we want to make use of architectures that allow us include this temporal context. This will be especially important when distinguishing smoke from clouds, as clouds look very similar to smoke but have very different temporal dynamics.

2.3 Recurrent Neural Networks

One approach to capture the temporal dynamics in the images, which will allow us to differentiate between similar-looking smoke and clouds, is to use a recurrent neural network, which is designed to analyze sequences [26]. These consist of a single network that is applied to every element in a sequence and outputs a "hidden state" for every element in the sequence. In some tasks, such as sequence labeling, each hidden state is then used to predict an output corresponding to that element of the sequence (e.g. labeling the part of speech of a word). In other tasks, such as our smoke prediction task, only the final hidden state is used - we only care about whether or not the current (last in sequence) frame has smoke in it, as we will already have predicted the status of previous frames. LSTM networks are a form of recurrent neural networks that use gated memory for better long-term information storage [12]. While our sequences are not necessarily long enough to require an LSTM, as only around 5 frames (corresponding to 200 seconds) are necessary to distinguish between the movement of smoke and clouds, we use them as we found them to empirically perform better than vanilla RNNs.

2.4 Transformer Networks

As mentioned in the earlier sections, CNNs are designed to account for spatial locality within pixels and RNNs/LTSMs are designed to account for temporal locality in sequences. A mechanism known as *attention* is a more general technique to capture dependencies between correlated parts of an input, and the transformer network is an architecture that uses only attention to operate on sequences. In networks that apply multiple steps of processing to an input, attention is the idea of re-weighting the input at a given index based on how relevant each part of the input is to the element at that index. This allows elements of the input to freely communicate with other elements regardless of the distance between them, based only on how related the elements are. This is helpful for our purposes because it allow us to pass forward information from more important frames (such as those where a fire might have started) regardless of how far behind they are.

The transformer network, originally designed for language tasks, takes in a sequential input and uses only attention to share information within that sequence. This has the

benefit of being highly parallelizable and also of having a better flow of gradients, which leads to faster training. As mentioned earlier, we know that the sort of the temporal dynamics we are trying to capture can be done in a fixed length, so this is not an issue. The faster training time is helpful, as is the reduced model complexity.

Each attention unit in a transformer network calculates three weight matrices: the key weights, the value weights, and the query weights. These are then multiplied by each input element x_i to obtain the key/value/query vectors for that element. If we want to compute the weighted combination of neighboring elements for a given element i, we first compute the attention weight between i and any element j as the softmax of the dot product of the query vector of i and key vector of j. We then multiply this by the value vector of j, and add that over all possible j to get our weighted input.

We use sequential transformer networks as an alternative to LSTMs due to the aforementioned benefits. We also use transformer networks in the form of a vision transformer [8] to share information between different tiles of the same image, allowing us to use the high resolution of our input images without having an excessively large network size. We will discuss this in further detail in the experiments section.

Chapter 3

Related Work

We first note that our dataset labels were developed for this project and have not been used in any publications. Furthermore, there are no benchmark datasets for the problem of smoke segmentation that we can compare to, and none of the cited smoke detection papers have released the datasets that they operated on. As a result, any performance metrics mentioned in this section cannot be fairly compared to those we discuss in our experiments section.

AI for Mankind has used some unlabeled images from the same camera source as our dataset for previous hackathons, but they have no published papers and their work still appears to be in the experimental phase. Their best detection times range from 3-13 minutes before a fire is detected, which are significantly longer than our models [1]. They use a pretrained Single Shot Detector [19] from the Tensorflow model zoo and fine tune it on their dataset, but there are no novel techniques used in their work that we can draw inspiration from.

There has been a relatively small amount of existing work on the subject of smoke detection. The limiting factor tends to be lack of labeled data, as images of smoke are hard to find. One paper [34] focuses on creating synthetic smoke images by overlaying images of smoke on natural backgrounds. They train an R-CNN based object detection model and show that a model trained on the synthetic dataset has 99.7% accuracy on a dataset of real fires. Unfortunately these numbers are misleading as both the train and test datasets are incredibly unbalanced - there are 30 non-fire images in the test set out of 12620 overall, so a network predicting only true would achieve a 99.8% accuracy.

Another paper [31] focuses on close-up smoke segmentation, using an encoder-decoder network with dual fine and coarse paths to precisely segment the smoke. The goal of this paper, however, is precise segmentation of smoke that covers the entire image, which is very different from our task of detecting small smoke in a small part of an image. They achieve an intersection over union of between 69-71% on three different datasets that they have created.

A major limitation of image segmentation models is that they do not account for movement in the image, which can be one of few distinguishing factors between smoke from fires and haze or clouds. We see in Figure 3.1 how a Faster-RCNN mistakes haze for smoke with high confidence

Figure 3.1: Haze being incorrectly labeled as smoke with 77% confidence.

More recent works have incorporated optical flow and background subtraction [32] to remove noise and non-smoke movement from a sequence of images before inputting them to a deep CNN. Their paper focuses more on background removal than on actual smoke detection, but we make use of the strategies mentioned in our experiments to try and improve performance. The first of two strategies mentioned is ViBe background subtraction [3], which matches pixels with nearby pixels in previous frames in an attempt to find which pixels are not background pixels (i.e., which ones do not have a match in previous frames). It uses ViBe to determine areas of interest, which it then highlights as input to the CNN. The second strategy mentioned is dense optical flow, calculated using the Farneback method [9]. This method uses quadratic polynomials to estimate the displacement of each pixel between subsequent frames, which results in a vector field of

predicted displacements for each pixel. The paper uses this vector field as additional input to its deep CNN model. These strategies are relevant to our task, so we implemented both and used them in our experiments.

Another recent work uses 3D CNN architectures [17] to incorporate temporal context, allowing the network to better differentiate smoke from other similar-looking objects. Again, their primary focus is on quality of segmentation and not on early detection, and they achieve an intersection over union of .78 on a monochrome background and .75 on a complex background. Since we feel that the goal of precise smoke segmentation is ancillary to our primary objective of detecting wildfires as soon as they occur, we did not further explore this technique.

Another recent paper [6] proposes the use of a bidirectional LSTM that is fed in potential smoke locations based on a background subtraction method. A bidirectional LSTM implies that information flows in both directions, which means they use both future frames and previous ones to predict the current frame. They use an attention mechanism within their bi-LSTM, and they use the aforementioned ViBe background subtraction method to detect areas of interest. Their objective is slightly different from ours - they have sequences of images that are either entirely fire or entirely non-fire, and their goal is to categorize such images. They use metrics such as precision and recall for their predictions, but their goal is not early detection of fires, as their network requires the entire sequence to be input at once.

Our objective is very different - we want to feed our network a constant stream of images and have it output smoke on the first frame that contains smoke and all subsequent frames. This is in line with our goal of detecting fires as early as possible, without as few false negatives as possible.

In the general field of high resolution object detection, Zeng et. al. [33] offer a deep-learning-based approach to high resolution image segmentation. This uses a network that fuses local information with more global information about the image in order to obtain an accurate, high-resolution segmentation of the image. We decided not to use this architecture for two reasons - first, it is focused on precise segmentation, which is not really necessary for our needs, as we only need to determine the general location of the fire. Second, it focuses on cases where objects form a signficant part of the image (i.e. more

than half the pixels of the image belong to the object) and this is why their local/global feature fusion is important. In our case, when we are more concerned with detecting very small fires within a much larger image, such a network is less useful.

Chapter 4

Data

As mentioned earlier, our dataset comes from the HPWREN camera network[1] which currently consists of 35 cameras across Southern California. These cameras are all connected to the HPWREN network, which enables high speed data transmission in remote areas. Each camera has a 360-degree view of its surroundings, and takes a picture every 60 seconds which it uploads to the central image repository. Overall, we have images of 263 fires, of which 134 are fully labeled. These are all color images, in two possible resolutions - 2048 by 1536 or 3072 by 2048 pixels in size. We resize all images to 2048 by 1536 so that our networks can have uniform input sizes.

These images have been labeled in two ways - first, for any fire that occurred in view of an HPWREN camera, the images from that camera ranging from 2400 seconds before the fire to 2400 seconds after the fire are combined into a folder and labeled with a relative timestamp (relative to when the fire started, so ranging from -2400 to 2400). This makes for a total of 81 frames. These labels are useful in the case of whole-image classification, since we can just use the relative timestamp label to determine whether or not an image has fire. All 263 fires are labeled in this manner.

The second set of labels we have are segmentation labels, which have two parts - bounding boxes and segmentation masks. The bounding box is a set of (x_1, y_1, x_2, y_2) coordinates denoting the area that smoke is within, and the segmentation is a set of points forming the convex hull of the precise location of smoke. The decision to use this format to represent the segmentation data instead of a binary mask is because it is more

[1] https://hpwren.ucsd.edu/cameras/

space efficient. The 134 fully labeled fires come with segmentation labels in most frames, although there are certain frames where the labelers were unable to detect or to properly segment the fire and these frames are skipped.

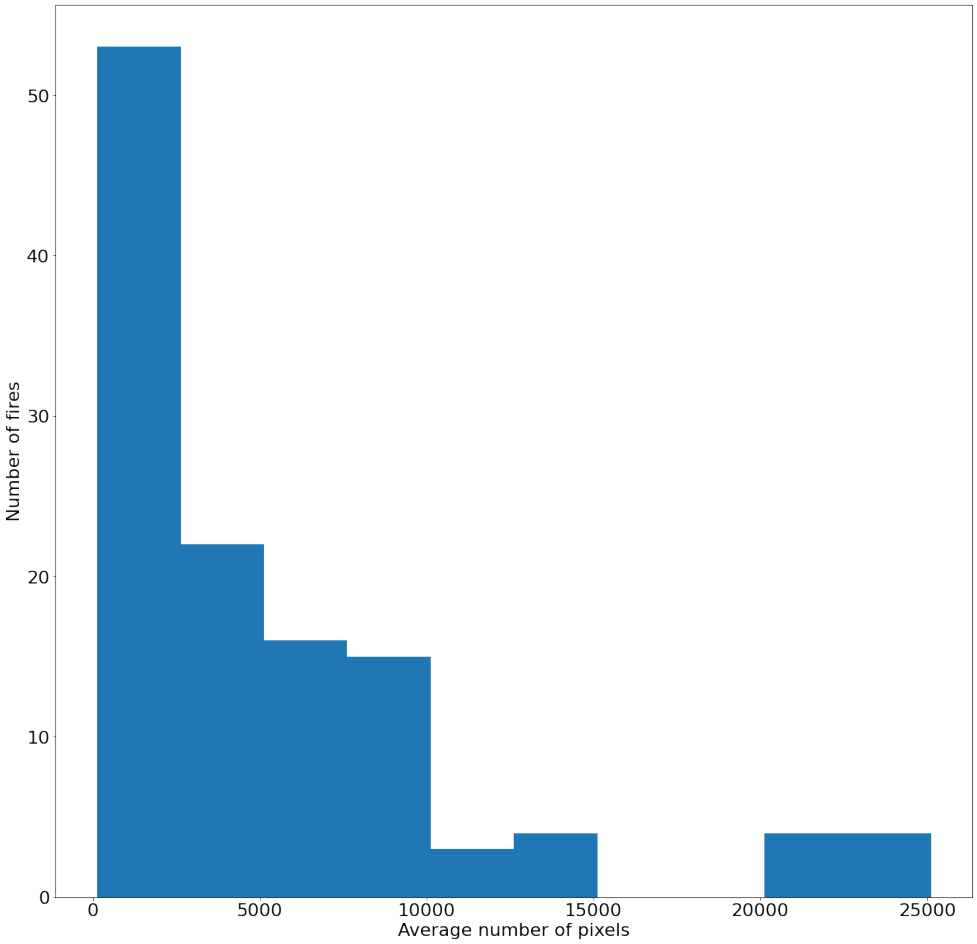

Figure 4.1: Histogram of fire sizes (with top 10% of fires removed)

In 4.1 we see the average size, in pixels, of each fully labeled fire. We note that the top 10% of fires are not shown in this histogram for visualization purposes, as their average size is over 80,000 pixels. These fires typically occurred very close to the cameras and took up a large portion of the field of view.

Of these images, we created a standardized train, validation and test split containing 75%, 10% and 15% of the images respectively. The split was done on a per-fire basis, so all images from a single fire would occur in exactly one of the three splits. As we will

discuss in the next section, some of our architectures took entire images as input, while others broke images down into tiles of size 224 by 224. This was done by splitting each image into 108 tiles with 9 rows and 12 columns, with each tile having an overlap of 20 pixels on each edge with its neighboring tiles (the tiles in the last row and last column had slightly larger overlap so they could fit entirely within the image). When operating on these tiles, we created binary labels ("smoke" or "no smoke") for each tile by checking if the segmentation mask for the image had any overlap with that tile.

When looking at the sizes of fire over time, there is a notable gap between the average size of a fire before 300 seconds ($\tilde{6}000$ pixels) and those after 300 seconds ($\tilde{1}200$ pixels). We can therefore choose 300 seconds as our arbitrary benchmark for an "early stage" fire, with the intent of focusing on how well we can detect such early stage fires.

Chapter 5

Experiments

Neural network architectures are parametrized by their weights θ. The goal of training a network on a dataset is to find the parameters that minimize the value of a desired loss function on the overall dataset:

$$\theta_{opt} = argmin_{\theta} \left(\Sigma_{i=n}^{n} Loss(N_{\theta}(x_i), y_i)\right)$$

Where the x_i are inputs and the y_i are targets, and $N_{\theta}(x_i)$ is the output of the network parametrized by θ with an input of x_i. As the space of possible weights is incredibly large, and as we do not typically have access to the entire dataset, we choose a differentiable loss function and use gradient descent to train our network. This involves initializing θ to a random value and then updating θ to move in the direction of the negative gradient of the loss function with respect to θ: $\theta = \theta - \gamma \frac{\nabla Loss}{\theta}$ where γ is a hyper-parameter. Typically this gradient calculation is done on randomly sampled batches of the dataset, not the entire dataset, in a process known as stochastic gradient descent [5]. Further techniques include Nesterov Momentum [22], which involves moving in the average direction of your previous movements before calculating the gradient at each step; this improves the speed at which a network converges to an optimal point. More complex optimizers such as Adam [13] use moments of the objective function to achieve more stable convergence. In our experiments, we will use either stochastic gradient descent with Nesterov Momentum (henceforth referred to as SGD) or the Adam optimizer to train our networks. All training was done on the PyTorch framework [23], which is also where all pretrained model weights

came from. Two standard forms of data augmentation were used across all images - horizontal flips and adding gaussian noise - to help prevent our network from overfitting. We did not use random cropping and vertical flips because all cameras are set to have a fixed field of view, so we felt that these augmentations would not match anything we would see in the real world.

5.1 Metrics

As mentioned earlier, we have a fixed split of train, validation and test data. Our general evaluation procedure is as follows: we first train a network for multiple iterations (i.e. epochs) using only data from the training set. We then evaluate the network from each iteration on the validation set, which it has never been trained on, and select the network that performs best on the validation set as our "final" network. We evaluate our final network on the test set to obtain our formal metrics for an experiment. All metrics provided are the average of five experiments. The metrics we use are as follows:

Accuracy

This is defined as the proportion of images that were correctly labeled: $\frac{\text{number correct}}{\text{total number}}$. Accuracy is an important metric for analyzing the overall performance of the model, but in severely imbalanced datasets it can lead to trivial networks having high performance. For example, in the dataset where we split each image into 108 tiles, the vast majority of those tiles (over 97%) do not have smoke in them. As a result, a network that simply predicts "not smoke" on every tile will have very high accuracy despite being highly undesirable.

Precision

This is defined as the proportion of "true" guesses that were actually true. In our case, it is $\frac{\text{number correctly labeled as smoke}}{\text{number network labeled as smoke}}$. Precision is important for our purposes, as an imprecise network will give too many false alarms, but it is not as important as recall.

Recall

This is defined as the proportion of "true" objects that were identified as true by our network. In our case, it is $\frac{\text{number correctly labeled as smoke}}{\text{number of ground truth labeled as smoke}}$. Recall is our most important metric, as a high recall implies our network is able to detect most fires. Nonetheless, it is important to balance precision with recall as a network predicting only "smoke" can have a perfect recall but will also not be very helpful.

Dice Score

For segmentation tasks, the DICE score between two sets X and Y [7] is defined as:

$$\frac{2|X \cup Y|}{|X| + |Y|}$$

For our purposes, X is the set of pixels predicted as smoke, while Y is the set of ground truth smoke pixels. The DICE score is a good metric for segmentation tasks as it balances precision and recall - a network performing well on dice must have a segmentation that is very close to the ground truth, without missing too many pixels or overpredicting on too many pixels.

5.2 Loss Functions

Our loss functions need to be differentiable to use gradient descent, and our metrics are often not. Instead we create loss functions that, when used to optimize a network, will result in that network having strong performance with regards to our metrics.

Binary Cross Entropy

Using our earlier definition where x_i are the inputs, y_i are targets, and $\text{net}(x_i)$ is our network's prediction on input x_i, we define binary cross entropy (BCE) loss as:

$$-\Sigma_{i=n}^{n} y_i log(\text{net}(x_i)) + (1 - y_i) log(1 - \text{net}(x_i))$$

We note that since y_i is binary, half of the inner sum is always zero for each part of the outer sum. BCE loss is designed to minimize the distance, as defined by KL-divergence [15],

between the true distribution of labels and our network's predicted distribution. If BCE loss is zero, it means our network is predicting the correct label for each element of the dataset, with total confidence. We can also weight BCE loss to emphasize recall over precision - we add an $\alpha > 1$ term in front of the $y_i log(\text{net}(x_i))$. This results in a higher loss for predicting 0 when the true value was 1 than for predicting 1 when the true value of 0. We use this weighted BCE loss in our experiments to obtain networks that have higher recall at the potential cost of precision, because as mentioned earlier recall is more important for our purposes.

Our models can be split into two subsets - those that operate on entire images, and those that operate on tiled images. The following are models that operate on entire images:

5.3 Networks on Entire Image

5.3.1 Binary CNN

Architecture

Our baseline model was a binary CNN that took as input entire images and predicted "smoke" and "no smoke". We used two different CNN models - one was a shallow 6-layer model trained from scratch and the other was a Resnet-101 model [11]. The 6-layer model was as follows (5.1):

Table 5.1: Description of Custom CNN Layers

Layer Type	Parameters
Convolutional Layer	5x5 Kernel, 16 Filters
MaxPool Layer	2x2 Kernel
Convolutional Layer	5x5 Kernel, 32 Filters
MaxPool Layer	2x2 Kernel
Convolutional Layer	5x5 Kernel, 64 Filters
MaxPool Layer	2x2 Kernel
Full Connected Layer	1024 ReLU Units
Full Connected Layer	64 ReLU Units
Full Connected Layer	1 ReLU Output Unit

The Resnet-101 model is a deep, 101-layer model that uses "shortcut" connections between layers for better gradient flow, allowing each of the 101 layers to train. Because the Resnet-101 architecture has orders of magnitude more parameters than the 6-layer CNN model, we initalize its parameters to those of a model that was pre-trained on Imagenet [27] and then fine-tuned it on the smoke dataset. The pretrained weights were provided by Facebook as part of their detectron framework [30].

As mentioned earlier, the raw images were resized to 2048 by 1536 pixels for input to the 6-layer CNN model, while they were resized to 224 by 224 pixels for input to the Resnet-101 model to match the pretrained weights.

The Resnet-101 was initialized to ImageNet weights, while the 6-layer CNN was initialized to random weights. Both were trained using SGD, with a learning rate of .001. Both were trained for 50 epochs on the entire dataset, with early stopping used to stop training after 5 epochs with no improvement in validation loss. The loss function used was BCE loss, with a weight of $\alpha = 5$ used to emphasize recall over precision.

Results

Table 5.2: Results on Binary CNN

Network	Accuracy	Precision	Recall
6-Layer CNN	58%	67%	46%
Resnet-101	65%	72%	53%

The Resnet-101 had better performance than the shallow model, but in both cases the per-image accuracy was only slightly better than chance, which is too low for our purposes. We calculate a separate recall on the "early stage" fires, defined previously as fires within their first 300 seconds, and the recall for both models was under 20%.

Directions for Improvement

There were two primary issues with these architectures - first, they were not powerful enough to accurately detect smoke in the fires. The 6-layer CNN lacked the expressivity to properly distinguish between clouds, haze and smoke; and the Resnet-101 was limited because the input image needed to be resized in order to fit the pretrained weights which

severely limited the resolution of the input. Second, these single-image input models did not account for the movement between subsequent frames, which can be very important in distinguishing between smoke and clouds. Because we have precise segmentation masks for many of the fires, we also wish to use an architecture designed for segmentation so we can make use of these labels, particularly one designed to detect smaller objects in larger images.

Optical Flow and Background Subtraction

As a minor improvement to both the binary CNN models and the Mask-RCNN models we will discuss in the next section, we computed the dense optical flow between subsequent frames as additional input to the CNN. The optical flow was computed using the Farneback method [9] and it was input as two additional channels (representing the flow in the x and y directions respectively) to the CNN.

The goal was to allow the model to use the movement from previous frames as a means of differentiating between smoke and clouds. This was done by simply computing and scaling the x and y vector values of optical flow at each pixel, and appending it to the RGB value of that pixel. In both models this had a mild improvement ($< 3\%$) on accuracy, but it did not solve any of the more pressing issues regarding failures on early smoke images. When visually inspecting the optical flow output, it appeared very noisy and seemed to be picking up on minor landscape moments due to wind more than it was picking up on the actual smoke movement. Part of this is due to the fact that the images are taken 40 seconds apart, and there can be a lot of movement between subsequent frames. We experimented with different hyperparameters in the optical flow, and also attempted to use morphological transforms (dilation and erosion) to reduce the amount of noise in the optical flow output.

We similarly used an implementation of the ViBe background subtraction [3] method in an attempt to distinguish smoke and other foreground objects from the background before input to the CNN. The output of ViBe is a binary mask indicating which pixels are background and which are foreground. We attempted to use this in two ways - in the first, we simply fed the binary mask as additional input to the CNN. In the second, we slightly expanded the size of the mask using a dilation transform and then applied the mask to

our image before feeding it to the CNN. In the first case, accuracy did not change, while in the second it decreased. Upon visual inspection, we saw that ViBe was often detecting trees moving in the wind as foreground, and at times it was not detecting actual smoke as being in the foreground. Ultimately, we chose to use neither optical flow nor ViBe in our final experiments.

5.3.2 Mask-RCNN

Architecture

As described earlier, the Mask-RCNN architecture is designed for detecting and segmenting multiple objects in an image. Due to its use of a two-stage detection process with a region proposal network, it is able to detect smaller objects that do not take up a large portion of the image. We used a standard Mask-RCNN model from Facebook's Detectron implementation [30], with a Resnet-50 backbone. We initialized our model to pretrained weights given by Facebook from a model trained on the COCO (Train 2017) dataset [18]. We also modified the RPN_ANCHOR_SCALES hyperparameter, which controls the sizes of the proposed anchor boxes, to be (16, 32, 64, 128, 256). This improves the network's detection of small objects, as small as 16 by 16 pixels. Again, because we were using a pre-trained Mask-RCNN model we had to resize the image to 224 by 224, which limited the level of fine-grained image information that the network had access to. Training was done using SGD, and for 150 epochs. Early stopping was used to end training early if validation loss did not improve after 5 iterations. The custom loss function used by Mask-RCNN was modified to emphasize recall over precision with a weight of 5.

Results

When considered on a per-image basis, this gave us better results than the earlier Binary CNN model with an accuracy of 80%, a precision of 73% and a recall of 85%. The recall on early-stage smoke was still low at 43%, but better than the earlier models. On a per-segmentation basis, however, the model was not very precise, with an average DICE score of .36. Part of the reason for the low DICE score is the inherent noise in labeling smoke, as it is often difficult to differentiate smoke from its surroundings. As a result, even

the ground truth labels had many cases where smoke was imprecisely or inconsistently labeled (Figure 5.1). This is why we use per-image results as our benchmark instead of segmentation DICE (per-image results are also more aligned with our ultimate goal of early fire detection, as the segmentation accuracy really does not matter). Our use of a custom loss function on the mask-RCNN head helps explain the higher number of false positives as compared to false negatives. Most of the false positives were clouds and most of the false negatives were earlier / smaller smoke plumes.

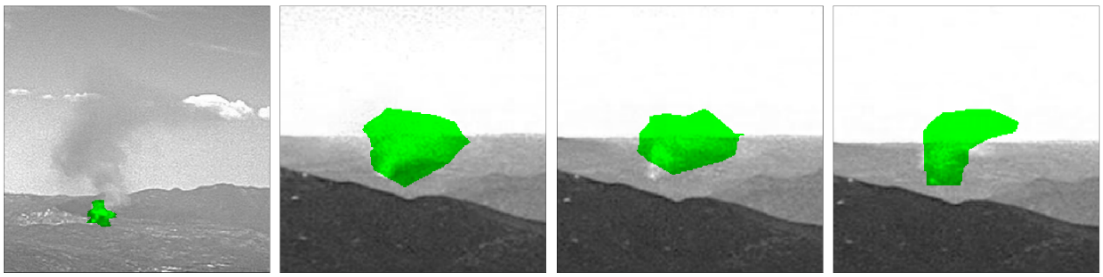

Figure 5.1: Examples of how the manual segmentations of smoke can be inconsistent and imprecise.

U-Net on Mask-RCNN Input

The U-Net architecture is an encoder-decoder fully convolutional architecture designed for precise segmentation [25]. It uses down-convolutions to encode an image then up-convolutions to decode it, predicting a mask as its output layer, and uses skip-connections to share information between corresponding encoder and decoder layers. Its fully convolutional nature allows it to be used on images of any size, and its segmentation is more precise than competing architectures.

Because the Mask-RCNN was imprecise at segmentation, we trained a U-Net to segment the predicted bounding boxes from the Mask-RCNN. Along with its segmentation, the Mask-RCNN outputs a bounding box for each object it detects. We cropped the image and the ground truth segmentation mask to the shape and location of this bounding box, then fed the input the U-Net to train.

The DICE score of the U-Net on these cropped images was .41, which was slightly better than that of the Mask-RCNN. Again, it is difficult for the U-Net to learn precise

segmentation because the images themselves were not precisely labeled. Furthermore, any object that was not detected by the Mask-RCNN would also not be detected by the U-Net, as the U-Net was only fed the output of the Mask-RCNN.

Overall, the Mask-RCNN seemed to be the best single-image object detection and segmentation architecture, but it was unable to use the full resolution of the images and it was also unable to use any of the temporal context available, both of which were important factors. However, the Mask-RCNN's two-stage detection process was effective at proposing potential smoke regions, and the technique of feeding images cropped to Mask-RCNN bounding boxes into another network (the U-Net) was somewhat successful. This inspired us to pursue architectures that augmented the Mask-RCNN performance by making use of temporal context.

5.3.3 LSTM on Mask-RCNN Input

Architecture

Because we felt that temporal context was important in distinguishing between smoke and non-smoke, we decided to use an LSTM that would take in a sequence of images and predict the class of the final image. Furthermore, because we observed that the CNN was unable to detect images well at a high resolution, we used the Mask-RCNN's region proposal network to focus only on potential smoke areas.

We ran the Mask-RCNN network on the downsized version of the full-resolution image, and extracted the potential regions that it suggested. We then went back to the full-resolution image, cropped the portion corresponding to the region proposal, and did so for the current frame as well as 5 previous frames for a total of 6. We resized these crops to a fixed size (224 by 224), inputted them to the 6-layer CNN for feature extraction, then inputted the extracted features into the LSTM network. The LSTM encoding of the final input to the network (i.e. the last frame in the sequence that we wanted to predict on) was then fed to a fully connected network, whose output was a binary prediction corresponding to "smoke" or "no smoke".

Since the LSTM was itself deciding whether or not an object had smoke, we could set the sensitivity threshold for the Mask-RCNN to a very low value; we used 20%. This prevented the Mask-RCNN from excluding too many actual smoke images from its pre-

dictions, at the cost of including more non-smoke images. The non-smoke images were not an issue because a well-trained LSTM would be able to classify them as not smoke. We chose not to use a Bi-LSTM because as noted earlier it has the drawback of requiring future frames to predict current ones.

One important thing to note here is that we used holdout sets on our Mask-RCNN when creating a training set for the LSTM. We felt that the Mask-RCNN would perform worse on the test set than on the training set, imprecisely bounding objects and having more false positives, and we wanted an LSTM network that would be robust to these changes in performance. To remedy this, we augmented the training set for the LSTM network - we trained 6 versions of the Mask-RCNN on the train set - one was trained on the entire train set and predicted on the entire train set, while the remaining 5 each saw only 80% of the train set and predicted on the remaining 20% that they had not seen. We labeled the noisy predictions from the latter 5 networks with their true labels, allowing us to have a more realistic training set that includes predictions by a Mask-RCNN on images that it had not been trained on before. This improves the generalizability of our network and better emulates the real-world usage.

The LSTM network is randomly initialized, and training is done using the SGD optimizer, with a learning rate of .001 for 50 epochs. Early stopping is used to stop training when there is no improvement in validation metrics for 10 epochs.

Results

We achieved an accuracy of 83%, a precision of 78% and a recall of 86%. The recall of early-stage smoke did not improve notably, and was 47%. This is our best result so far - adding the temporal context improved the precision of our detections, likely because it allowed us to filter clouds and other objects apart from smoke. However, it did not have a significant impact on recall. We noticed that in the LSTM, the gradient flow to the CNN was very small, and the weights of the CNN network were not changing by much while training. Overall, we felt that our idea of combining an LSTM with a Mask-RCNN allowed us to better use temporal context to precisely classify objects, but the lack of gradient flow to the CNN was an important issue we needed to resolve.

5.3.4 Transformer on Mask-RCNN Input

Architecture

To fix the issue of the LSTM not training as well, we decided to use a transformer network with ReZero initialization [2](to appear in UAI 2021). The ReZero architecture adds a residual connection to transformers and removes the LayerNorm, leading to better gradient flow and faster training (see Figure 5.2 for an architecture diagram).

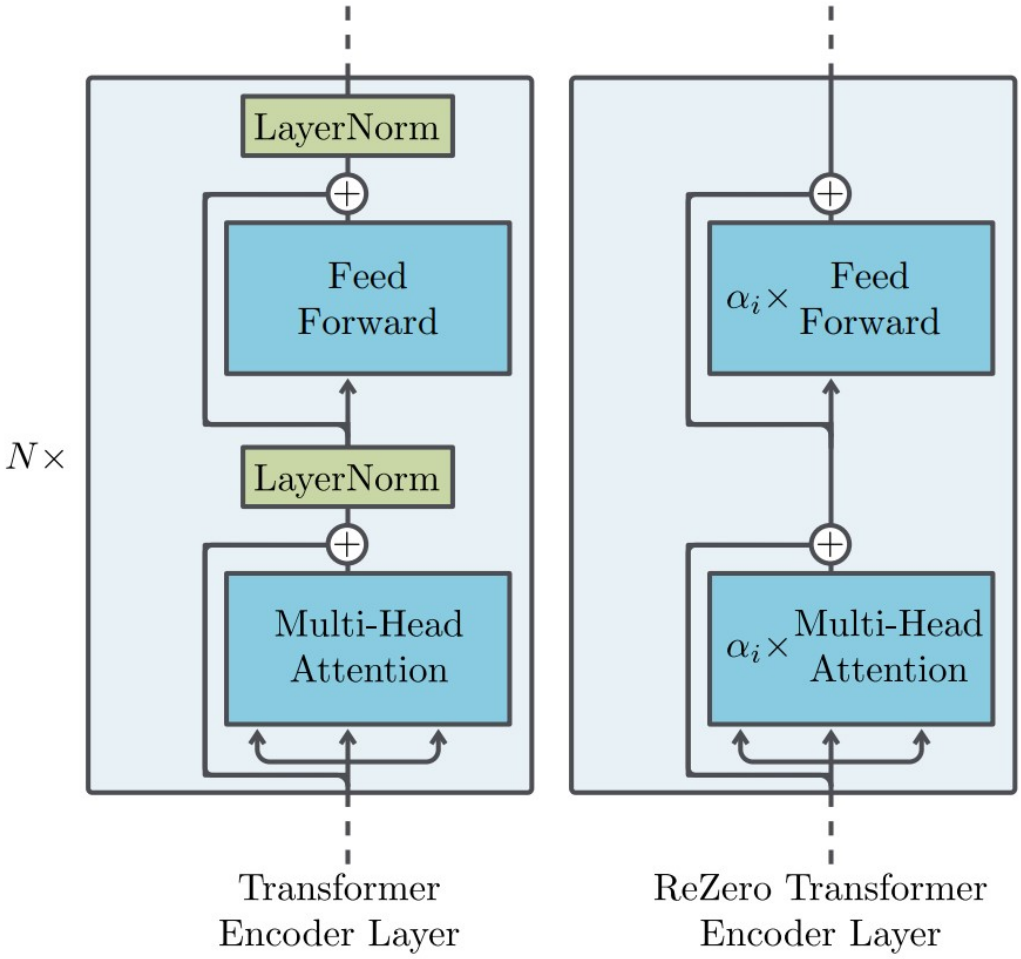

Figure 5.2: Diagram of the ReZero Transformer (from [2]).

The goal was to improve upon the limitations of the LSTM model, namely that it was not propagating gradients to the CNN. The same technique of using the Mask-RCNN region proposal network was used, and the same 6-layer CNN was used to extract features

from the Mask-RCNN proposal areas. The transformer had 6 towers, one corresponding to each timestep, and a sinusoidal embedding is added to the input as a positional embedding. After running the transformer encoding network for 6 steps, we extract the encoded value of the final tower (corresponding to the final element in the sequence, the one whose value we want to predict). We then run it through a fully connected network with a binary output. The transformer network is randomly initialized, and training is done using the Adam optimizer, with a learning rate of .001 for 50 epochs. Early stopping is used to stop training when there is no improvement in validation metrics for 10 epochs.

Results

This network trained efficiently, with the CNN weights changing and the overall network achieving an accuracy of 96% on the overall training dataset. The high train accuracy was a case of overfitting, and the model that performed best on the validation dataset was an earlier one with a lower training accuracy. This model attained an accuracy of 86%, with a precision of 82% and a recall of 87%. Again, it seems as though the recall of the model was limited by the initial Mask-RCNN predictions, as it did not improve much from the LSTM, but the higher accuracy and prediction mean the better gradient flow led to a better-trained network. In Figures 5.3 and 5.4, we see examples of the Mask-RCNN predictions on two images, with the confidence thresholds of the bounding box modified to be those outputted by the transformer network. The top of the figure contains the full-resolution image, while the bottom is cropped closer to the actual detection. In these cases, we can see how the Mask-RCNN has high confidence on its predictions, and how it is precise in both segmenting and bounding the location of the smoke.

We show the number of true positive and false positive predictions over time for the entire test set in Figure 5.5. We note that for this figure a prediction was only considered valid if it had overlap with a ground truth prediction. This figure shows that while the network has some false positives, as shown before 0 seconds, it manages to predict the post-fire frames as well as the ground truth does.

One thing to note is the reason there are more correct predictions than ground truth predictions - as noted earlier, the ground truth labels are sometimes inaccurate, and at times they do not provide segmentation masks for a fire's first few frames (in cases where

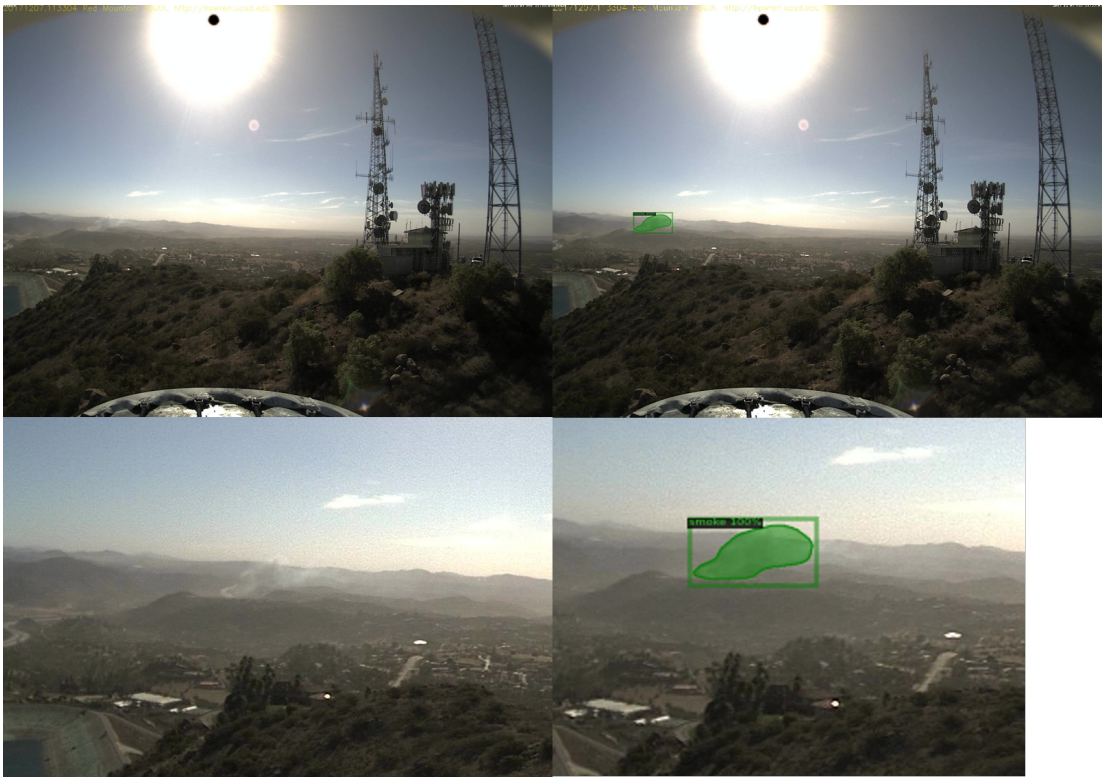

Figure 5.3: Visualized Prediction on 20171207 Lilac rm-s-mobo/1512675184 +01200.

Figure 5.4: Visualized Prediction on 20170722 FIRE hp-e-mobo-c/1500761547 +00300.

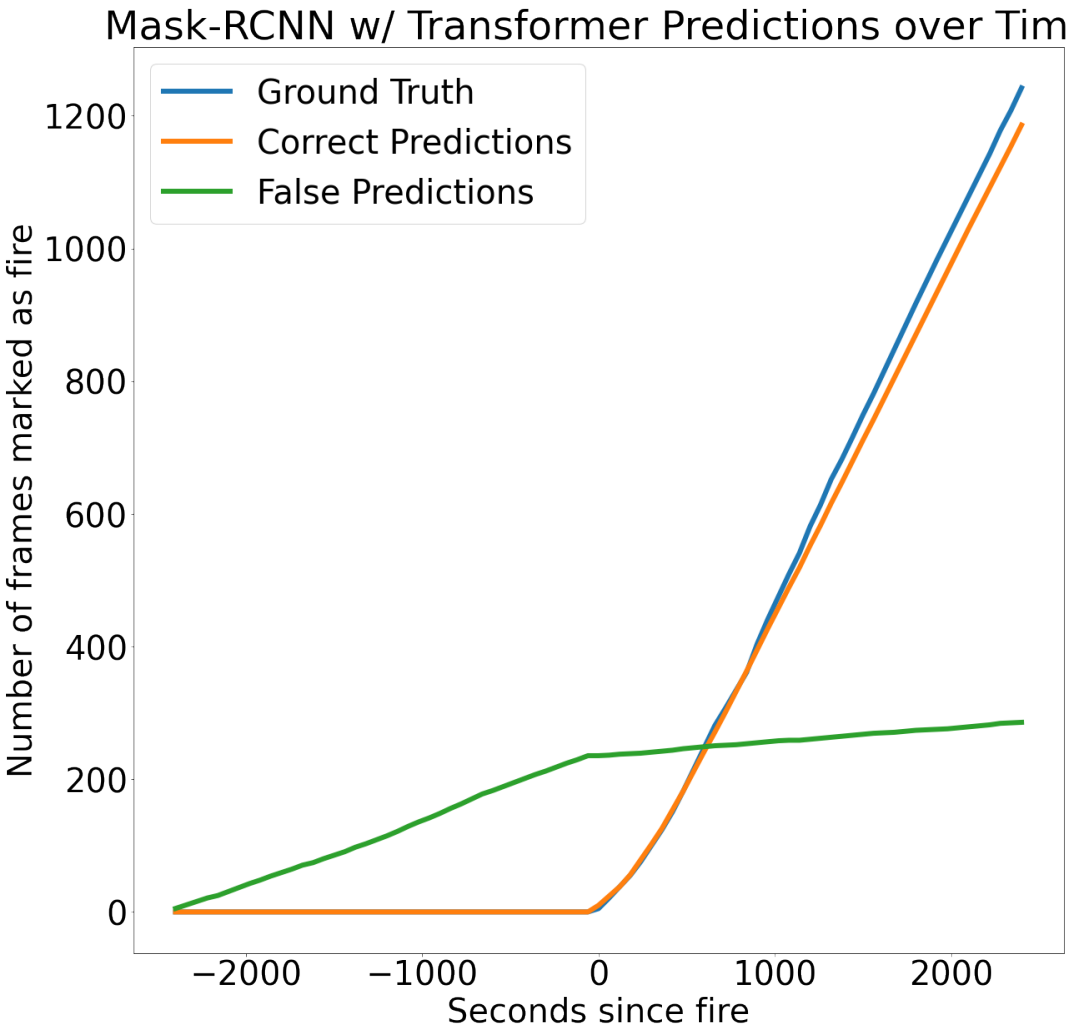

Figure 5.5: Seconds since fire versus number of frames detected.

the labeler did not notice the fire). As a result, it is possible that the network predicts an area as positive, we know there is a fire in that photo (since the timestamp, which is based on when the fire is known to have started, is ≥ 0), but we don't know where the fire actually is because there is no ground truth label. This is because the segmentation labeling is separate from the timestamp labeling, and at times the segmentation labelers were unable to find a fire even though they knew one had started by the timestamp of the frame. In this case, we extrapolate backwards from the first *labeled* frame we have - assuming the first labeled frame is at timestep t, we give all timesteps from 0 to t the same label as timestep t. This label is obviously incorrect for the earlier frames, but we

make the assumption that wherever the fire was in earlier frames it must have been within the same boundaries as a fire in later frames, as fires only grow with time. This means if the network detected a fire in an unlabeled frame with timestamp ≥ 0, and the area it detected a fire in was marked as fire in the first labeled timestep t, we would consider it a true positive detection even if the frame is unlabeled.

Overall, it seems as though this network is an effective refinement of the Mask-RCNN architecture, with significantly better results that correlate well with our desired metrics. However, the recall appears to have hit a ceiling, and that ceiling appears to be based on two factors - first, the original object proposal is done by the Mask-RCNN, which only looks at a single frame. Sometimes seeing movement between frames can alert a network to potential smoke, but in our architecture the movement can only be used to determine that an object the Mask-RCNN thought was smoke isn't actually smoke, not to propose initial smoke locations. Second, the Mask-RCNN looks at a downsized version of the image, which is less than 2% of the resolution of the original image. This prevents it from seeing details that could be important in detecting early stage smoke. Again, even though the sequence networks have access to the full resolution of the image, they can only use it to determine that a proposal from the Mask-RCNN isn't smoke, as they cannot propose smoke locations themselves. One natural solution to these problems is to consider networks that operate on smaller portions of the image, allowing them to view the full resolution of the image.

5.4 Networks on Tiled Images

5.4.1 Resnet-101 on Tiled Input

Architecture

A natural solution to these problems is to have a network that is able to see both the full resolution of the image as well as the entire temporal context before it predicts potential smoke locations. Unfortunately it is too computationally expensive to run our networks on the full-size images, so an alternate proposal is to break the image into smaller tiles, each of which can individually be fed (at its full resolution) to a classification network.

This architecture is the same as the Resnet-101 architecture described earlier that was used for binary classification on whole images, except this model now operates on full-resolution 224 by 224 pixel tiles from the original image (each image is broken into 108 tiles). We again use the BCE loss weighted for higher recall with $\alpha = 5$, and the ground truth labels for tiles are computed as described earlier by calculating the intersection of the ground truth segmentation with each tile. The same training schedule is used as before - the Resnet-101 is initialized to ImageNet weights and trained using SGD for 50 epochs with a learning rate of .001. Early stopping is used to stop training after 5 epochs with no improvement in validation loss.

Results

This network achieves a per-tile accuracy of 97%, with a precision of 53% and a recall of 73%. We note that per-tile accuracy and recall don't correlate directly with per-image precision and recall, since there are far more "non-smoke" tiles as compared to "smoke" tiles. The per-image accuracy of this model is 74%, a precision of 68% and a recall of 76%. This was computed by labeling an entire image as "smoke" if any tile was predicted as smoke, and labeling an image as a true positive if any tile in the image was correctly predicted as "smoke". One thing to note here is that per-image accuracy suffers even relative to the Mask-RCNN models because we never explicitly optimized for it. If the model predicts one "smoke" tile and 107 "non-smoke" tiles in an image with 108 "non-smoke" tiles, its average loss will still be somewhat low. However, on a per-image basis this counts as a false positive. We can improve this by changing our losses to better account for per-image metrics, which we will discuss in a later section. Another issue with this model is that it considers each tile independently, not accounting for important information from neighboring tiles. This can make it difficult for the model to determine whether a small section of white within its tile is from smoke, a cloud, or something else altogether. For better performance, we should provide the model with the spatial context around its tile.

5.4.2 Vision Transformer on Tiled Input

Architecture

As mentioned before, considering each tile without its spatial context precludes our network from understanding global characteristics of the image, such as whether a given tile is closer to the sky or to the ground. Early smoke is more likely to occur near the ground, so our network should focus on tiles located here. Furthermore, the contiguity of smoke with nearby smoke can help distinguish it from clouds - if we confidently detect smoke in one tile, we should propagate that information to nearby tiles where smoke is harder to classify.

Here, we take inspiration from the Vision Transformer [8] and apply a transformer network to every tile in the grid. Two key differences between our network and the Vision Transformer are that our network separately classifies each tile in the grid while the Vision Transformer attempts to classify the image as a whole, and our network uses CNNs to extract features from each grid before applying the transformer while the Vision Transformer works on the raw pixels. The CNN used is the same 6-layer CNN we used for the whole-image experiments, and it is again trained from scratch. We also use much higher resolution images than most experiments with the Vision Transformer, which means each tile has a higher resolution and there are more of them (108 per image). This, combined with our relatively small dataset and limited compute resources, necessitates the use of a CNN for feature extraction. The network is randomly initialized, and training is done using the Adam optimizer, with a learning rate of .001 for 50 epochs. Early stopping is used to stop training when there is no improvement in validation metrics for 10 epochs.

Results

This network attains 98% accuracy on the test set for the per-tile classification task, with a precision of 81% and a recall of 86%. When considered on a per-image basis, the network obtains an accuracy of 87%, with a precision of 82% and a recall of 89%. These are competitive with the combined Mask-RCNN and transformer model (Section 5.3.4), and better than any of the other architectures.

Broadly, there are three issues remaining with this model architecture. First, the

network no longer has access to temporal context. Second, the CNN architecture is not as powerful as the Resnet-101 architecture, although it remains to be seen whether the network requires that level of expressivity. Finally, the binary loss function on a per-grid basis lends itself to solutions that emphasize predicting zeros. While we do weight the cross entropy loss to discourage the network from predicting zeros, there are other approaches where the loss is better correlated with our ultimate goal, as we will see in section 5.4.3.

5.4.3 Using Resnet-101 Embeddings

Architecture

As mentioned previously, we want to increase the learning capacity of our model by using a more powerful feature extractor. Unfortunately, we have two issues with this - first, our compute capacities are limited and we cannot train a Resnet-101 with a transformer network end to end. Even when using a Resnet-50, the combined architecture does not fit in 12GB of GPU memory. Second, the vanishing gradient problem is especially challenging when using a complex model such as a Resnet-101 as the feature extractor. It is likely that the Resnet-101 will not receive enough signal to train if it is directly connected to the transformer network.

To resolve both these issues, primarily that of our inability to fit the combined model in memory, we first train the Resnet-101 to classify individual tiles as "smoke" or "no-smoke", then we run inference on every tile within the dataset and use the values of the first (size 2048) fully-connected layer of the Resnet-101 as the embedding for that tile. These are the result of average pooling on the final convolutional layer. We then feed these embeddings into a fully connected network which inputs into the transformer network.

This allows the Resnet-101 to train directly on the binary classification task and learn to extract features relevant to that task, and then allows the transformer network to modify the embeddings of the Resnet-101 as needed for its purposes. We also reintroduce the idea of using previous frames by stacking embeddings for a given tile across the 5 most recent timesteps. The transformer network can then learn to either only use information from the most recent timestep or incorporate information from previous timesteps as needed. The decision to stack frames together as a form of temporal context is inspired by the Deepmind Atari network [21] which stacked subsequent frames of an Atari game

together as input to their network so it could determine the temporal dynamics of the game. The goal is that these modifications can resolve the first two issues mentioned in the previous architecture, namely the lack of temporal context and the less powerful feature extractor.

The training schedules for both the resnet and the vision transformer are the same as before - the Resnet-101 is initialized to ImageNet weights and trained using SGD for 50 epochs with a learning rate of .001. Early stopping is used to stop training after 5 epochs with no improvement in validation loss. The vision transformer is randomly initialized, and training is done using the Adam optimizer, with a learning rate of .001 for 50 epochs. Early stopping is used to stop training when there is no improvement in validation metrics for 10 epochs. A custom loss function was used to train this model, the details of which will be discussed in the next section.

Specificity and Sensitivity

When we began training models that operated on the grid of tiles instead of whole images, we noticed that binary cross entropy was not well aligned with our desired metrics. Since the proportion of smoke in the image was far lower than the proportion of non-smoke, a trivial classifier that always predicted zeros could do very well according to binary cross entropy loss. While weighting the loss is one method of resolving this issue, we felt a better technique would be to choose a loss function better aligned with our goals of a higher precision and recall in smoke images. One possibility is to compute a differentiable version of precision and recall and using it as part of our loss function, but this leads to another issue - precision and recall are both defined with true positives in the numerator (recall from earlier that precision is true positives over total true predictions and recall is true positives over total positive ground truths. In cases where there are no true positives, such as in half of our dataset (the frames before a fire starts), they do not provide a strong signal. Recall is clearly still very important to us, and an important metric for post-fire frames, but we want a better choice than precision to get more signal from our predictions on the pre-fire frames. One such option is specificity, the natural complement of recall - specificity is defined as true negatives over total negative ground truths. This means in pre-fire frames it tells us how good our network is at identifying non-fire tiles, allowing

it to still provide a signal to our network. For our needs, correctly identifying negative examples is beneficial as it increases the human-in-the-loop's confidence in the model.

As a result, we choose to use specificity and recall (also knows as sensitivity) as parts of our auxiliary loss function. We calculate both in a differentiable format, and weight sensitivity (i.e. recall) higher than specificity. This allows us to precisely tune our model for the desired levels of recall (at the obvious loss of sensitivity), which the regular dice score did not.

Because we always try and minimize the loss, but we want to maximize sensitivity and specificity, we subtract the soft sensitivity and specificity values from 1.

$$L_{aux} = 1 - \left(\lambda \frac{(\text{preds})(\text{target})}{sum(target) + \epsilon} + (1 - \lambda) \frac{(1 - \text{preds})(1 - \text{target})}{sum(1 - target) + \epsilon} \right)$$

Here, preds is the post-softmax predictions for each tile, target is the binary target value for each tile, and ϵ is used to prevent dividing by zero. Note that the first term in this loss is the sensitvity (i.e. recall) and the second term is the specificity.

We first train the model in a perfectly balanced (.5/.5) average of sensitivity and specificity, then we modify the loss weights to get a .75 to .25 weight ratio of sensitivity to specificity. This leads to a model whose per-tile precision is 64.0% and per-tile recall is 89.4%. The accuracy is around 99.3%.

These are higher numbers than any of our previous models, but one issue is that on a per-image basis there are still examples of false positives where a pre-fire image has one tile labeled as positive, which will still lead to the human-in-the-loop being asked to look at it.

This is because one negative prediction in a grid with 108 does not greatly affect the specificity of our network, but even one negative prediction for an image means that a human will be asked to check if there is a fire. This means our network can have a very low auxiliary loss but still have many false negatives on a per-image basis, which shows that our loss function is not well aligned with our objective. To fix this, we need to add some form of per-image loss that balances the per-grid losses.

Per-Image Loss

We notice that while the precision and recall numbers on a per-tile basis are high, there is still overprediction of the "smoke" class on a per-image basis. Essentially, if the network predicts only one tile out of 108 as having fire, even if that prediction is incorrect the overall recall remains at 100% while the precision is 0%. Since we are optimizing for recall over precision, such a prediction is not heavily penalized. However, on a per-image basis, this means that almost every image is being tagged as fire, which counteracts the benefit of having an automated detection system in the first place. We want to only alert when there is a fire, to ensure that we are minimizing the amount of human intervention needed. To do so, we can use a smooth maximum function that calculates a pseudo-maximum among the elements of a grid and compares it with the image's label. For example, if there are two tiles in the whole image that are predicted as "smoke", then the max over the image will have the label "smoke". We then take the binary cross entropy loss of this max compared to the image's actual label. The smooth maximum function is as follows:

$$\text{smoothmax}(x_1, ..., x_n) = \frac{\sum_{i=1}^{n} x_i e^{\alpha x_i}}{\sum_{i=1}^{n} e^{\alpha x_i}}$$

Where $x_1, ..., x_n$ are the per-grid values. α is a term used to determine how smooth or sharp the maximum is, and since in our case we want gradient to flow through multiple tiles we set $\alpha = .5$ for a smoother maximum. We then calculate the BCE loss between this smooth max and the true value for that image, allowing us to penalize the network for false positives. We note that we only use this loss function for frames occurring before a fire begins - in frames after a fire begins, this loss function would either do very little (in case the network predicted at least one grid as having fire) or penalize every grid element for predicting zero (since if every grid element is zero, the smooth maximum will be an evenly divided average of all of them, so the loss will be evenly distributed among all of them). The first of these cases is meaningless, and the second case is actually detrimental to our goal, as we don't want every grid to be predicting a positive (only the ones that actually have smoke).

So our final loss function is:

$$L(\text{preds}, \text{targets}) = BCE_\alpha(\text{preds}, \text{targets}) + 1 - \left(\lambda \frac{(\text{preds})(\text{target})}{sum(target) + \epsilon} + (1-\lambda)\frac{(1-\text{preds})(1-\text{target})}{sum(1-target)+\epsilon}\right) + (1 - \max(\text{target}))BCE_\alpha(\text{smoothmax}_\beta(\text{preds}), \max(\text{target}))$$

Where α weights the BCE towards penalizing false negatives more (we use $\alpha = 5$) and β indexes the smooth maximum (we use $\beta = .5$). Note that the first BCE is calculated element-wise between two vectors (then aggregated as a sum) while the second one is between two real numbers. The (1-max(target)) term before the second BCE loss means it is set to zero on post-fire images, so it only activates on pre-fire images.

This is the loss function that was used to calculate the final metrics for the vision transformer, which we will discuss in the next section.

Results

This network attains 98% accuracy on the test set for the per-tile classification task, with a precision of 85% and a recall of 90%. When considered on a per-image basis, the network obtains an accuracy of 89%, with a precision of 91% and a recall of 89%.

This model has the strongest results so far. Our modified loss function encourages the transformer network to optimize for precision and recall on a per-image basis. One issue with this network architecture is that we are not using a network designed to analyze sequences. While a single tower of a transformer network can hypothetically learn how to interrelate information between the stacked embeddings, it is less prone to doing so than a network specializing in sequence-based information. Ideally, we would use a network that is designed for sequence analysis, such as an LSTM or a transformer, on the per-timestep embeddings instead of simply stacking them.

In Figures 5.6 and 5.7 we see the visualized predictions of the transformer network. Note that as opposed to the Mask-RCNN predictions, which included tight bounding boxes and segmentation results, the Transformer predictions only indicate which tile, if any, of the image contains smoke. This is sufficient for our purposes, as we have a human in the loop to very the results. These images show that our network is precise (on a per-tile basis)

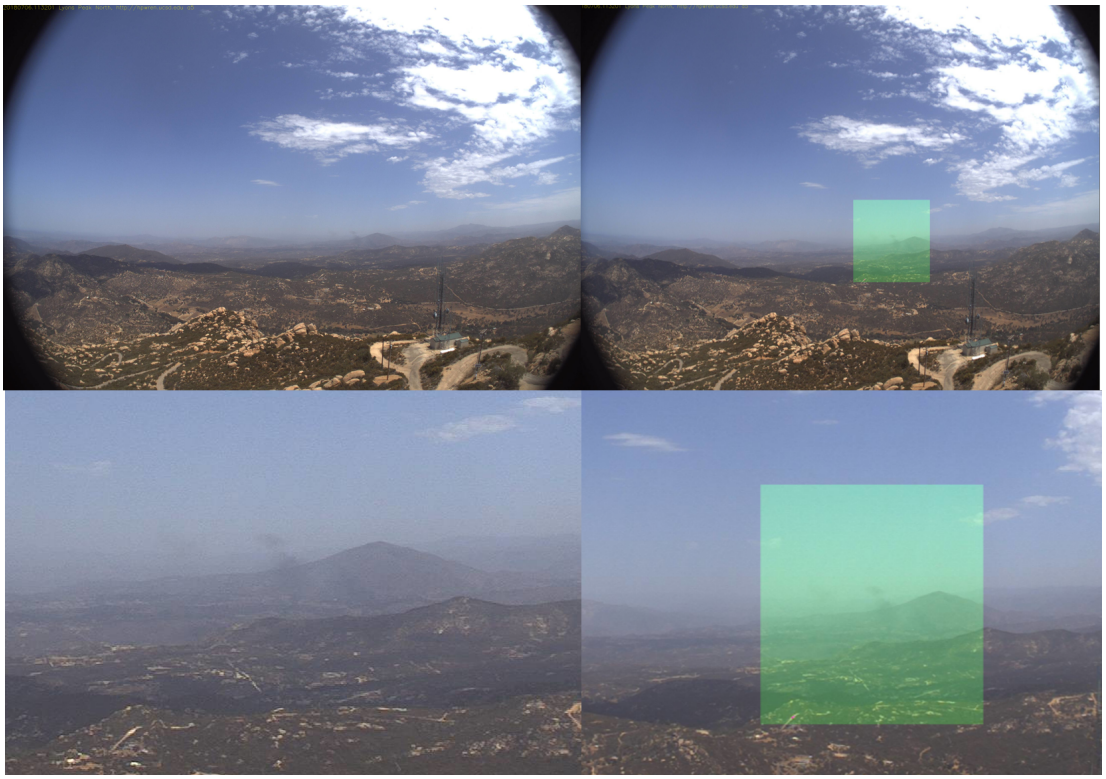

Figure 5.6: Visualized Prediction on 20180706 West lp-n-mobo-c/1530901921 +00240.

Figure 5.7: Visualized Prediction on 20180504 FIRE smer-tcs10-mobo-c/1525471979 +00180

at labeling its images, as only the tiles with smoke were labeled as such. Furthermore, they indicate how early the transformer was able to find smoke - in 5.7, the smoke is barely visible even when zoomed in, yet the transformer was able to detect it.

We can also visualize the results by plotting the number of true and false positive detections over time, in the same way we visualized the Mask-RCNN results. As we can see in Figure 5.8, the vision transformer network has less than half as many false positives as the Mask-RCNN architecture. The performance on true positives is very similar, with both able to detect fires as soon as they begin. On average, the vision transformer was able to detect a fire 43 seconds before there were any ground truth labels, and 12 seconds after it started.

Overall, this network is able to fully exploit both the spatial and temporal context available to perform well at the task of detecting smoke, and our custom loss functions align well with our desired goal of finding smoke as early as possible while minimizing false positives.

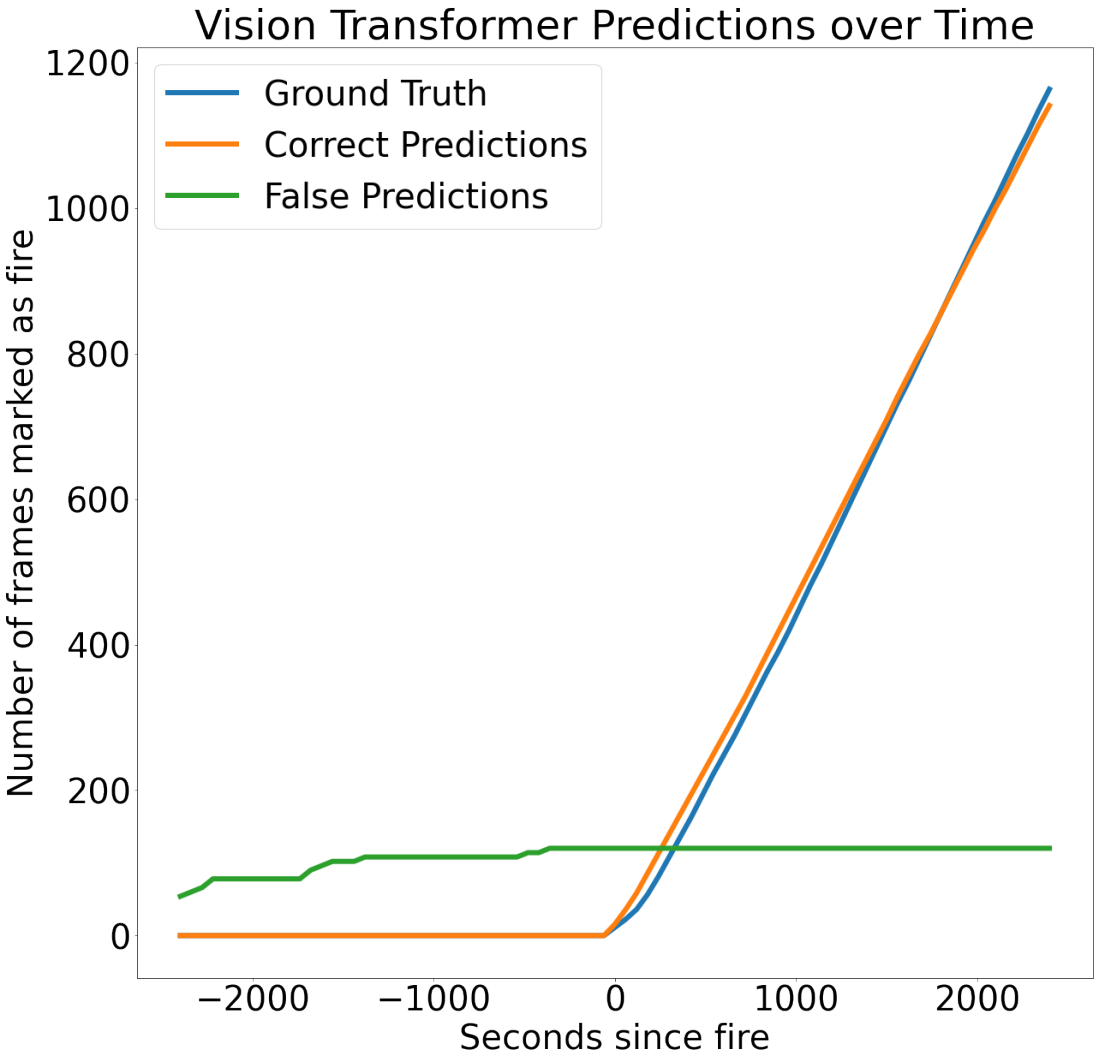

Figure 5.8: Seconds since fire versus number of frames detected.

5.4.4 Temporal Transformer on Tiled Input

Architecture

One of the issues with the Vision Transformer network shown above is that it is not designed for sequence inputs - we have to stack the timestep embeddings together, which a transformer network is not necessarily designed for. Furthermore, the vision transformer lacks a sense of locality - each tile can communicate with every other tile, and while this may help tiles gain a more global context it may also create too much noise for each tile. To remedy both of these situations and see the extent to which they affect network performance, we propose a temporal transformer on the tiled input. One key idea in our architecture is that we will provide each tile two pieces of information - its own embedding and the average embedding of its neighbors. To create the average embedding of its neighbors, we use a matrix that is designed to emulate a 3 by 3 average convolution operation (one that ignores the value at the center and any value that is out of bounds). An example of this matrix, which we can label M, for a 3 by 3 tile is in Table 5.3. Subtable a shows what the first 5 rows look when visualized as 3 by 3 tiles, while subtable b shows the entire table for a 3 by 3 tile.

We calculate this matrix, M, for our 9 by 12 grid. We then combine the rows and columns of the grid into a single dimension, making an embeddings matrix E of size 108 by 2048 (number of grid elements by size of embedding). We then calculate ME to get the result of one round of convolution, and M^2E to get the result of two rounds. We average these to get our "contextual embedding" - $\frac{1}{2}(ME + M^2E)$. This essentially averages the elements in a 5 by 5 neighborhood of a given element, weighting closer elements more than further away ones. The idea is inspired by the paper Simplifying Graph Convolutions [29] that uses a similar technique on graph inputs to efficiently emulate graph neural networks. These joint embeddings for each timestep - one containing the current tile and one containing the weighted average of its neighbors - are stacked together and fed as input to the transformer network. The transformer then takes as input five of these joint embeddings in a row, and outputs a binary prediction on the final embedding. Ideally, this allows the network to make better use of the temporal context while reducing the noise in the spatial context.

Table 5.3: Weighted Average Matrix Example

0	1/3	0
1/3	1/3	0
0	0	0
1/5	0	1/5
1/5	1/5	1/5
0	0	0
0	1/3	0
0	1/3	1/3
0	0	0
1/5	1/5	0
0	1/5	0
1/5	1/5	0
1/8	1/8	1/8
1/8	0	1/8
1/8	1/8	1/8
0	1/5	1/5
0	1/5	0
0	1/5	1/5

(a) First five rows visualized

0	1/3	0	1/3	1/3	0	0	0	0
1/5	0	1/5	1/5	1/5	1/5	0	0	0
0	1/3	0	0	1/3	1/3	0	0	0
1/5	1/5	0	0	1/5	0	1/5	1/5	0
1/8	1/8	1/8	1/8	0	1/8	1/8	1/8	1/8
0	1/5	1/5	0	1/5	0	0	1/5	1/5
0	0	0	1/3	1/3	0	0	1/3	0
0	0	0	1/5	1/5	1/5	1/5	0	1/5
0	0	0	0	1/3	1/3	0	1/3	0

(b) Table for 3 by 3 grid

Results

Unfortunately, this architecture does not perform as well as the vision transformer. It had a per-tile accuracy of 97%, precision of 58% and recall of 76%, and a per-image accuracy of 82%, precision of 78% and recall of 84%. This is better than the Resnet-101 alone was doing, but not as effective as the vision transformer network. The reasons for this are likely the following: first, since there are only five previous frames used (the reason for this is that we don't need longer-term context for our predictions) we do not need a powerful sequence network for temporal context. Second, the benefit of the vision transformer is that it allows each tile to have global information about the image, which our form of "limited spatial context" does not allow for. Furthermore, through limiting the number of attention heads to only 8, we prevent each tile from having too much noise from spatial neighbors. Overall this experiment is important because it shows us that the vision transformer network is well-suited to the task and that a more powerful sequence network is likely not needed.

Chapter 6

Conclusion

In this thesis, we studied the important problem of early detection of wildfires using computer vision. We applied methods that had never previously been applied to this problem, created novel architectures and joint loss functions that have never before been used, and achieved results better than any existing literature on the topic that correlate very strongly with our desired real-world goals. Our key contributions include: being the first to study this problem from a perspective of early fire detection instead of as a binary classification problem, creating novel architectures that support joint spatial and temporal context so that we can use full-resolution images without losing information due to compartmentalization, and being the first to use a joint binary cross entropy on a per-image basis along with a weighted sensitivity and specificity on a per-tile basis to match our dual goals of early detection with low false positives. Additionally, we plan on implementing this model in the real world before the start of the Fall 2021 wildfire season, so the project will make a positive contribution to the important problem of preventing wildfires in California.

Overall, this paper makes three key contributions to the area of early object detection in high-resolution images. Our first contribution is the use of the vision transformer for high-resolution object detection with limited dataset size. The original vision transformer paper focuses on image classification, and we extend it to show that it can perform competitively on object detection tasks by predicting labels on each tile independently instead of the image as a whole. In high-resolution images where having a general idea of where an object lies is more important than its precise location, this can serve as a

Table 6.1: Cumulative Results (all results are per-image)

Network	Accuracy	Precision	Recall
6-Layer CNN Binary Image Classification	58%	67%	46%
Resnet-101 Binary Image Classification	65%	72%	53%
Mask-RCNN	80%	73%	85%
Mask-RCNN w/ LSTM	83%	78%	86%
Mask-RCNN w/ Transformer	86%	82%	87%
Resnet-101 Tiled Image Classification †	74%	68%	76%
Vision Transformer †	87%	82%	89%
Vision Transformer w/ Resnet-101 Embeddings †	**89**%	**91**%	**89**%
Sequence Transformer w/ Tiled Embeddings †	82%	78%	84%

† - Architectures that operate on tiled images

powerful alternative to standard object detection models. Furthermore, the original paper shows how the vision transformer requires large datasets and a large amount of compute to fine-tune, while our strategy of separately training a Resnet-101 on a binary classification task then using its embeddings to warm-start our vision transformer allows us to fine-tune a model with a relatively small dataset and with limited compute power. This architecture gives us state of the art results on this dataset, both in terms of accuracy and detection time.

The second contribution is the set of architectures combining a region proposal network (in our case, Mask-RCNN) with a sequence network (in our case, either LSTM or transformer) for two-stage high-resolution object detection. While there exists previous work [20][28] using LSTMs for object detection in videos, this work focuses on using a standard object detection network to generate bounding boxes and then using an LSTM to analyze a sequence of bounding boxes and refine the predictions. Our work is novel because it introduces the idea using a highly sensitive object detection network on a lower-resolution image, then feeding high-resolution crops into the sequence network for precise classification. This allows us to use the full resolution of the image without having heavy compute requirements (such as the 108 parallel models needed for tiled classification). Furthermore, our architecture is online - it always runs an object detection model on the most recent frame each time, using the previous frames for context. In contrast to other architectures that are designed for whole-sequence classification, this allows us to have the earliest detection times possible.

Our final contribution is our joint loss function that penalizes false alarms to the human-in-the-loop while encouraging both high precision and recall in detecting actual smoke in images. These combined loss functions are broadly applicable to any instance of grid-based object detection where one may want a per-image objective that is different from their per-tile objective (in our case, the precision within a grid is less important, but the precision on a per-image basis is important). Instead of considering each tile's prediction independently and applying a per-tile loss function over the predictions, we instead treat them as part of a whole, allowing us to use per-image loss functions such as specificity and recall. We also introduce the idea of using a smooth maximum function over the grid, which allows us to use any standard loss function on a whole-image basis (in our case, we use BCE loss).

Bibliography

[1] aiformankind. *Wildfire Smoke Detection Research*. https://aiformankind.org/wildfire-smoke-detection-research/.

[2] Thomas Bachlechner, Bodhisattwa Prasad Majumder, Huanru Henry Mao, Garrison W. Cottrell, and Julian McAuley. *ReZero is All You Need: Fast Convergence at Large Depth*. 2020. arXiv: 2003.04887 [cs.LG].

[3] Olivier Barnich and Marc Droogenbroeck. "ViBe: A Universal Background Subtraction Algorithm for Video Sequences". In: *Image Processing, IEEE Transactions on* 20 (July 2011), pp. 1709–1724. DOI: 10.1109/TIP.2010.2101613.

[4] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çaglar Gülçehre, H. Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey R. Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. "Relational inductive biases, deep learning, and graph networks". In: *CoRR* abs/1806.01261 (2018). arXiv: 1806.01261. URL: http://arxiv.org/abs/1806.01261.

[5] Léon Bottou. "Online Algorithms and Stochastic Approximations". In: *Online Learning and Neural Networks*. Ed. by David Saad. revised, oct 2012. Cambridge, UK: Cambridge University Press, 1998. URL: http://leon.bottou.org/papers/bottou-98x.

[6] Y. Cao, F. Yang, Q. Tang, and X. Lu. "An Attention Enhanced Bidirectional LSTM for Early Forest Fire Smoke Recognition". In: *IEEE Access* 7 (2019), pp. 154732–154742. DOI: 10.1109/ACCESS.2019.2946712.

[7] Lee R. Dice. "Measures of the Amount of Ecologic Association Between Species". In: *Ecology* 26.3 (1945), pp. 297–302. ISSN: 00129658, 19399170. URL: http://www.jstor.org/stable/1932409.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[9] Gunnar Farnebäck. "Two-Frame Motion Estimation Based on Polynomial Expansion". In: *Image Analysis*. Ed. by Josef Bigun and Tomas Gustavsson. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 363–370. ISBN: 978-3-540-45103-7.

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. "Mask R-CNN". In: *CoRR* abs/1703.06870 (2017). arXiv: 1703.06870. URL: http://arxiv.org/abs/1703.06870.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[12] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[13] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 1097–1105.

[15] S. Kullback and R. A. Leibler. "On Information and Sufficiency". In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86. DOI: 10.1214/aoms/1177729694. URL: https://doi.org/10.1214/aoms/1177729694.

[16] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. "Object Recognition with Gradient-Based Learning". In: *Shape, Contour and Grouping in Computer Vision*. Berlin, Heidelberg: Springer-Verlag, 1999, p. 319. ISBN: 3540667229.

[17] X. Li, Z. Chen, Q. M. J. Wu, and C. Liu. "3D Parallel Fully Convolutional Networks for Real-Time Video Wildfire Smoke Detection". In: *IEEE Transactions on Circuits and Systems for Video Technology* 30.1 (2020), pp. 89–103. DOI: 10.1109/TCSVT.2018.2889193.

[18] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV].

[19] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. "SSD: Single Shot MultiBox Detector". In: *Lecture Notes in Computer Science* (2016), pp. 21–37. ISSN: 1611-3349. DOI: 10.1007/978-3-319-46448-0_2. URL: http://dx.doi.org/10.1007/978-3-319-46448-0_2.

[20] Yongyi Lu, Cewu Lu, and Chi-Keung Tang. "Online Video Object Detection Using Association LSTM". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2363–2371. DOI: 10.1109/ICCV.2017.257.

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. *Playing Atari with Deep Reinforcement Learning*. 2013. arXiv: 1312.5602 [cs.LG].

[22] Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. 1st ed. Springer Publishing Company, Incorporated, 2014. ISBN: 1461346916.

[23] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-

pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[24] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *CoRR* abs/1506.01497 (2015). arXiv: 1506.01497. URL: http://arxiv.org/abs/1506.01497.

[25] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV].

[26] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning Representations by Back-Propagating Errors". In: *Neurocomputing: Foundations of Research*. Cambridge, MA, USA: MIT Press, 1988, pp. 696–699. ISBN: 0262010976.

[27] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[28] Xiao Wang, Xiaohua Xie, and Jianhuang Lai. "Convolutional LSTM Based Video Object Detection". In: *Pattern Recognition and Computer Vision*. Ed. by Jian-Huang Lai, Cheng-Lin Liu, Xilin Chen, Jie Zhou, Tieniu Tan, Nanning Zheng, and Hongbin Zha. Cham: Springer International Publishing, 2018, pp. 99–109. ISBN: 978-3-030-03335-4.

[29] Felix Wu, Tianyi Zhang, Amauri H. Souza Jr., Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. "Simplifying Graph Convolutional Networks". In: *CoRR* abs/1902.07153 (2019). arXiv: 1902.07153. URL: http://arxiv.org/abs/1902.07153.

[30] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. *Detectron2*. https://github.com/facebookresearch/detectron2. 2019.

[31] Feiniu Yuan, Lin Zhang, Xue Xia, Boyang Wan, Qinghua Huang, and Xuelong Li. "Deep smoke segmentation". In: *Neurocomputing* 357 (2019), pp. 248–260. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2019.05.011. URL: http://www.sciencedirect.com/science/article/pii/S0925231219306435.

[32] Jie Yuan, Lidong Wang, Peng Wu, Chao Gao, and Lingqing Sun. "Detection of Wildfires along Transmission Lines Using Deep Time and Space Features". In: *Pattern Recognition and Image Analysis* 28.4 (Dec. 2018), pp. 805–812. DOI: 10.1134/s1054661818040168.

[33] Yi Zeng, Pingping Zhang, Jianming Zhang, Zhe L. Lin, and Huchuan Lu. "Towards High-Resolution Salient Object Detection". In: *CoRR* abs/1908.07274 (2019). arXiv: 1908.07274. URL: http://arxiv.org/abs/1908.07274.

[34] Qi-xing Zhang, Gao-hua Lin, Yong-ming Zhang, Gao Xu, and Jin-jun Wang. "Wildland Forest Fire Smoke Detection Based on Faster R-CNN using Synthetic Smoke Images". In: *Procedia Engineering* 211 (2018). 2017 8th International Conference on Fire Science and Fire Protection Engineering (ICFSFPE 2017), pp. 441–446. ISSN: 1877-7058. DOI: https://doi.org/10.1016/j.proeng.2017.12.034. URL: http://www.sciencedirect.com/science/article/pii/S1877705817362574.