2022-12-14

Wildfire Modeling with Data Assimilation

Andrew Johnston
Brigham Young University

Wildfire Modeling with Data Assimilation

Andrew Johnston

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Blake Barker, Chair
Emily Evans
Jared Whitehead

Department of Mathematics

Brigham Young University

ABSTRACT

Wildfire Modeling with Data Assimilation

Andrew Johnston
Department of Mathematics, BYU
Master of Science

Wildfire modeling is a complex, computationally costly endeavor, but with droughts worsening and fires burning across the western United States, obtaining accurate wildfire predictions is more important than ever. In this paper, we present a novel approach to wildfire modeling using data assimiliation. We model wildfire spread with a modification of the partial differential equation model described by Mandel et al. in their 2008 paper [1]. Specifically, we replace some constant parameter values with geospatial functions of fuel type. We combine deep learning and remote sensing to obtain real-time data for the model and employ the Nelder-Mead method to recover optimal model parameters with data assimilation [2]. We demonstrate the efficacy of this approach on computer-generated fires, as well as real fire data from the 2021 Dixie Fire in California. On generated fires, this approach resulted in an average Jaccard index of 0.996 between the predicted and actual fire perimeters and an average Kulczynski measure of 0.997. On data from the Dixie Fire, the average Jaccard index achieved was 0.48, and the average Kulczynski measure was 0.66.

Keywords: wildfire model, fire perimeter, data assimilation, partial differential equations, image segmentation, finite difference, finite element, remote sensing

Acknowledgements

I would like to thank my advisors, Blake Barker and Emily Evans, for their support and guidance throughout this experience. Their encouragement led me to pursue difficult and interesting questions, and their patience and expertise were invaluable as we sought solutions. Thank you to Jared Whitehead for being an excellent instructor in addition to agreeing to be on my advisory committee and always being willing to field my questions. Thank you to Tyler Jones for being a good friend and a patient sounding board for a lot of bad ideas. Lastly, I would like to thank my family and friends for supporting me throughout my education and making life better.

Contents

List of Tables

List of Figures

Chapter 1. Introduction

The United States is in the midst of one of the worst megadroughts in history; this period of aridity is estimated to rival any period over the last 1,200 years [3]. With this extended period of drought comes an increased wildfire risk across the country and much of the globe. In recent years, climate change, irresponsible fire safety on the individual level, and a buildup of fuel have exacerbated this risk, leading to wildfires taking more lives and destroying more property and natural resources than ever previously recorded [4, 5]. This danger has inspired government agencies on the local and federal levels to pour in time and resources toward preparation for and mitigation of wildfire hazards.

Government agencies employ firefighters at the municipal and county level, support regional mutual aid organizations, and fund state agencies across the country, such as the California Department of Forestry. At the federal level, the National Interagency Fire Center coordinates wildfire response in cooperation with the U.S. National Weather Service, and other agencies within the Department of the Interior, the Department of Agriculture, the Department of Homeland Security, and the Department of Commerce. In extreme cases, even the United States Army may become involved, providing fire squadrons to help fight larger fires.

These fire management teams can greatly benefit from accurate, real-time models for wildfire spread. An understanding of where the fire perimeter will be in coming hours or days informs decisions regarding evacuation of danger zones, allocation of resources for optimally restricting the spread of fire, and strategic prioritization for mitigating damage to property and resources.

In this paper we present a novel approach to wildfire modeling with data assimilation. The following sections in this chapter further illustrate the current wildfire risk landscape, discuss challenges inherent to wildfire modeling, and summarize our approach, highlighting new contributions to the field. In chapter 2, we discuss the state of the art in wildfire

modeling with the FARSITE system, where our approach differs from it, and a partial differential equation (PDE) model by Mandel et al. [6, 1]. Chapter 3 presents numerical methods for approximating the solutions to the PDE model. In chapter 4, we discuss a novel approach to fuel mapping with deep learning. We use these fuel estimates and other sources to perform data assimilation and improve our model, as described in chapter 5. In chapter 6, we evaluate our model, and then we conclude in chapter 7 with final remarks and ideas for future work and improvements.

1.1 The Current Wildfire Risk Landscape

In 2021, 83% of the area of the Western United States was classified as being in a state of moderate or worse drought conditions [3]. That's an estimated 57 million people living in drought conditions which drastically increase the risk of widespread, high-intensity wildfires [3, 4]. In the 20th century, aggressive suppression efforts reduced wildfire size and intensity, but caused a buildup of fuels that has resulted in larger wildfires in the 21st century [5].

Fires that burn a minimum of 1,000 acres are considered "large fires." From 1970 to 2015, the average number of large fires burning each year more than tripled in the Western U.S., with a more than sixfold increase in acreage burnt [7]. In 2020, over 13 million acres burned and over 3 billion dollars were spent in fire suppression efforts alone [8]. Wildfires have burned over 10 million acres during several years since 2015. As of November 16th, we have seen 9.7 million acres burnt in 2022, resulting in over 3 billion dollars in suppression costs and 3,796 structures destroyed [9]. The year 2022 has had the most active wildfire season in over a decade, as measured by the number of recorded wildfires in the YTD report in early November [10].

Of particular note in the United States are the wildfires in California. A national wildfire risk assessment identified the geographic regions at greatest risk of wildfire to be California (50.22% risk) and the Southern Area (15.53% risk) [11]. Since the time of the report, the California wildfire risk has only increased. A 2019 study found that from 1972 to 2018,

the average area burned each year increased fivefold, while the area burned by summer fires increased eightfold [12]. The largest 7 fires in California history have happened since July 2018, six of which occurred in 2020 or 2021 [13]. In 2020, a series of particularly large wildfires burned across California, Oregon, and Washington. They were described as an "unprecedented, climate change-fueled event"[14]. The largest fire in California history, the August Complex Fire, burned in 2020 and was estimated to have burned over 1 million acres across seven counties [8]. The fires destroyed over 10,000 structures and cost over $12.079 billion in damages, including over $10 billion in property damage and $2.079 billion in fire suppression costs [8, 15].

This trend continued in 2021 where we saw record breaking fires. The Dixie Fire was the second largest fire (behind August Complex) and was chosen for our study because it's the largest single (non-complex) wildfire in recorded California history, burning 963,309 Acres over 5 counties during 103 days [16, 17]. In Section 6.2, we demonstrate the efficacy of our wildfire modeling approach on data from the Dixie Fire. It is particularly suitable as a demo due to its long burn time, singular source, and gigantic area. Performing the same analysis on any other wildfire in the United States can reasonably be expected to take less time, be less computationally intensive, and have a higher resolution with more accurate predictions.

1.2 Challenges in Wildfire Modeling

While the benefits of accurate wildfire modeling are evident and the problem has been studied for decades, it remains incredibly challenging to obtain accurate fire predictions due to the complexity of the problem, the cost and difficulty of securing high-quality data, and the speed with which that data changes according to climate, season, or man-made infrastructure developments.

Wildfire dynamics include surface fire spread (fire along the surface of the earth), crown fire spread (fire spreading across the tops of tall vegetation), and short and long-range spotting (in which firebrands are carried by the wind to ignite another area). These different

mechanisms are often modeled by a series of coupled empirical or PDE models. Additionally, wildfire spread is affected by fuel type, volume, and density, live and dead fuel moisture, surrounding atmosphere, including wind and precipitation, and surface topography.

The systems required to accurately model all these factors are computationally costly, but the greatest hindrance in modeling wildfires is access to a comprehensive dataset. The Landfire Program aims to assemble a variety of data resources nationwide for modeling and responding to wildfires; these include maps of vegetation, operational roads for transportation, topography, fuel characteristics, and historical disturbances [18]. This data is assembled through a combination of remote sensing and grounded research teams. It is costly to obtain and only select parts of the country are updated frequently. The data may be months or years old, and when it comes to what fuel is present in an area, for example, the difference of a few months can mean drastic changes. The work in chapter 4 aims to address this problem and demonstrates success in obtaining real-time fuel type estimation.

1.3 Our Contributions

Novel contributions in this work include a modification of the PDE model presented by Mandel et al. in 2008 in which we replace constant values with functions dependent on geospatial fuel distributions [1]. Accompanying our changes is code to solve the model using a Crank-Nicolson finite difference scheme in two dimensions or a finite element method using triangles implemented in Python with the FEniCS library [19, 20, 21, 22]. This is detailed in chapter 3, with the code available on GitHub at `https://github.com/drewjohnston13/wildfire_modeling` [23]. In chapter 4, we present a deep learning approach to real-time fuel classification, an application of the U-Net architecture that, to our knowledge, has never been used in this context [24]. These fuel classifications are combined with weather data and fire perimeter data for data assimilation in parameter recovery through the Nelder-Mead algorithm, as described in chapter 5 [2]. While data assimilation methods, such as ensemble Kalman filters, have been previously employed in wildfire modeling, our approach using a

simplex search algorithm for parameter recovery is, too our knowledge, unique for wildfire modeling.

Chapter 2. Background

The current state of the art in wildfire modeling is a system called FARSITE developed by Finney et al. at the Missoula Fire Sciences Laboratory [6]. The system is used by the USDA Forest Service under the U.S. Department of Agriculture and is currently available as part of the desktop application FlamMap. The complexity of this system leads to heavy computational costs that require considerable time and computing resources to use. In this chapter we summarize the FARSITE system as well as explain a simpler model by Mandel et al. that sacrifices some aspects of modeling fire growth for lower complexity [1]. It is this model by Mandel et al. that we build upon for the rest of the paper.

2.1 FARSITE

FARSITE is a fire growth simulation modeling system that combines weather and wind data with topography and fuel maps to simulate fire growth in 2 dimensions. It incorporates existing models for spread of surface fires and crown fires, spotting, and fire acceleration in the presence of a variety of terrains, weathers, and fuels.

FARSITE receives as inputs spatial information regarding topography and fuels and outputs potential fire behavior characteristics and environmental conditions over a given landscape. Examples of fire behavior characteristics include spread rate, flame length, crown fire activity, and environmental conditions include dead fuel moistures, mid-flame wind speeds, and solar irradiance. The system can be used to simulate air and ground suppression actions, enabling a fire management team to understand potential results of an incident response plan.

The following models are coupled together in the FARSITE system:

- Rothermel's (1972) surface fire spread model [25]

- Van Wagner's (1977) crown fire initiation model [26]
- Rothermel's (1991) crown fire spread model [27]
- Albini's (1979) spotting model [28]
- Nelson's (2000) dead fuel moisture model [29]
- Albini et al.'s (1995) post-frontal combustion model [30].

These models are all based on knowledge of physical processes and aim to describe specific physical phenomena. For a more comprehensive survey of the histories and uses of wildfire models, see Pastor et al.'s 2003 paper [31].

2.2 Model by Mandel et al. [1]

While the FARSITE model is impressive in its ability to capture the many physical processes involved in wildfire spread, this comes at the cost of speed. To obtain a faster running model, Mandel et al. developed a simplification of a variety of physical properties of fire that coalesced into two PDE's coupled together, one describing temperature, and the other describing the relative ratio of remaining fuel. This combustion model, while simple, has been shown to produce solutions with traveling combustion waves, which are of paramount importance when describing wildfire spread. These traveling waves are described by Mandel et al. as "a propagating area of localized combustion made up of the preheated area ahead of the fire, the combustion zone, and the post-frontal burning region" [1]. This PDE is relatively simple to compute, but complex in its nonlinear behavior. For these reasons, we build our approach in this work with the below PDE system at the center.

The following definitions are from Section 2 of Mandel et al.'s paper [1].

> We consider the model of fire in a layer just above the ground. First, we define the following terms:

T (K) is the temperature of the fire layer,

$S \in [0, 1]$ is the fuel supply mass fraction (the relative amount of fuel remaining),

k $(m^2 s^{-1})$ is the thermal diffusivity,

A $(K s^{-1})$ is the temperature rise per second at the maximum burning rate with full initial fuel load and no cooling present,

B (K) is the proportionality coefficient in the modified Arrhenius law,

C (K^{-1}) is the scaled coefficient of the heat transfer to the environment,

C_S (s^{-1}) is the fuel relative disappearance rate,

T_a (K) is the ambient temperature, and

$\vec{v}$ $(m s^{-1})$ is the wind speed given by atmospheric data or model.

The coupled PDE model from Mandel et al. is then derived from principles of conservation of energy, fuel reaction rate from the Arrhenius law, and fuel supply balance:

$$\frac{dT}{dt} = \nabla \cdot (k \nabla T) - \vec{v} \cdot \nabla T + A \left(S e^{-B/(T - T_a)} - C(T - T_a) \right), \tag{2.1}$$

$$\frac{dS}{dt} = -C_S S e^{-B/(T - T_a)}, \quad T > T_a \tag{2.2}$$

with the initial values

$$S(t_0) = 1 \text{ and } T(t_0) = T_0. \tag{2.3}$$

The diffusion term $\nabla \cdot (k \nabla)$ describes heat transfer by radiation, $\vec{v} \cdot \nabla T$ is an advection term that models temperature change due to wind, $S e^{-B/(T - T_a)}$ is the rate at which fuel is consumed when burning, and $AC(T - T_a)$ describes the change in temperature due to heat transfer to the atmosphere. The exponential term $e^{-B/(T - T_a)}$ is from the reaction rate $e^{-B/T}$ from the Arrhenius law, modified so that no reaction occurs when the local temperature matches ambient temperature.

For a more detailed look at the derivation of the model from physical principles, see section 4 of their paper [1].

Mandel et al. also make use of data assimilation to improve performance of their model; they utilize an ensemble Kalman filter to modify temperature and fuel state outputs with the intention to later couple the model with another PDE describing the local atmospheric changes due to the fire [1]. In our data assimilation approach, rather than state updates, we use a simplex search algorithm for parameter recovery, as described in chapter 5.

Chapter 3. Numerical Methods for Solving the PDE Model

As we cannot solve the PDE analytically, we seek a numerical approximation to the solution of the PDE. First, we explore the Crank-Nicolson finite difference scheme. In the second section, we discuss an approach with a finite element method.

3.1 Crank-Nicolson Method

The Crank-Nicolson method is an implicit method based on the trapezoidal rule, giving second-order convergence in time. It was selected because it is unconditionally stable for diffusion equations, provides second-order convergence, and can be solved efficiently with tri-diagonal or band-diagonal matrix algorithms.

First, we show a derivation and demonstration of the Crank-Nicolson finite difference scheme in one dimension. After we've established that our code is working as expected, we extend to the 2-dimensional case.

3.1.1 1-Dimensional Case.

In one dimension we have

$$T_t = kT_{xx} - vT_x + A\left(Se^{-B/(T-T_a)} - C(T-T_a)\right)$$
$$S_t = -C_S Se^{-B/(T-T_a)}, \;\; T > T_a$$

with the initial values

$$S(t_0) = 1 \text{ and } T(t_0) = T_0$$

Let T_i^n be the value of T evaluated at the point $i\Delta x$ on a line segment at time $t_0 + n\Delta t$. Then the Crank-Nicolson scheme gives

$$\begin{aligned}
S_t &= \frac{S_i^{n+1} - S_i^n}{\Delta t} \\
T_t &= \frac{T_i^{n+1} - T_i^n}{\Delta t} \\
T_x &= \frac{1}{2}\left[\frac{1}{2\Delta x}\left(T_{i+1}^n - T_{i-1}^n + T_{i+1}^{n+1} - T_{i-1}^{n+1}\right)\right] \\
T_{xx} &= \frac{1}{2}\left[\frac{1}{\Delta x^2}\left(T_{i-1}^n - 2T_i^n + T_{i+1}^n + T_{i-1}^{n+1} - 2T_i^{n+1} + T_{i+1}^{n+1}\right)\right].
\end{aligned}$$

Then this gives

$$\begin{aligned}
\frac{T_i^{n+1} - T_i^n}{\Delta t} &= \frac{k}{2\Delta x^2}\left(T_{i-1}^n - 2T_i^n + T_{i+1}^n + T_{i-1}^{n+1} - 2T_i^{n+1} + T_{i+1}^{n+1}\right) \\
&\quad - \frac{v}{4\Delta x}\left(T_{i+1}^n - T_{i-1}^n + T_{i+1}^{n+1} - T_{i-1}^{n+1}\right) \\
&\quad + A\left(S_i^n e^{-B/(T_i^n - T_a)} - C(T_i^n - T_a)\right) \\
\frac{S_i^{n+1} - S_i^n}{\Delta t} &= -C_S S_i^n e^{-B/(T_i^n - T_a)}.
\end{aligned}$$

or

$$\begin{aligned}
T_i^{n+1} &= T_i^n + \frac{k\Delta t}{2\Delta x^2}\left(T_{i-1}^n - 2T_i^n + T_{i+1}^n + T_{i-1}^{n+1} - 2T_i^{n+1} + T_{i+1}^{n+1}\right) \\
&\quad - \frac{v\Delta t}{4\Delta x}\left(T_{i+1}^n - T_{i-1}^n + T_{i+1}^{n+1} - T_{i-1}^{n+1}\right) \\
&\quad + \Delta t A\left(S_i^n e^{-B/(T_i^n - T_a)} - C(T_i^n - T_a)\right) \\
S_i^{n+1} &= S_i^n - \Delta t C_S S_i^n e^{-B/(T_i^n - T_a)}.
\end{aligned}$$

We make use of array splicing so that the finite difference scheme can be solved on all points of our discretization at once with matrices in a linear solver. For this demonstration, we have used Dirichlet boundary conditions for T that we set to be the value of ambient temperature, T_a, and numerical boundary conditions for S that result in zero flux across the boundary. See the appendix for selections from the code used in this paper, or visit the GitHub repository for this project to view all relevant code [23].

In Figure 3.1, we use the following values to test our solution on the line segment $[0, 10]$: $k = 1$, $A = 10000$, $B = 300$, $C = \frac{1}{5A}$, $C_s = 2$, $v = 0$, and $T_a = 300$. To create a smooth spike in temperature around $x = 4$, we use the initial condition:

$$T_0(x) = T_a + \frac{1200}{\cosh(10(x-4))}. \tag{3.1}$$

To initialize a hill of fuel about $x = 5$ we set

$$S_0(x) = \begin{cases} 2\exp\left(\left(\left(\frac{x-5}{2}\right)^2 - 1\right)^{-1}\right) & \text{if } x \in [3, 7] \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

Under these conditions we can expect the fire to move toward the center of the fuel at $x = 5$, increase in temperature until the fuel is burned away, and then die out, as displayed in Figure 3.1.

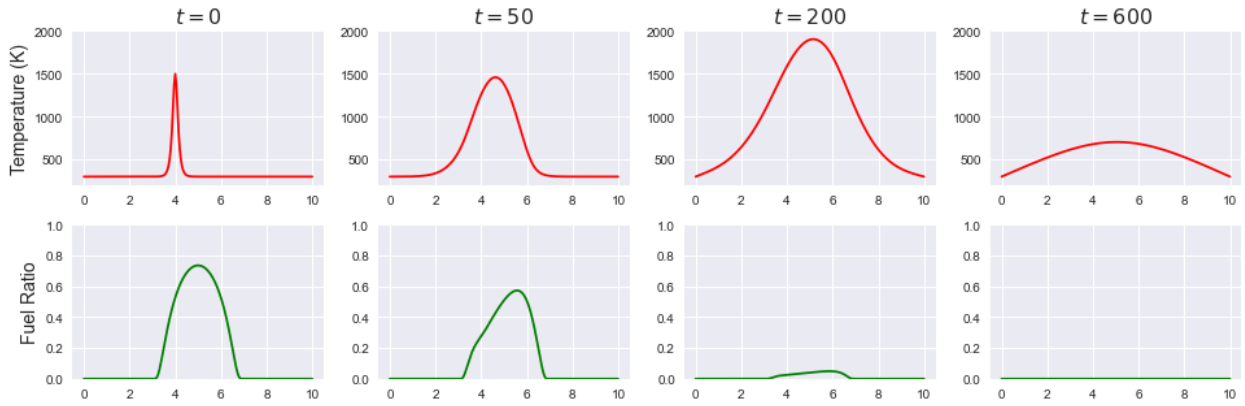

Figure 3.1: 1-Dimensional Crank-Nicolson Example (No Wind Present)

To test our advection term, we now add wind to our previous conditions by setting $v = 2$. This results in the fire being pushed to the right, as shown in Figure 3.2.

We now extend to the 2-dimensional case.

3.1.2 2-Dimensional Case.

In two dimensions we have

$$\begin{aligned} T_t &= \nabla \cdot (k\nabla T) - \vec{v} \cdot \nabla T + A\left(Se^{-B/(T-T_a)} - C(T - T_a)\right) \\ &= k(T_{xx} + T_{yy}) - (v_1 T_x + v_2 T_y) + A\left(Se^{-B/(T-T_a)} - C(T - T_a)\right), \\ S_t &= -C_S S e^{-B/(T-T_a)}, \quad T > T_a \end{aligned}$$

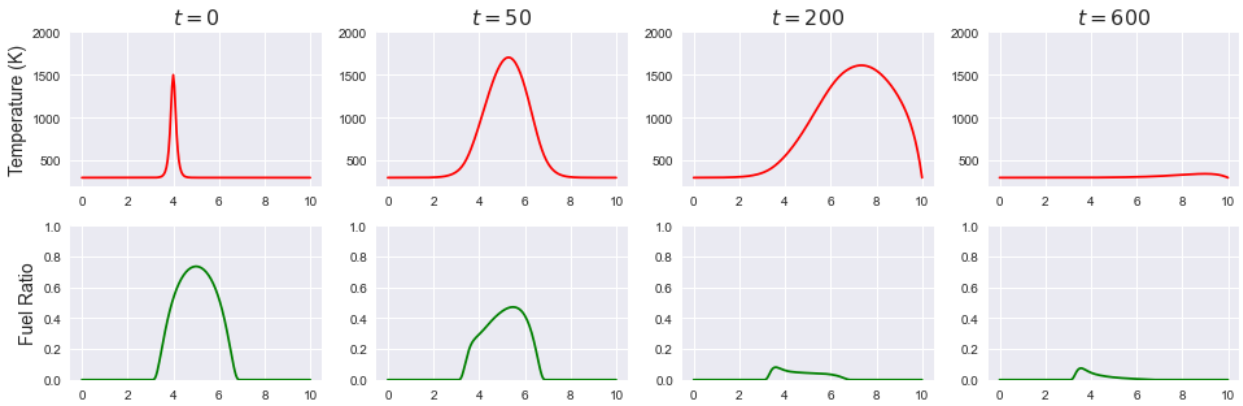

Figure 3.2: 1-Dimensional Crank-Nicolson Example with Wind

with the initial values

$$S(t_0) = 1 \text{ and } T(t_0) = T_0.$$

Let $T^n_{i,j}$ be the value of T evaluated at the point $(i\Delta x, j\Delta y)$ in our plane at time $t_0 + n\Delta t$. Then the Crank-Nicolson scheme gives

$$\begin{aligned}
S_t &= \frac{S^{n+1}_{i,j} - S^n_{i,j}}{\Delta t} \\
T_t &= \frac{T^{n+1}_{i,j} - T^n_{i,j}}{\Delta t} \\
T_x &= \frac{1}{2}\left[\frac{1}{2\Delta x}\left(T^n_{i+1,j} - T^n_{i-1,j} + T^{n+1}_{i+1,j} - T^{n+1}_{i-1,j}\right)\right] \\
T_y &= \frac{1}{2}\left[\frac{1}{2\Delta y}\left(T^n_{i,j+1} - T^n_{i,j-1} + T^{n+1}_{i,j+1} - T^{n+1}_{i,j-1}\right)\right] \\
T_{xx} &= \frac{1}{2}\left[\frac{1}{\Delta x^2}\left(T^n_{i-1,j} - 2T^n_{i,j} + T^n_{i+1,j} + T^{n+1}_{i-1,j} - 2T^{n+1}_{i,j} + T^{n+1}_{i+1,j}\right)\right] \\
T_{yy} &= \frac{1}{2}\left[\frac{1}{\Delta y^2}\left(T^n_{i,j-1} - 2T^n_{i,j} + T^n_{i,j+1} + T^{n+1}_{i,j-1} - 2T^{n+1}_{i,j} + T^{n+1}_{i,j+1}\right)\right].
\end{aligned}$$

Then for $\vec{v} = (v_1, v_2)$ we have

$$\begin{aligned}
\frac{T_{i,j}^{n+1}-T_{i,j}^{n}}{\Delta t} &= \frac{k}{2}\Bigg[\frac{1}{\Delta x^2}\left(T_{i-1,j}^{n}-2T_{i,j}^{n}+T_{i+1,j}^{n}+T_{i-1,j}^{n+1}-2T_{i,j}^{n+1}+T_{i+1,j}^{n+1}\right)\\
&\quad+\frac{1}{\Delta y^2}\left(T_{i,j-1}^{n}-2T_{i,j}^{n}+T_{i,j+1}^{n}+T_{i,j-1}^{n+1}-2T_{i,j}^{n+1}+T_{i,j+1}^{n+1}\right)\Bigg]\\
&\quad-\frac{v_1}{4\Delta x}\left(T_{i+1,j}^{n}-T_{i-1,j}^{n}+T_{i+1,j}^{n+1}-T_{i-1,j}^{n+1}\right)\\
&\quad-\frac{v_2}{4\Delta y}\left(T_{i,j+1}^{n}-T_{i,j-1}^{n}+T_{i,j+1}^{n+1}-T_{i,j-1}^{n+1}\right)\\
&\quad+A\left(S_{i,j}^{n}e^{-B/(T_{i,j}^{n}-T_a)}-C(T_{i,j}^{n}-T_a)\right)\\
\frac{S_{i,j}^{n+1}-S_{i,j}^{n}}{\Delta t} &= -C_S S_{i,j}^{n}e^{-B/(T_{i,j}^{n}-T_a)}.
\end{aligned}$$

We assume a square grid with $\Delta x = \Delta y$. Then we have

$$\begin{aligned}
\frac{T_{i,j}^{n+1}-T_{i,j}^{n}}{\Delta t} &= \frac{k}{2\Delta x^2}\big(T_{i-1,j}^{n}+T_{i,j-1}^{n}-4T_{i,j}^{n}+T_{i,j+1}^{n}+T_{i+1,j}^{n}\\
&\qquad+T_{i-1,j}^{n+1}+T_{i,j-1}^{n+1}-4T_{i,j}^{n+1}+T_{i+1,j}^{n+1}+T_{i,j+1}^{n+1}\big)\\
&\quad-\frac{1}{4\Delta x}\Big(v_1\left[T_{i+1,j}^{n}-T_{i-1,j}^{n}+T_{i+1,j}^{n+1}-T_{i-1,j}^{n+1}\right]\\
&\qquad+v_2\left[T_{i,j+1}^{n}-T_{i,j-1}^{n}+T_{i,j+1}^{n+1}-T_{i,j-1}^{n+1}\right]\Big)\\
&\quad+A\left(S_{i,j}^{n}e^{-B/(T_{i,j}^{n}-T_a)}-C(T_{i,j}^{n}-T_a)\right)\\
\frac{S_{i,j}^{n+1}-S_{i,j}^{n}}{\Delta t} &= -C_S S_{i,j}^{n}e^{-B/(T_{i,j}^{n}-T_a)}
\end{aligned}$$

or

$$\begin{aligned}
T_{i,j}^{n+1} &= T_{i,j}^{n}+\frac{k\Delta t}{2\Delta x^2}\big(T_{i-1,j}^{n}+T_{i,j-1}^{n}-4T_{i,j}^{n}+T_{i,j+1}^{n}+T_{i+1,j}^{n}\\
&\qquad+T_{i-1,j}^{n+1}+T_{i,j-1}^{n+1}-4T_{i,j}^{n+1}+T_{i+1,j}^{n+1}+T_{i,j+1}^{n+1}\big)\\
&\quad-\frac{\Delta t}{4\Delta x}\Big(v_1\left[T_{i+1,j}^{n}-T_{i-1,j}^{n}+T_{i+1,j}^{n+1}-T_{i-1,j}^{n+1}\right]\\
&\qquad+v_2\left[T_{i,j+1}^{n}-T_{i,j-1}^{n}+T_{i,j+1}^{n+1}-T_{i,j-1}^{n+1}\right]\Big)\\
&\quad+\Delta t A\left(S_{i,j}^{n}e^{-B/(T_{i,j}^{n}-T_a)}-C(T_{i,j}^{n}-T_a)\right)\\
S_{i,j}^{n+1} &= S_{i,j}^{n}-\Delta t C_S S_{i,j}^{n}e^{-B/(T_{i,j}^{n}-T_a)}.
\end{aligned}$$

We flatten our matrix of values of $T_{i,j}^{n}$ into one long array so that we can consider every point in our 5-point stencil as part of a single vector for each i, j; then we define a matrix

that maps the flattened array to the finite difference scheme. Then, because our system is linear in time (the only nonlinearity in space is known at each time step), we can frame this finite difference method as a linear system of $(J-1)^2$ equations for J steps in space (not J^2 equations because the boundaries can be solved separately at each step). This is significantly faster than solving a nonlinear system, especially if we make use of sparse computing methods.

Suppose our spatial discretization has N_x points in the x direction and N_y in the y direction, and let $[x]_k$ represent the kth entry of a vector x. Our linear system then looks like:

$$
\begin{aligned}
(I-M)x^{n+1} &= b^n \\
\text{where} \quad [x^n]_{N_y i+j} &= T^n_{i,j}, \\
[Mx^n]_{N_y i+j} &= \frac{k\Delta t}{2\Delta x^2}\left(T^n_{i-1,j} + T^n_{i,j-1} - 4T^n_{i,j} + T^n_{i,j+1} + T^n_{i+1,j}\right) \\
&\quad - \frac{\Delta t}{4\Delta x}\left(v_1\left[T^n_{i+1,j} - T^n_{i-1,j}\right] + v_2\left[T^n_{i,j+1} - T^n_{i,j-1}\right]\right), \\
\text{and} \quad [b^n]_{N_y i+j} &= [(I+M)x^n]_{N_y i+j} + \Delta t A\left(S^n_{i,j} e^{-B/(T^n_{i,j}-T_a)} - C(T^n_{i,j} - T_a)\right).
\end{aligned}
$$

To view an explicit example of how to define M in this system, see the code in section A.1 of the appendix.

In our code we again make use of array splicing so that the finite difference scheme can be solved on all points of our discretization at once with matrices in a linear solver. In our 2-dimensional case, the matrices are largely sparse, so we make use of SciPy's sparse linear solvers for greater efficiency [32]. These matrices are also more difficult to define; view the code in section A.1 for details.

We use similar values to test our 2-D solution on the square $[0,10]\times[0,10]$: $k=1$, $A=10000$, $B=300$, $C=0$, $C_s=3$, $v=[0,0]$, and $T_a=0$. To create a smooth spike in temperature around the point $(4,4)$, we use the initial condition:

$$T_0(x,y) = T_a + \frac{1200}{\cosh((x-4)^2+(y-4)^2)}. \tag{3.3}$$

To initialize a hill of fuel centered at the point $(5, 5)$ we set

$$S_0(x, y) = \begin{cases} 2 \exp\left(\left(\frac{(x-5)^2+(y-5)^2}{2^2} - 1\right)^{-1}\right) & \text{if } (x, y) \in [3, 7] \times [3, 7] \\ 0 & \text{otherwise.} \end{cases} \tag{3.4}$$

Under these conditions we expect the fire to move toward the center of the fuel at $(5, 5)$, increase in temperature until the fuel is burned away, and then die out. We can view this behavior in the 3D plots of Figure 3.3. Here, the z-axis takes on the values of temperature and the remaining fuel ratio. Heat maps of these plots can be seen in Figure B.1 of the appendix, if desired.

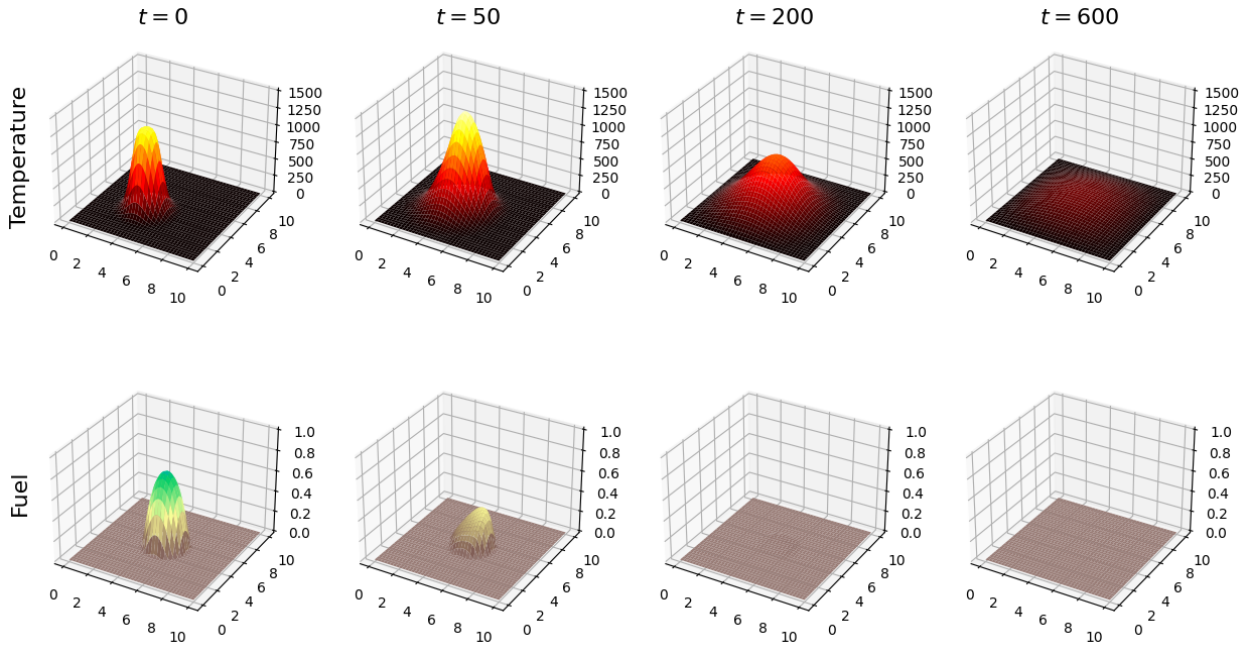

Figure 3.3: 2-Dimensional Crank-Nicolson Example (No Wind Present)

We again test our advection term by adding nonzero wind to our previous conditions. We set $v = [2, 2]$. This results in the fire being pushed in the positive x and y directions, as shown in the panels of Figure 3.4. Again, heat maps can be viewed in the appendix, Figure B.2.

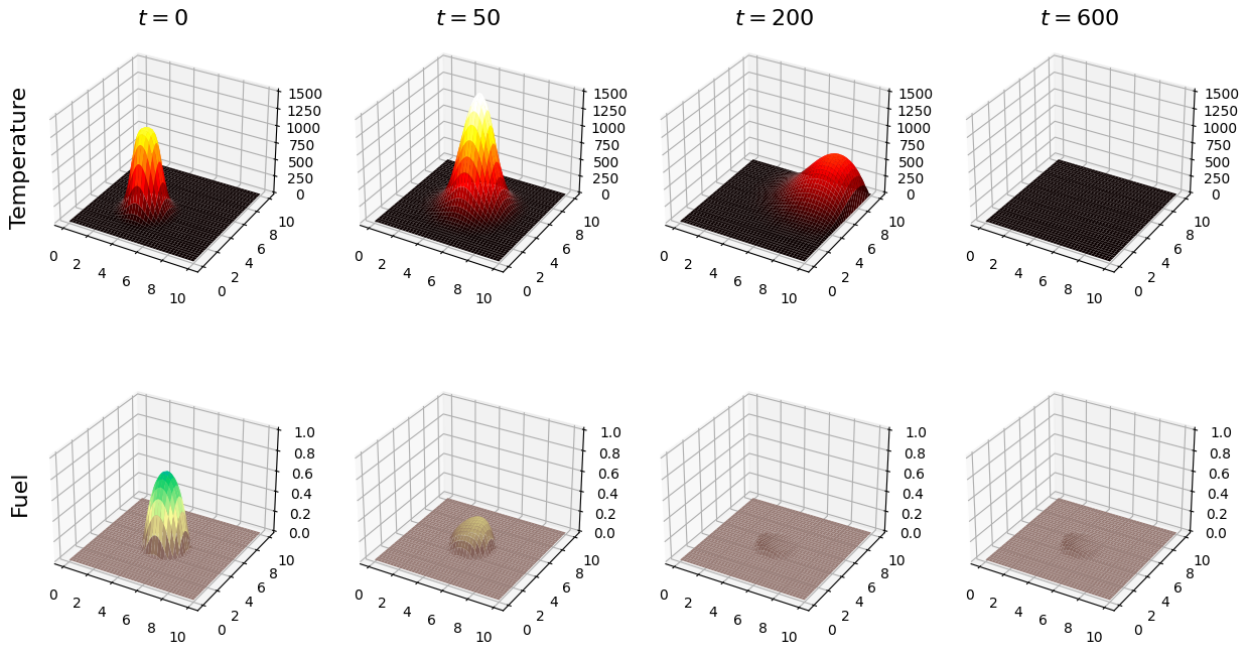

Figure 3.4: 2-Dimensional Crank-Nicolson Example with Wind

3.2 Finite Element Method

While the Crank-Nicolson method is admirable, a custom PDE solver is difficult to integrate with PyMC, a Python package for Bayesian statistical modeling and probabilistic machine learning [33]. While a Bayesian approach is not ultimately what this paper focuses on, it was long considered during the course of our research, which led us to implement a solution to the PDE in FEniCS, a Python package for solving PDE's using finite element methods [20]. FEniCS integrates well with PyMC, and has the added advantage of being easily parallelizable. FEniCS code can be adapted with a few simple changes to parallelize meshes, functions, and solvers with OpenMPI, a Message Passing Interface library [34]. This allows us to consider problems on much larger domains and solve problems considerably faster.

In order to use our finite element method, we must derive the variational form of our PDE system. Beginning with Equation 2.1, we apply a forward Euler finite difference scheme for the time derivative. Then we multiply by a test function, q, and integrate over our domain,

Ω, to get:

$$\int_\Omega \frac{(T_{n+1}-T_n)}{dt} q \, d\vec{x} = \int_\Omega \Big(k\Delta T_n q - \vec{v}\cdot\nabla T_n q + A\big(S_n e^{-B/(T_n-T_a)} q - C(T_n - T_a)q\big)\Big)\, d\vec{x}$$
$$= \int_{\partial\Omega} k\nabla T_n \cdot \hat{\mathbf{n}} q \, d\vec{s} + \int_\Omega \Big(- k\nabla T_n \cdot \nabla q - \vec{v}\cdot\nabla T_n q + A\big(S_n e^{-B/(T_n-T_a)} q$$
$$- C(T_n - T_a)q\big)\Big)\, d\vec{x}$$

by Green's first identity where $\hat{\mathbf{n}}$ is the outward pointing unit normal at each point on the boundary.

We apply pure Neumann boundary conditions, so the integral over the boundary is zero. Then, by collecting all nonzero terms on one side of the equation we get

$$\int_\Omega (T_{n+1}-T_n)q + dt\Big(k\nabla T_n\cdot\nabla q + \vec{v}\cdot\nabla T_n q - A\big(S_n e^{-B/(T_n-T_a)} q - C(T_n-T_a)q\big)\Big)\, d\vec{x} = 0$$

We perform the same process on our fuel equation with a different test function, p, and get

$$\int_\Omega \frac{(S_{n+1}-S_n)}{dt} p \, d\vec{x} = \int_\Omega -C_S S_n e^{-B/(T_n-T_a)} p \, d\vec{x}$$

which simplifies to

$$\int_\Omega (S_{n+1}-S_n)p + dt\big(C_S S_n e^{-B/(T_n-T_a)} p\big)\, d\vec{x} = 0.$$

Upon testing, this variational form results in diverging temperature values due to the instability of the forward Euler method with diffusion equations. To resolve this issue, we use a backward Euler finite difference scheme for the time derivative instead. This gives

$$\int_\Omega (T_{n+1}-T_n)q + dt\Big(k\nabla T_{n+1}\cdot\nabla q + \big(\vec{v}\cdot\nabla T_{n+1} - A\big(S e^{-B/(T_{n+1}-T_a)} - C(T_{n+1}-T_a)\big)\big)q\Big) d\vec{x} = 0$$

and

$$\int_\Omega (S_{n+1}-S_n)p + dt\big(C_S S_{n+1} e^{-B/(T-T_a)} p\big)\, d\vec{x} = 0.$$

While our variational form is now stable, we no longer have linearity in T_{n+1}. In order to avoid depending on nonlinear solvers, we linearize our nonlinear term by the following Taylor expansion:

$$e^{-B/(T_{n+1}-T_a)} = e^{\frac{-B}{T_n-T_a}} + \frac{B e^{\frac{-B}{T_n-T_a}}}{(T_n-T_a)^2}(T_{n+1}-T_n) + O(T_{n+1}^2).$$

Now that our variational form is linear, we can define it in FEniCS and leverage parallelized linear solvers such as MUMPS (MUltifrontal Massively Parallel Solver) [35, 36]. To see example code, view section A.2 of the appendix. We test our code with the same values from our 2D Crank-Nicolson example without wind, except the initial fuel is a circle centered at (5,5) instead of a hill. The result is found in Figure 3.5.

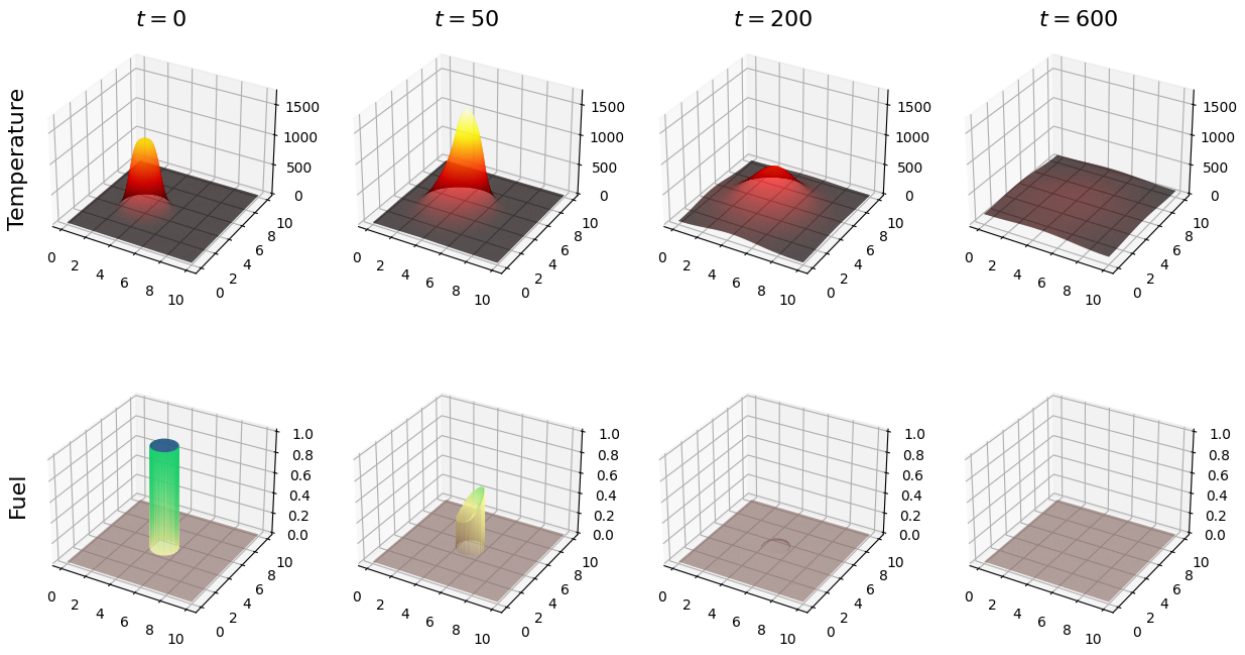

Figure 3.5: Finite Element Example Using FEniCS

Chapter 4. Image Segmentation Using Deep Learning

One vital aspect of wildfire modeling is obtaining an accurate geospatial fuel distribution map. However, current methods for doing this are costly and time consuming. These fuel maps may include information about the type of fuel present as well as the volume, density, and moisture level of the fuel. We aim to implement a pipeline for extracting fire fuel distributions from satellite images in close-to-real time. To accomplish this goal, we simplify the fuel data and focus on extracting accurate vegetation classification maps from satellite

images, which can be adapted to fuel distributions through different fire models. In this chapter we outline how we develop a novel dataset and make use of the well-known UNet deep learning architecture to achieve over 81% accuracy on 4-class vegetation classification, and over 74% accuracy with 12 classes [24]. Once we have vegetation classifications in the form of geospatial raster data, we adjust the values of our fuel-dependent parameters. These parameters are k (thermal diffusivity), A (the temperature rise per second at the maximum burning rate with full initial fuel load and no cooling present), and C_s (the fuel relative disappearance rate). Instead of constant values across our domain, these parameters become functions of geographical location with different values being selected for each vegetation class in our domain.

To develop fuel distribution maps we require accurate geospatial vegetation classification across the area of interest, so we set out to define suitable vegetation classes for fuel modeling and acquire sufficient data to train a deep learning model for image segmentation. All the models discovered in our literature review for mapping vegetation with remote sensing were either limited to binary classification (vegetation vs no vegetation), or were heavily supplemented by field sample data from ground-based teams. This is the current standard for vegetation fuel mapping, but due to restrictions on cost, man-power, and weather patterns, these vegetation maps cannot be completed everywhere with a high degree of accuracy. Even in the places where accurate surveys are possible, such surveys are infrequent and quickly become outdated as fuel distributions for fire models. For these reasons, we implement a deep learning segmentation model to generate vegetation maps for use in wildfire modeling in near-real time.

4.1 Data Collection

As we were unable to find any available datasets that fit our needs, we assemble our own dataset from publicly available databases. The European Space Agency provides public access to images from the Sentinel 2 satellite, and NASA does the same with the Landsat 8

satellite [37]. While they both provide bands in the red, green, blue, near-infrared, and short-wave infrared spectrum frequencies, Sentinel 2 has much better resolution (10m compared to 30m) for many bands, as well as a number of additional medium frequency bands that Landsat 8 does not have, called vegetation red edge bands. These bands capture reflectances from frequencies between red and near-infrared bands that have historically been used to measure vegetation health. These bands have not been used to discern between vegetation classes previously, but we set out to test whether they hold any efficacy in that regard. For those reasons, Sentinel 2 was chosen as our source for image data.

As we collected appropriate Sentinel 2 products to build a dataset, we made some simplifying assumptions about our problem to make it feasible in the time frame we had. We only considered daytime images of the western United States with under 5% cloud coverage. Additionally, we restricted our data to spring or summer months of 2020 when vegetation would be thickest and most vibrant. The most dangerous wildfires in the United States occur in this geographical area during this time of year, and cloud coverage greatly restricts the effectiveness of a majority of the bands in question. While making our model robust to other data is important in the long term, this restriction is justified for a first attempt. We identified seven suitable products of size 12,980 by 12,980 and divided each of them up into 1,296 smaller images of size 305 by 305 using Python's RasterIO and ImageIO libraries. These smaller images were stored in folders to be used as training input for our model.

To complete our dataset we used data from the 2020 Fuel Vegetation Type Continental United States raster provided by the LANDFIRE Program [38]. This program is run by the U.S. Department of Agriculture Forest Service and the U.S. Department of the Interior and provides the most accurate vegetation fuel maps in the country. While these maps are the best we have for vegetation labels to give our models, they are inappropriate for use in fire models due to how quickly vegetation data changes week to week and season to season. Nevertheless, we believe a deep learning approach can become robust enough to perform reasonably well on novel data after training on this restricted dataset.

4.2 Preprocessing

Across all images and reflectance bands, noise could be present that would bring individual pixel values outside of the expected range for that reflectance. To combat this, we clip each of the 13 spectral bands included in a Sentinel 2 product by replacing any value above the expected range with the max value. After this filter is applied, each band is normalized to have values between 0 and 1.

Throughout much of the literature in this domain, experts have made use of the Normalized Difference Vegetation Index, as evaluated by Huang et al. in their 2021 paper [39]. The NDVI for each image was calculated from the red (R) and near-infrared (NIR) bands as follows: $NDVI = \frac{NIR-R}{NIR+R}$. This index was the first data we used for image segmentation, but we wanted to test out other bands to see what kind of performance they could achieve.

A number of different band compositions were mentioned in literature for different uses. The band compositions we focused on were RGB or true color (red, green, and blue), false color (red, green, and near-infrared), SWIR (red, near-infrared, and shortwave infrared 2), Agriculture (blue, near-infrared, and shortwave infrared 1), and Geology (blue and shortwave infrared 1 and 2). The false color, Agriculture, and Geology bands have been used to monitor plant health and density. The Agriculture, Geology, and SWIR bands have shown to be useful in estimating moisture content in soil and vegetation. The SWIR band has also been used to identify snow, water, clouds, and newly burned land.

With image data preprocessed, we had only to geospatially match the images to the labels in order to train our models. We had longitude and latitude coordinates for the corners of each satellite product, and we could use RasterIO to map the LANDFIRE raster to the same coordinate reference system as the satellite products. From there, we initially tried to use traditional affine transformations to match the raster to the satellite image position, but found difficulty in getting an exact match. This was because the raster data was made as if from the perspective directly above each pixel, while the satellite image was taken from an angle by a camera at a fixed location on a satellite orbiting the earth.

In order to project the raster data onto the satellite image, we made planar assumptions about both the satellite image and the raster and calculated an appropriate homography using the coordinates for the four corners of the satellite image. To accomplish this we used OpenCV's `find_homography()` function and a variety of methods supported by RasterIO. See Figure 4.1 for an illustration of the process, in which the raster data being plotted is a map of NDVI values colored by a gradient map from red to yellow to green. First, a square containing the satellite image coordinates (blue dots on the left plot of the figure) was extracted from the raster. Then this section of the raster was cut out with a mask, and a homography was calculated taking the corners of the mask into a square. This homography was applied to the raster to get the final plot of Figure 4.1, which matched the satellite product.

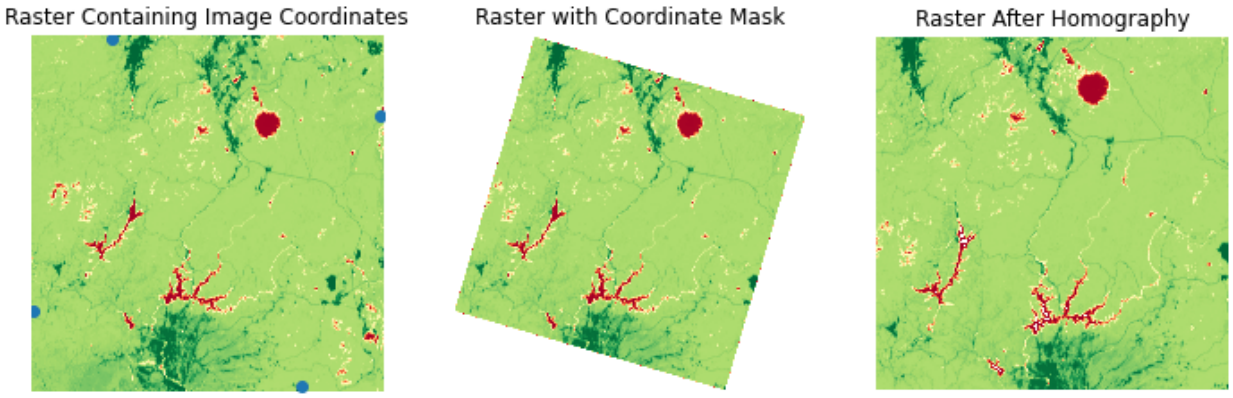

Figure 4.1: Matching raster data to satellite product

These labels contained hundreds of specific classes of vegetation. In order to simplify the segmentation problem we built a dictionary to map each specific class into more general classes of vegetation. One dictionary was built to map everything into one of four classes: Trees, Shrubs, Grasses, and Other. We used another dictionary to map the classes into 12 groups based on known differences in fire behavior across these vegetation types. The 12 classes included wet grasses, dry grasses, wet shrubs, dry shrubs, deciduous trees, evergreen trees, mixed trees, other trees, water, snow, urban area, and sparse vegetation.

4.3 UNet Model

The UNet model, described in Ronneberger et al.'s 2014 paper, is a popular and effective deep learning architecture for image segmentation [24]. We chose to implement this model in PyTorch using the convolutional, pooling, and upsampling layers as described in the original paper. We used ReLU activation functions, a cross-entropy loss function across all vegetation classes present, a learning rate of 10^{-3}, and a dropout layer with a dropout rate of 0.25 after every block in the architecture with a linear layer except the final layer. This approach provided pixel-wise probability estimates for each class which were turned to class predictions by way of softmax regression. The learning rate and dropout ratio were decided by experimentation.

4.4 Results

We began by examining results attained by training a model on the NDVI data. For the 4-class labels, we produced accuracy and loss graphs shown in Figure 4.2.

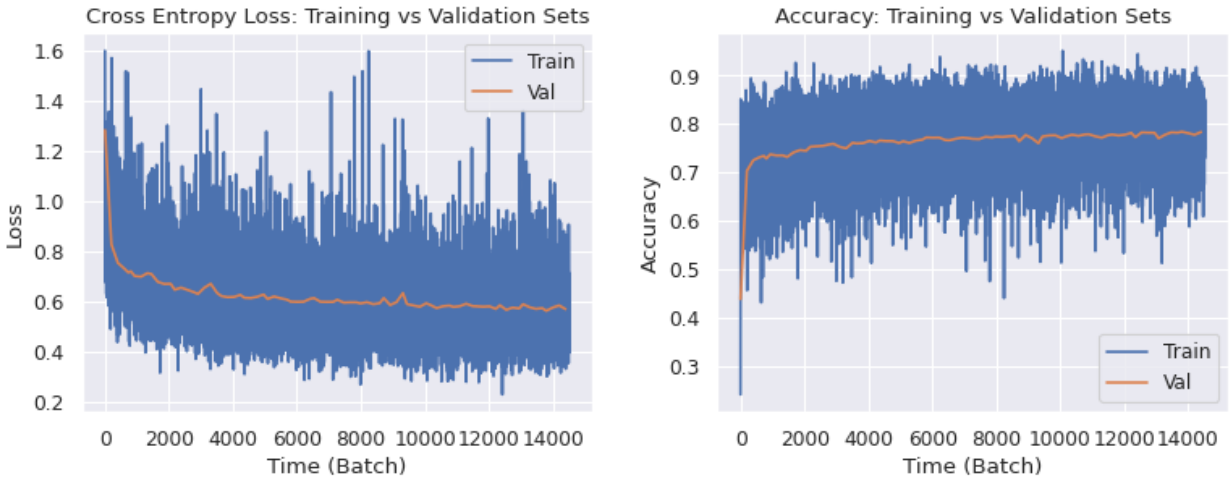

Figure 4.2: Accuracy and Loss

We don't see signs of overfitting, loss consistently decreases in the validation set, and accuracy consistently increases in the validation set over time. Figure 4.3 displays an example image, the ground truth classes, and our model's prediction on 12 classes.

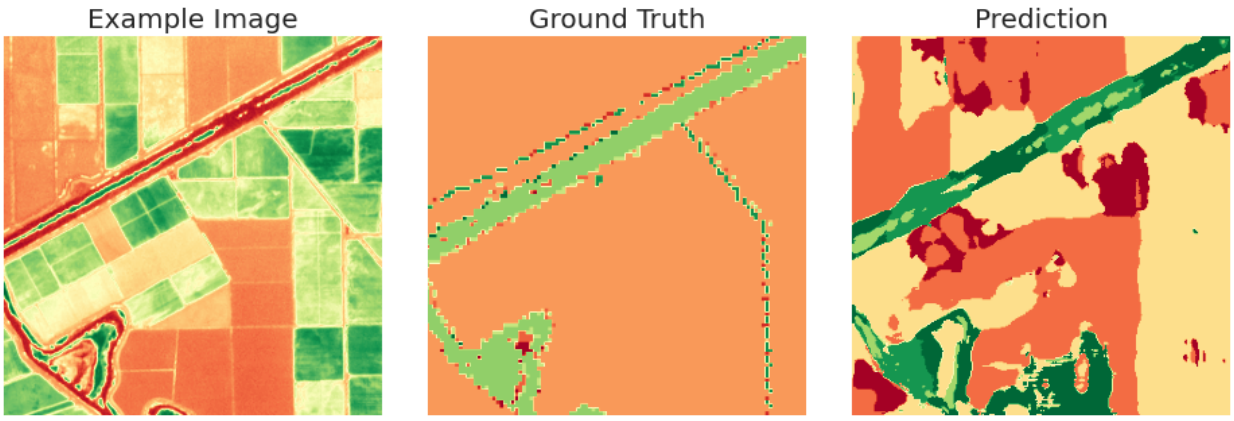

Figure 4.3: Example of Model Performance on 12 Classes

Note that the model is making steps toward differentiating between different crop types, which our truth labels are unable to do. This is an example of a limitation of the data we have that we aim to overcome through deep learning.

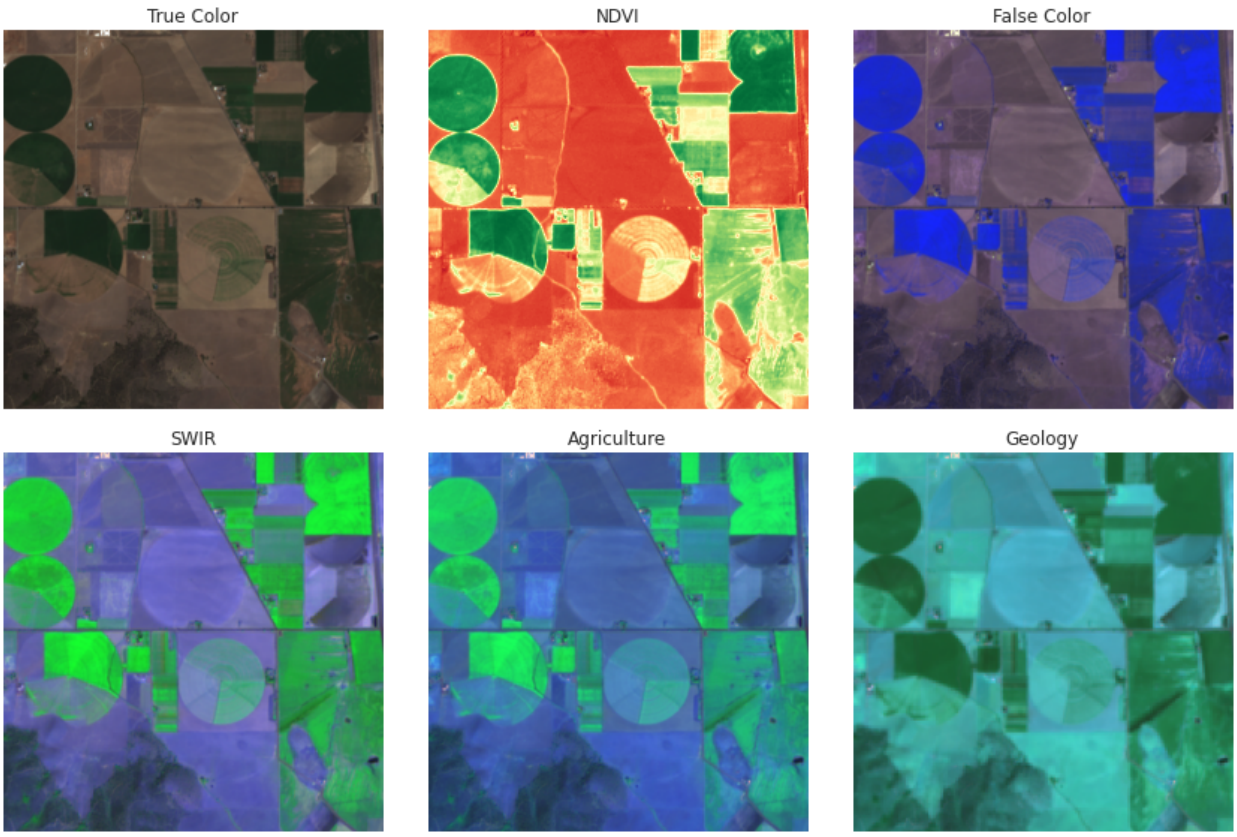

Figure 4.4: Example Image for Band Compositions

Different band compositions capture different properties in each image. In Figure 4.4 we

see an example of what the different compositions look like over the same area. Each band has different uses and illustrates different aspects of the vegetation. This is an interesting area of potential further research. Using deep learning, or more explainable statistical learning techniques, one could attempt to create an accurate profile of the information gained from each band composition. While some uses for each composition have been explained by literature, we did not come across literature comparing and contrasting what each band composition captures regarding vegetation or land features specifically.

The different performances achieved by each band composition are displayed in Table 4.1 below. Overall, performances are comparable across the board in both accuracy and F1-score. As we are considering multi-class problems, we average the F1-score across the classes and show the results of computing with weighted averaging and macro-averaging.

Table 4.1: Performance across band compositions

	All	NDVI	True Color	False Color	SWIR	Agriculture	Geology
4-Class Accuracy	0.81	0.79	0.78	0.81	0.82	0.80	0.81
4-Class F1 (weighted)	0.78	0.76	0.76	0.79	0.79	0.78	0.78
4-Class F1 (macro)	0.37	0.35	0.35	0.36	0.38	0.35	0.36
12-Class Accuracy	0.75	0.71	0.74	0.75	0.74	0.75	0.74
12-Class F1 (weighted)	0.71	0.66	0.69	0.70	0.70	0.70	0.71
12-Class F1 (macro)	0.14	0.11	0.11	0.13	0.13	0.13	0.14

As an example of the different predictions from different data, consider Figure 4.5. While the SWIR composition outperforms NDVI when it comes to our metrics, in this example the model trained on SWIR incorrectly reads a large swath of forest (dark green) as shrub (tan), where NDVI does not. Further exploration should be done to identify in what situations band compositions flourish and when they're likely to fail.

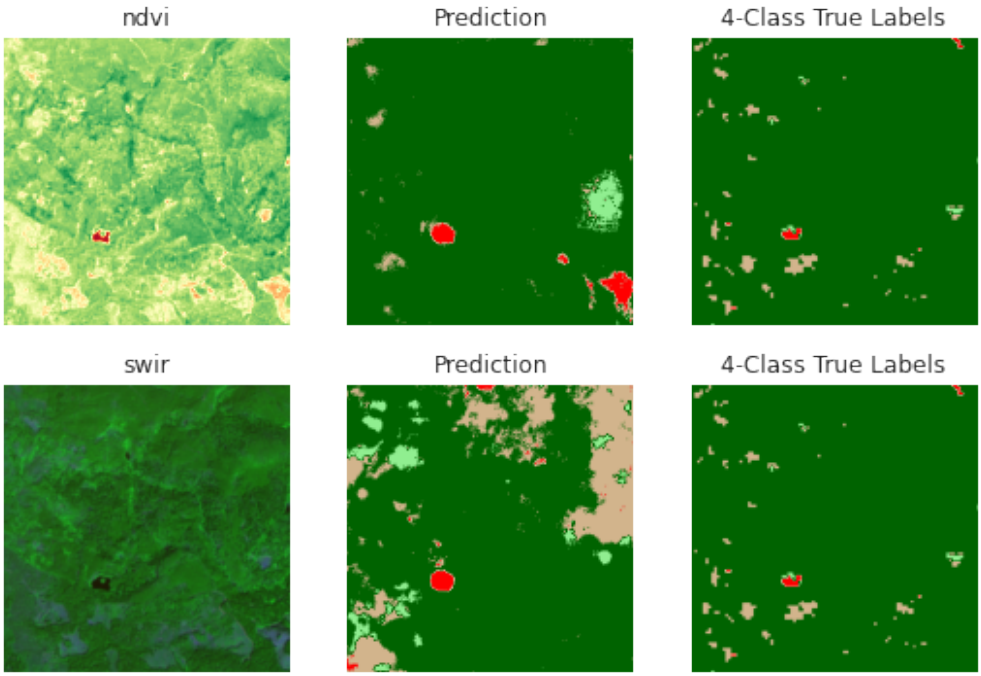

Figure 4.5: Example Comparing Results Using NDVI vs SWIR Bands

Figure 4.6 shows another comparison, this time between a model trained on the RGB composition and one trained on the Geology band composition. Note that while the RGB

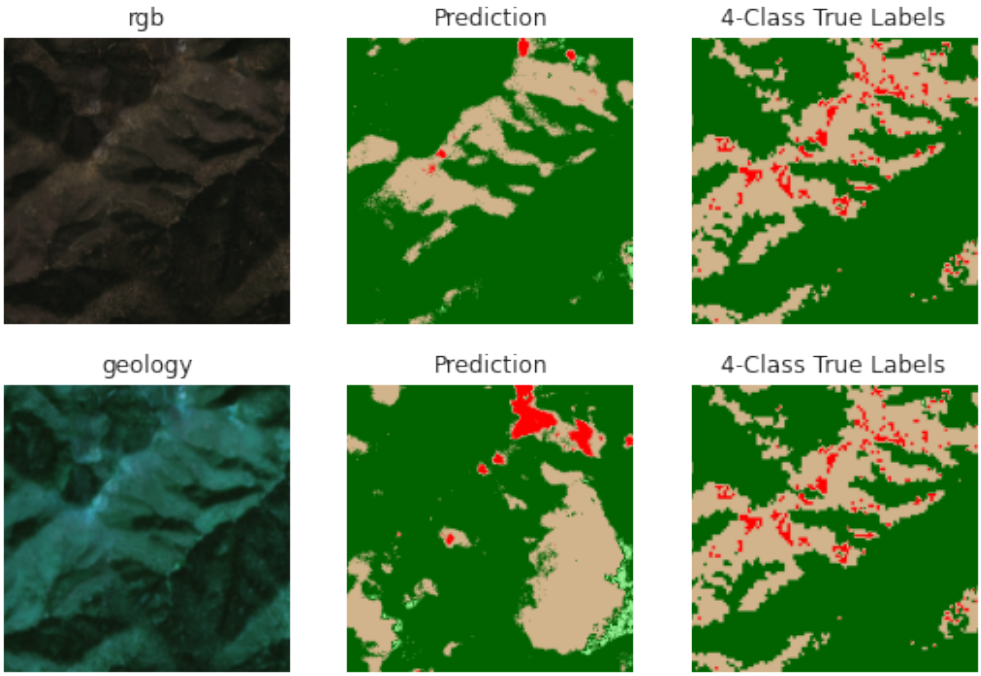

Figure 4.6: Example Comparing Results Using RGB vs Geology Bands

model captures much of the shape of the landscape represented in the 4-Class True Labels panel, the Geology model does not seem to match predictions at all.

For our work, we elect to use a model trained on NDVI, largely due to qualitative results and recommendations from literature [39]. To view more comparisons between results using different bands, see Appendix B.

4.5 Potential Improvements

Looking to the future, there are some clear areas of improvement. One potential improvement would be to adjust the loss function. Experimenting with alternative loss functions, such as Dice Loss, Focal Loss, Intersection Over Union, Boundary Loss, or even some adversarial loss may yield crisper boundaries, better ways to deal with unbalanced segmentation, or greater performance generally.

Another potential improvement to be considered is to use alternative architectures, such as Atrous Spatial Pyramid Pooling. More modern improvements to the current UNet architecture should also be considered, such as Dual Attention, MA UNet, and UNet++.

One important area of improvement is the robustness of our model. Incorporating alternative data (and possibly nested model techniques) may allow our model to perform well with significant cloud cover, handle images captured at night, and work well for images from different seasons of the year and geographical areas. These limitations are significant and should be addressed for the model to generalize to fires in a variety of circumstances.

Chapter 5. Data Assimilation

The goal of data assimilation is to optimally combine a mathematical model with observed data. For example, Mandel et al. make use of observed temperature data to update the states of their model with an ensemble Kalman filter [1]. In our approach, we seek to recover the optimal parameters for our PDE to most accurately describe and predict wildfire

behavior. We have access to geospatial data describing fire perimeters at discrete times as a wildfire evolves, provided to the public on FTP servers by the National Interagency Fire Center [40]. As data assimilation in this context has a number of complexities, the first part of this chapter details the data assimilation process for parameter recovery with the SIR (Susceptible-Infected-Removed) model for spread of disease, a much simpler PDE model. We use this as a stepping stone toward data assimilation in the context of wildfire prediction.

5.1 SIR model example

5.1.1 The SIR Model. An SIR model is a well-known epidemiological model that computes the number of people in a population that are infected with a contagious disease, susceptible to it, or "removed" (either they have deceased or they have recovered and developed immunity to the infection).

Among the simplest SIR models is the Kermack-McKendrick Model, given by

$$\begin{aligned}\frac{dS}{dt} &= -kSI \\ \frac{dI}{dt} &= kSI - qI \\ \frac{dR}{dt} &= qI\end{aligned}$$

where t is time, $S(t)$ is the fraction of susceptible people in the population, $I(t)$ is the fraction of the population that is infected, $R(t)$ is the fraction of the population who have been removed, k is the infection rate, and q is the recovery rate. For our example, we use this model and choose $k = 1$ and $q = 0.3$. The curves for S, I, and R produced with these parameter values from $t = 0$ to $t = 20$ are shown in Figure 5.1. In this example, the majority of the population is classified as "susceptible" at the beginning of the epidemic, but that portion of the population gradually declines to near 0 by the end. The infected population increases until $t = 7.5$, then decreases back toward zero. The recovered population increases throughout the duration of the experiment.

5.1.2 Data Assimilation with the Nelder-Mead Method. We begin by generating noisy "observable" data with which to perform data assimilation. We do this by randomly selecting t values from the first quarter of the model's lifespan ($t \in [0, 5]$), then calculating $I(t) + \epsilon_t$ and $R(t) + \delta_t$ where $\epsilon_t \sim \mathcal{N}(0,\, 0.05)$ and $\gamma_t \sim \mathcal{N}(0,\, 0.05)$ for each t. These noisy data points become our "observed" data. We plot the observed data along with the "true" SIR model in Figure 5.1.

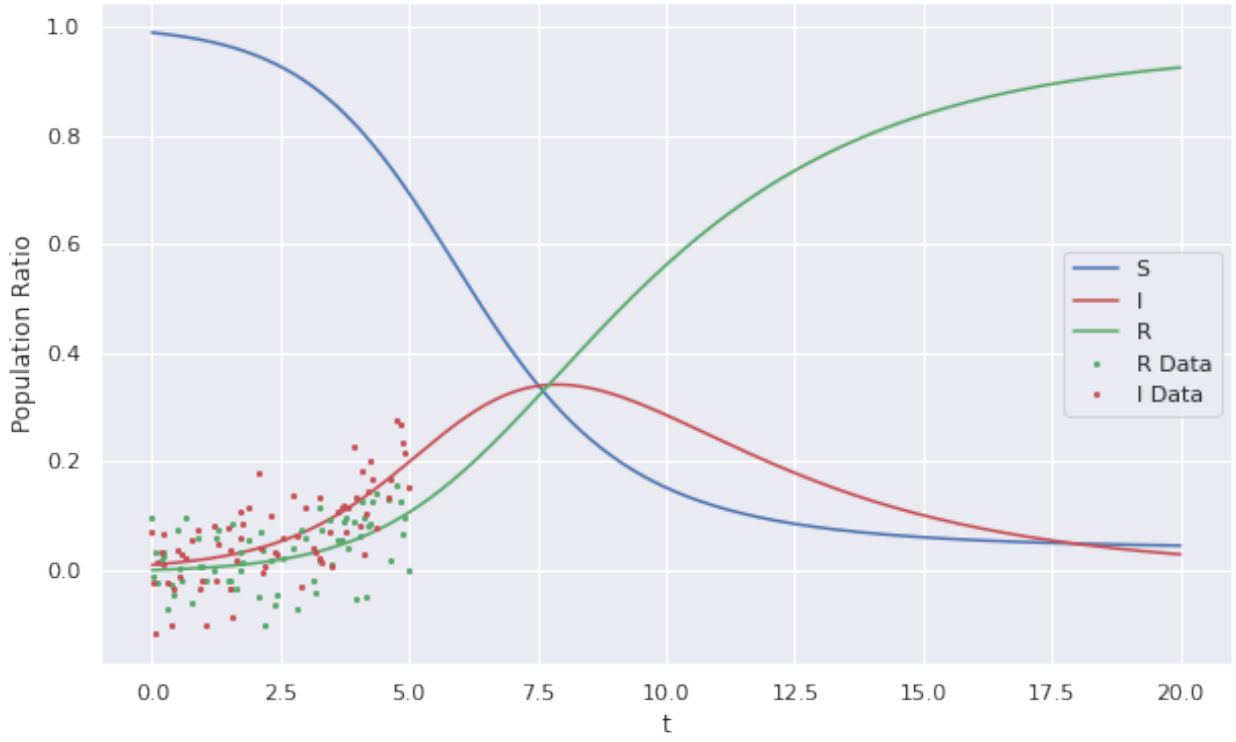

Figure 5.1: The True SIR Model and Generated Observable Data Points

The goal of our data assimilation process now is to find the parameters that define the SIR model that best fits our data. Our approach to accomplishing this is to define a function that accepts proposed values for k and q as inputs. The function solves the SIR model with these proposed parameters at the t values for which we have observed data, then computes the 2-norm of the vector of differences between the values of the proposed SIR model and the observed data. We then use the Nelder-Mead method to find the parameters that minimize this function [2]. There are a plethora of optimization algorithms to choose from, but we

select the Nelder-Mead method because it is among the best algorithms for multidimensional unconstrained optimization without using derivatives. While we can find gradients in this SIR example, we will not be defining a smooth function to optimize in our wildfire context. As a simplex search algorithm, Nelder-Mead is ideal for non-smooth optimization problems.

For our example, we provide initial guesses of 0.8 and 0.5 for our parameters, and the Nelder-Mead algorithm returns optimal values of 0.984 and 0.284, which are off by only 0.016 in both cases from our chosen values $k = 1$ and $q = 0.3$.

We plot the predicted SIR model based on these recovered parameters alongside the true model in Figure 5.2. Note the darker colors are associated with the predicted solution.

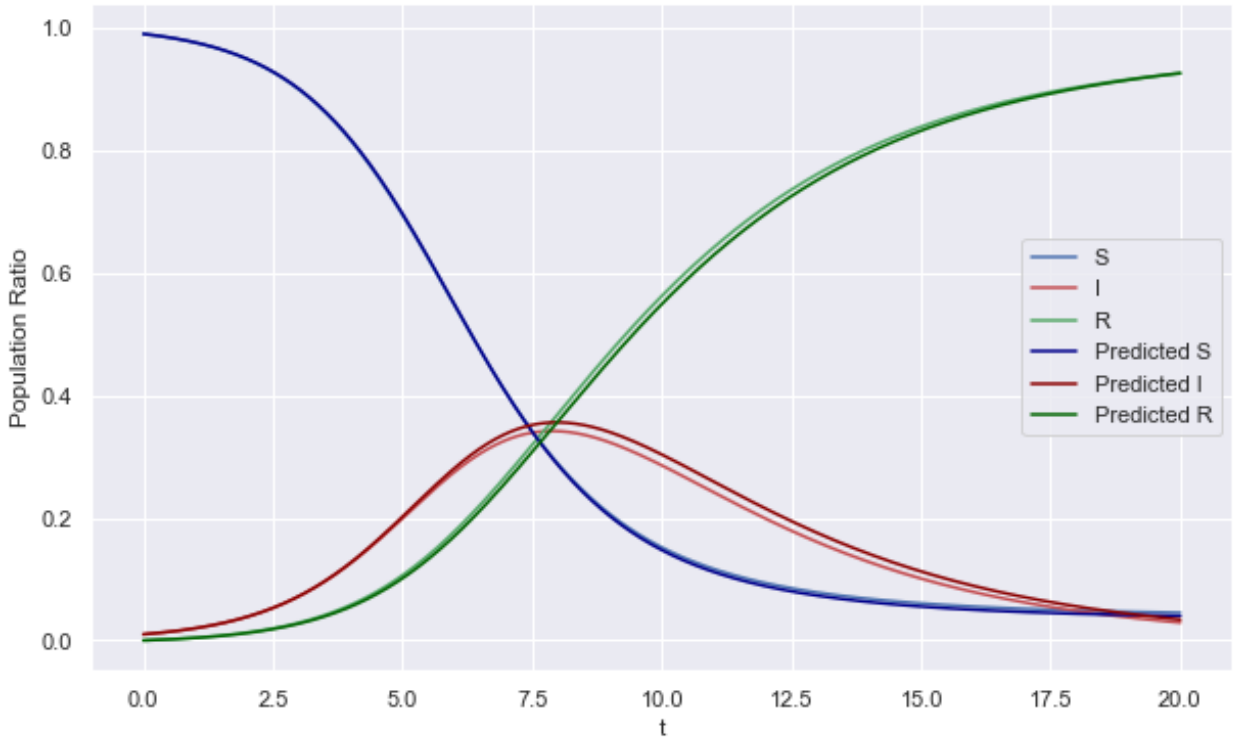

Figure 5.2: Comparing True SIR Model to Model from Nelder-Mead Solution

This approach to parameter recovery for data assimilation is effective and relatively efficient. In the following section we explore an alternative approach with Bayesian data assimilation.

5.1.3 Bayesian Data Assimilation. Data assimilation can also be performed in a Bayesian framework. The idea of Bayesian inference is to use Bayes' Theorem to update the probability of a hypothesis by using knowledge from informative data. In the context of Bayesian inference of parameters, Bayes' Theorem describes the relationship between prior knowledge of the parameters, knowledge provided through observed data, and updated knowledge after observing the data. The theorem is given by

$$P(\theta|x) = \frac{P(x|\theta)P(\theta)}{P(x)}, \tag{5.1}$$

where each P defines a probability distribution, x represents data, and θ represents a set of parameters. We call $P(\theta)$ the "prior" distribution, which captures our beliefs about how the parameters are distributed prior to observing data. We call $P(x|\theta)$ the "likelihood" as it is the likelihood of the observed data as a function of the model parameters. Then $P(\theta|x)$ is our "posterior" distribution that combines our prior beliefs about our parameters and our likelihood calculated from our data. $P(x)$ is a normalization factor that can be difficult to solve for, but this is overcome through the use of Markov Chain Monte Carlo (MCMC) methods, Variational Inference methods, or others. Because we have some idea of what parameters are realistic for a wildfire, this Bayesian approach seems to have promise.

For our SIR model, we model k and q with LogNormal distributions so that they cannot take on negative values. For our example, we choose prior distributions
$k \sim$ LogNormal(log(0.9), 0.1) and $q \sim$ LogNormal(log(0.5), 0.2). We define our likelihood function to be a Normal probability density function centered at our predicted model values for S or I at the times for which we have observations to compare.

For this problem we use NUTS (No U-Turn Sampler), a Hamiltonian Monte Carlo Method. This method converges faster and chooses samples better than MCMC methods such as Metropolis-Hastings, but it requires gradient computations. This makes it infeasible for our wildfire problem, but it is a powerful tool in the SIR context.

The details of these methods are beyond the scope of this paper, but to recover best model parameters, we fit a distribution to a collection of samples from the posterior distribution

that we retreive from our Monte Carlo algorithm, find the mode of the distribution, and call that our maximum a posteriori (MAP) estimate. This histograms of samples, fitted distributions, and MAP estimates for each parameter are found in Figure 5.3

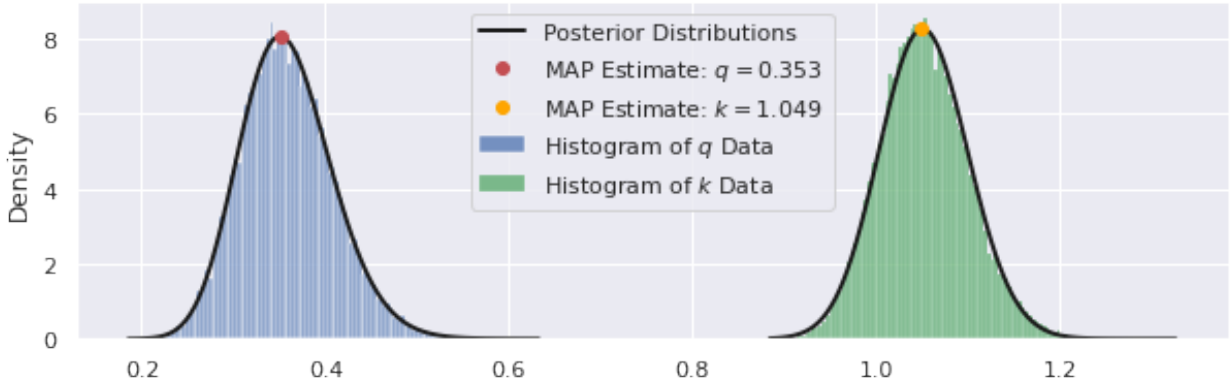

Figure 5.3: Posterior Distributions of Parameters for SIR Model

The parameters we recover from this process are $k = 1.049$ and $q = 0.353$. To evaluate our parameter recovery process, we plot our predicted SIR model using these values next to

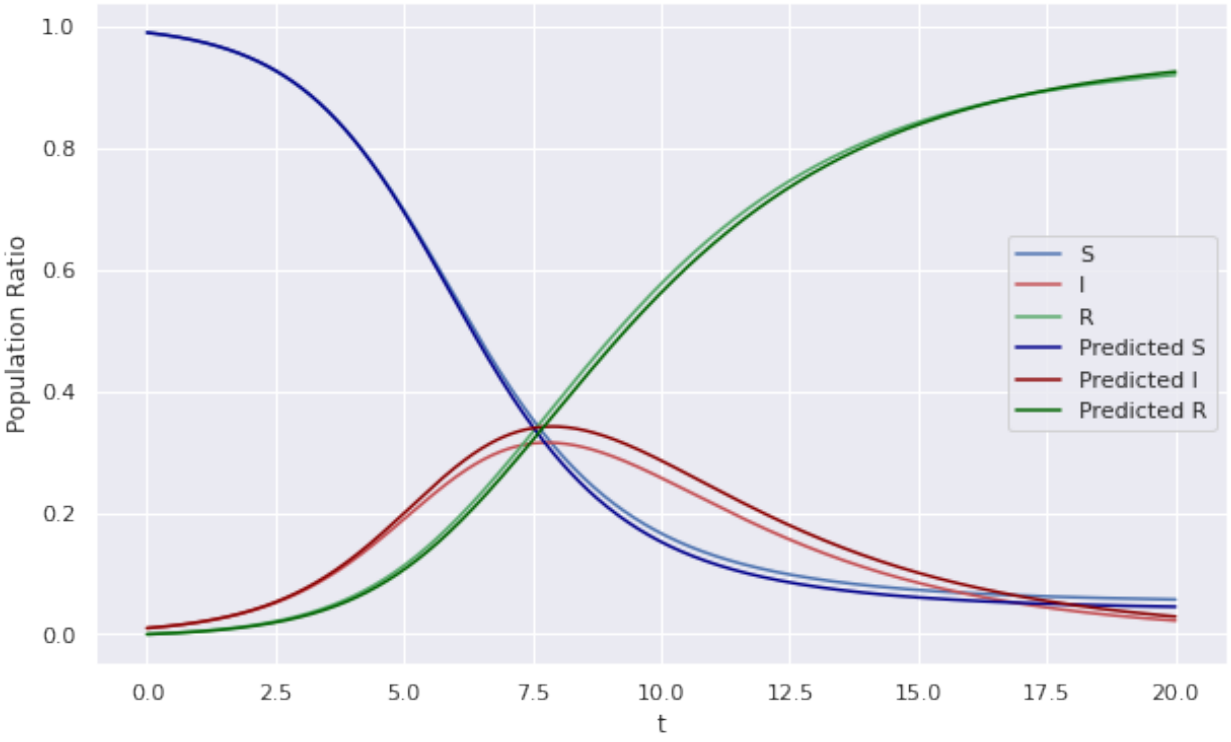

Figure 5.4: Comparing True SIR Model to Model from NUTS Solution

the true model in Figure 5.4 and see that the curves produced are quite similar, with the largest error being a small overestimation of the infected population during the peak of the epidemic.

5.2 Extension to wildfire model

The main idea of our approach to data assimilation with the wildfire model is the same as with the SIR model: we recover parameters that minimize a function that measures the difference between our model output and observed data. However, additional complexities in our wildfire model necessitate some changes to be outlined in this chapter. The PDE's in our wildfire model are more computationally expensive and must be performed many more times because of the size of the geographical mesh considered for a real wildfire, as well as the length of time the model must be solved over. Therefore, it takes much longer to solve the whole system ($\sim$ 2 hours compared to under a second for our SIR model example). Parallelization reduces this time by more than half, but this is not enough to make a Bayesian data assimilation approach feasible, as that would require potentially tens of thousands of function calls to generate enough samples to accurately approximate the posterior distributions of our parameters. Another obstacle to Bayesian inference is defining an effective likelihood function. Effective metrics for prediction of wildfire spread are often taken from set theory and are generally not smooth, such as the Jaccard Index (also known as Intersection over Union, or IoU), given by

$$J(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}, \tag{5.2}$$

for any two sets S_1 and S_2. This is a much better approach than comparing whether fire is present pointwise across all the coordinates of our predicted fire and the actual data. Such a naive result would be skewed by the size of the geographic area considered; for example, if we consider the entire State of California as part of our domain, we could predict no fire at all and still get 99% of the points correct. Ideally, a likelihood function would look

something more like intersection over union, but a non-differentiable likelihood function further complicates MCMC methods and makes an already infeasible problem even more computationally expensive. For this reason, we will only consider the Nelder-Mead method for data assimilation in the context of wildfires.

5.2.1 Performance Metrics. This IoU measure is one of many valuable metrics for assessing wildfire models [41]. To describe IoU in terms of fire perimeters, let a refer to the area that burns in both the predicted and actual fire perimeters, b refer to the area that burns in the predicted fire perimeter, but not the actual fire perimeter, and c be the area that burns in the actual fire perimeter, but not the predicted one. Then

$$IoU = \frac{a}{a+b+c}. \tag{5.3}$$

Other metrics include Simpson's coefficient,

$$\frac{a}{\min(a+b, a+c)}, \tag{5.4}$$

Braun's coefficient,

$$\frac{a}{\max(a+b, a+c)}, \tag{5.5}$$

and the Kulczynski measure, which is the arithmetic mean of Simpson's coefficient and Braun's coefficient [42, 43, 44, 41]. When predicting the spread of a fire, it's better to overestimate the size of the fire than to underestimate it and fail to protect people or resources in an area that was not expected to burn. With that in mind, we choose to use a weighted Kulczynski measure that favors $\frac{a}{a+c}$ over $\frac{a}{a+b}$. Then the performance metric we will focus on for data assimilation is given by

$$K = \frac{1}{4}\left(\frac{a}{a+b}\right) + \frac{3}{4}\left(\frac{a}{a+c}\right). \tag{5.6}$$

For each of these metrics, 1 indicates perfect overlap and 0 indicates no overlap at all. To view visualizations demonstrating these measures and many others, see Jeffrey Chao's excellent thesis on the subject [41].

5.2.2 The Final Model. We have some final adjustments to make to our model as we put everything together. While our Sentinel 2 images have 10m resolution, this would require a $12,000 \times 12,000$ square mesh of 10m units to cover the area burned by the fire. We adjust the resolution to 500×500 so that it's more manageable for our constraints on computation time and memory required to store our mesh. We further adjust our atmospheric data to reflect proper units. We have ambient temperature data and wind data from the National Centers for Environmental Information that need to be adjusted [45]. The temperature is transformed from Fahrenheight to Kelvin, and the wind data is converted from speed (s) and direction (d) values to a vector to be used in our PDE by the following function, $f(s, d) = [s\sin(d), s\cos(d)]$. These values are adjusted every hour for which we have data, and the wind vector is scaled by a parameter, w, for which we solve in the Nelder-Mead simplex.

The data we are using for assimilation describes only the perimeters of the fire at discrete times, as shown in Figure 5.5. As we do not have temperature data, we initialize our temper-

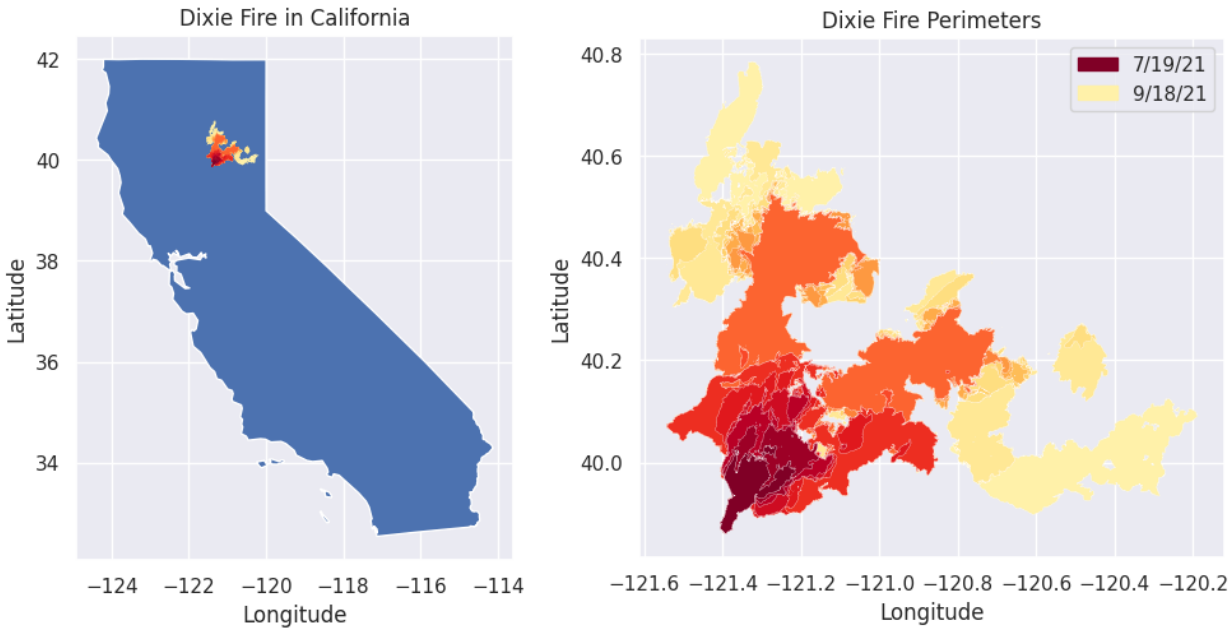

Figure 5.5: The Dixie Fire (California, 2021)

ature function with the ambient temperature everywhere outside the initial fire perimeter,

and 800C within the fire perimeter. We record the maximum temperature at each coordinate for the duration of the model, choose a generous cut-off value of 600C, and consider a point to be within our fire perimeter if at any point its temperature exceeded this cut-off. These values are inspired by a typical temperature trace of a flame front during a grass fire, as recorded by Mike Wonton at the Firelab of the University of Toronto [46]. We initialize our fuel function in a similar way, setting every point outside the initial fire perimeter to 1 and every point within the fire perimeter to 0.2, signifying the majority of the fuel has burned.

In addition to w, our model has parameters B and C, from the PDE model due to Mandel et al. [1]. These values are independent of fuel type, and so remain constant throughout the geographical area of interest. The other parameters, k, A, and C_s are functions of fuel type. For each of these parameters, we use the fuel map with 4 fuel types (Grass, Shrub, Tree, and Other) from our deep learning model to create a function that maps 4 values to the locations of the 4 fuel types in our fuel map. Then, for example, if grass is present at location (x_i, y_j), the PDE will use values k_1, A_1, and C_{s_1} to compute the change in temperatuer and fuel at that location. If shrubs are present, the values will be k_2, A_2, and C_{s_2}, and so on.

Now the function to be minimized by the Nelder-Mead method has 15 parameters: w, B, C, k_1, k_2, k_3, k_4, A_1, A_2, A_3, A_4, C_{s_1}, C_{s_2}, C_{s_3}, and C_{s_4}. This function computes our PDE solution with proposed parameters and calculates the weighted Kulczynski values comparing each predicted and real fire perimeter. Then it takes $1-K_p$ for each Kulczynski measure, K_p, of perimeter p, and returns the sum $\sum_{p=1}^{n}(1-K_p)$ where n is the total number of perimeters for which we have data. We complete our data assimilation by using Nelder-Mead to find the parameters that minimize this function.

Chapter 6. Model Evaluation

We evaluate our resulting model both qualitatively and quantitatively. We calculate IoU and Kulczynski values to measure how well the predicted and actual perimeters match up, and plot the perimeters on top of one another to evaluate how well they match visually.

6.1 On Generated Fires

In this section we generate fires and fuel maps on which to test our model.

6.1.1 Two ignition points and a lake. In this example, we ignite two square fires in a grassland with a circular lake in the center. See the corresponding fuel map in Figure 6.1. Red represents fire, green is grass, and blue is water. With these initial conditions, and initial

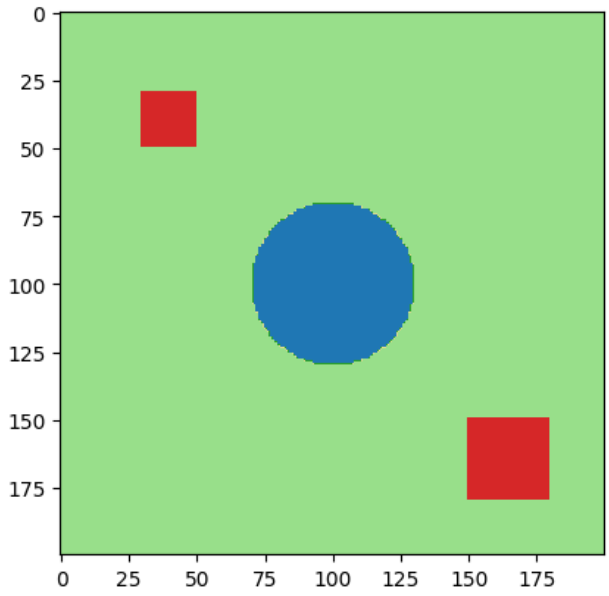

Figure 6.1: Example 1 Fuel Map

fuel map of 1 everywhere but the fires, we choose parameters $w = 0$, $B = 100$, $C = 0.0001$, $k_1 = 0$, $k_2 = 0.0333$, $k_3 = 0.0417$, $k_4 = 0.05$, $A_1 = 8000$, $A_2 = 15000$, $A_3 = 12000$, $A_4 = 10000$, $C_{s_1} = 3$, $C_{s_2} = 2$, $C_{s_3} = 2.5$, and $C_{s_4} = 3$. This generates fire perimeters on our domain as shown in Figure 6.2.

After performing our data assimilation parameter recovery method with random initial starting parameters, we recover parameters $w = 0$, $B = 125.799108$, $C = 0.00018934749$, $k_1 = 0$, $k_2 = 0.037337778333333335$, $k_3 = 0.04667222291666667$, $k_4 = 0.0560066675$, $A_1 = 10588.94016$, $A_2 = 19854.2628$, $A_3 = 15883.41024$, $A_4 = 13236.1752$, $C_{s_1} = 0.449094034$, $C_{s_2} = 0.29939602266666665$, $C_{s_3} = 0.37424502833333334$, and $C_{s_4} = 0.449094034$. While these values do not precisely match the actual parameters, and therefore the models may

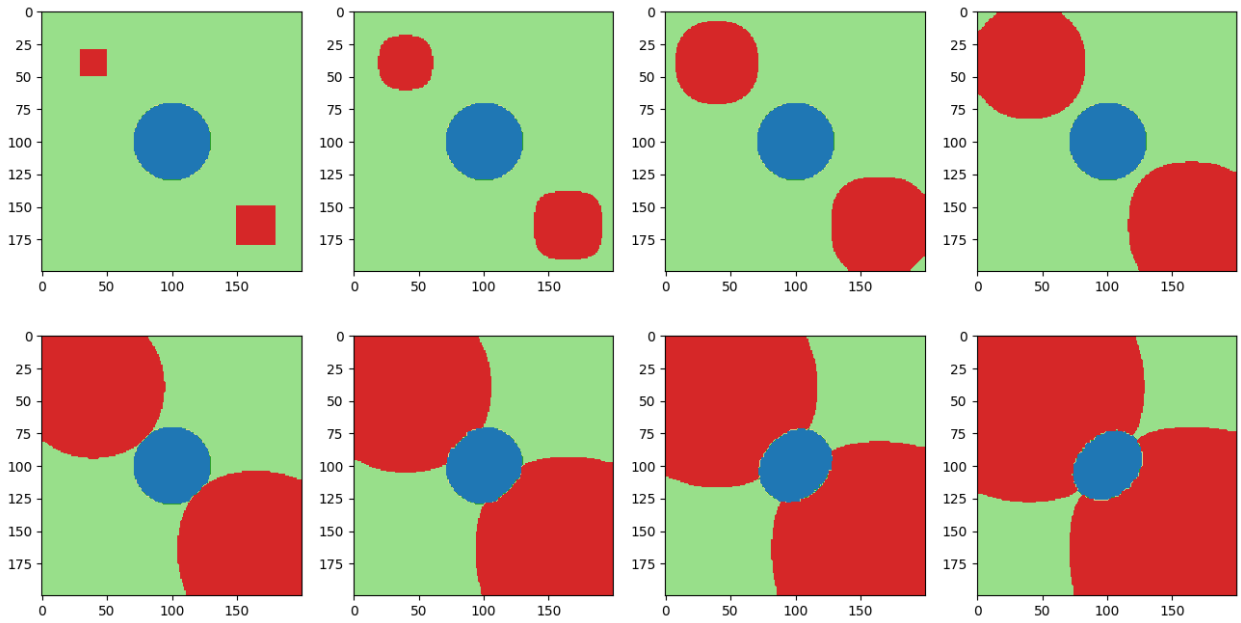

Figure 6.2: Example 1 Fire Perimeters

produce different temperature predictions, we see that the actual fire perimeters predicted overlap almost exactly with the true fire perimeters. In Table 6.1, we can see the resulting IoU and weighted Kulczynski scores, which are all very close to 1. This is an excellent result!

Table 6.1: Example 1 Evaluation Metrics

Perimeter #	1	2	3	4	5	6	7	8
Kulczynski	1	0.990	0.996	0.996	0.997	0.999	0.999	0.999
IoU	1	0.986	0.994	0.995	0.997	0.998	0.998	0.998

We have recovered parameters that accurately describe how the fire has developed in our domain. In Figure 6.3, we see this visually. The true fire perimeters are plotted in white, and the predicted ones are plotted over them in red. The resulting orange-pink hue indicates that they overlap almost exactly.

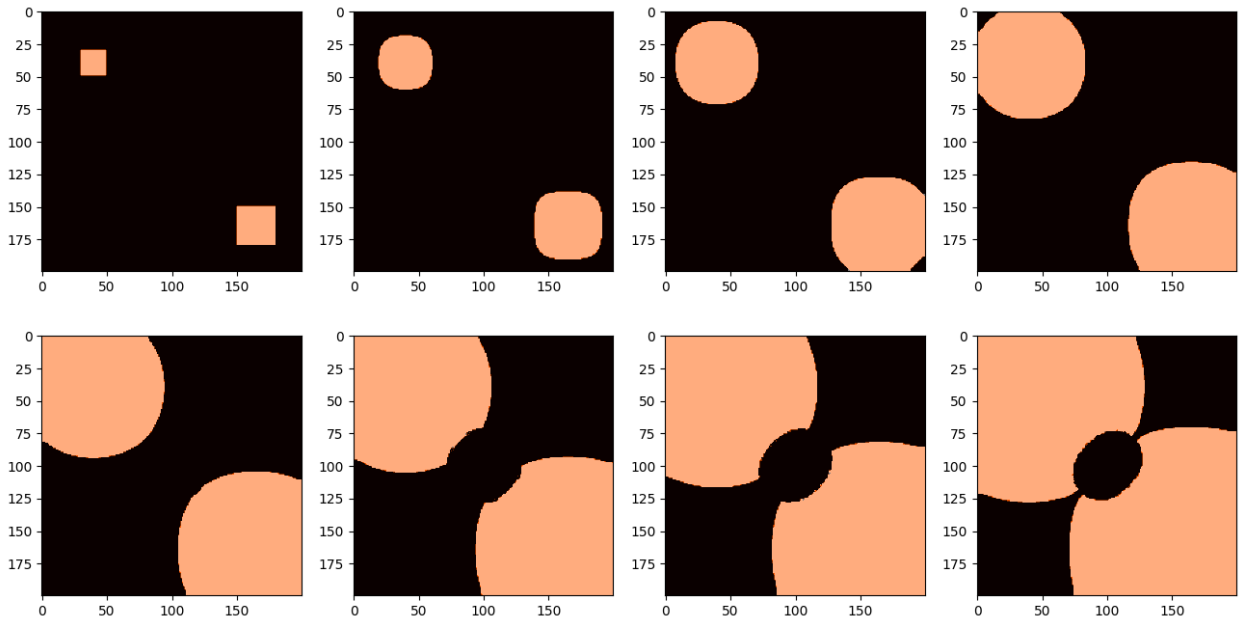

Figure 6.3: Example 1 Predictions

6.1.2 Heterogeneous fuel map. In this second example, we consider a heterogeneous fuel map, as shown in Figure 6.4. Once again, blue represents water, red is fire, and light green is grass. However, now we have added a tan color to represent shrubs, and a dark green to represent trees.

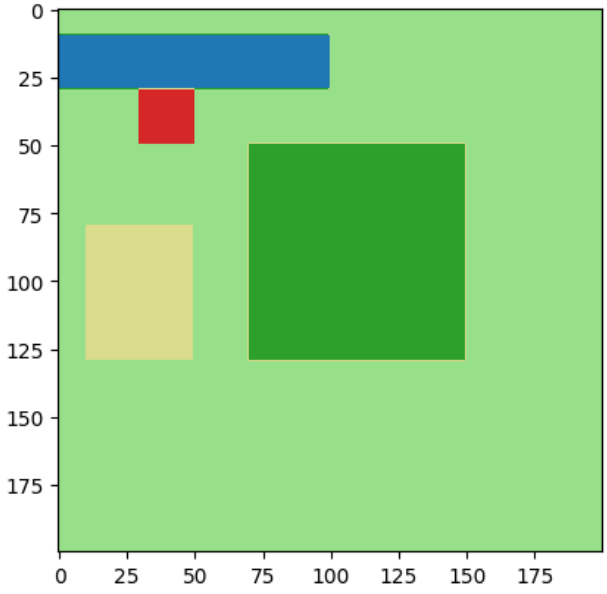

Figure 6.4: Example 2 Fuel Map

We use parameters $w = 0$, $B = 100$, $C = 0.000125$, $k_1 = 0$, $k_2 = 0.03333$, $k_3 = 0.0416667$, $k_4 = 0.1$, $A_1 = 1600$, $A_2 = 12000$, $A_3 = 9600$, $A_4 = 8000$, $C_{s_1} = 3$, $C_{s_2} = 2$, $C_{s_3} = 2.5$, and $C_{s_4} = 3$. These values were chosen partially to demonstrate drastic differences in the temperature and speed at which different fuel types burn. In Figure 6.5, we can see the resulting fire perimeters. Note how the fire burns through grass more quickly than shrubs or trees.

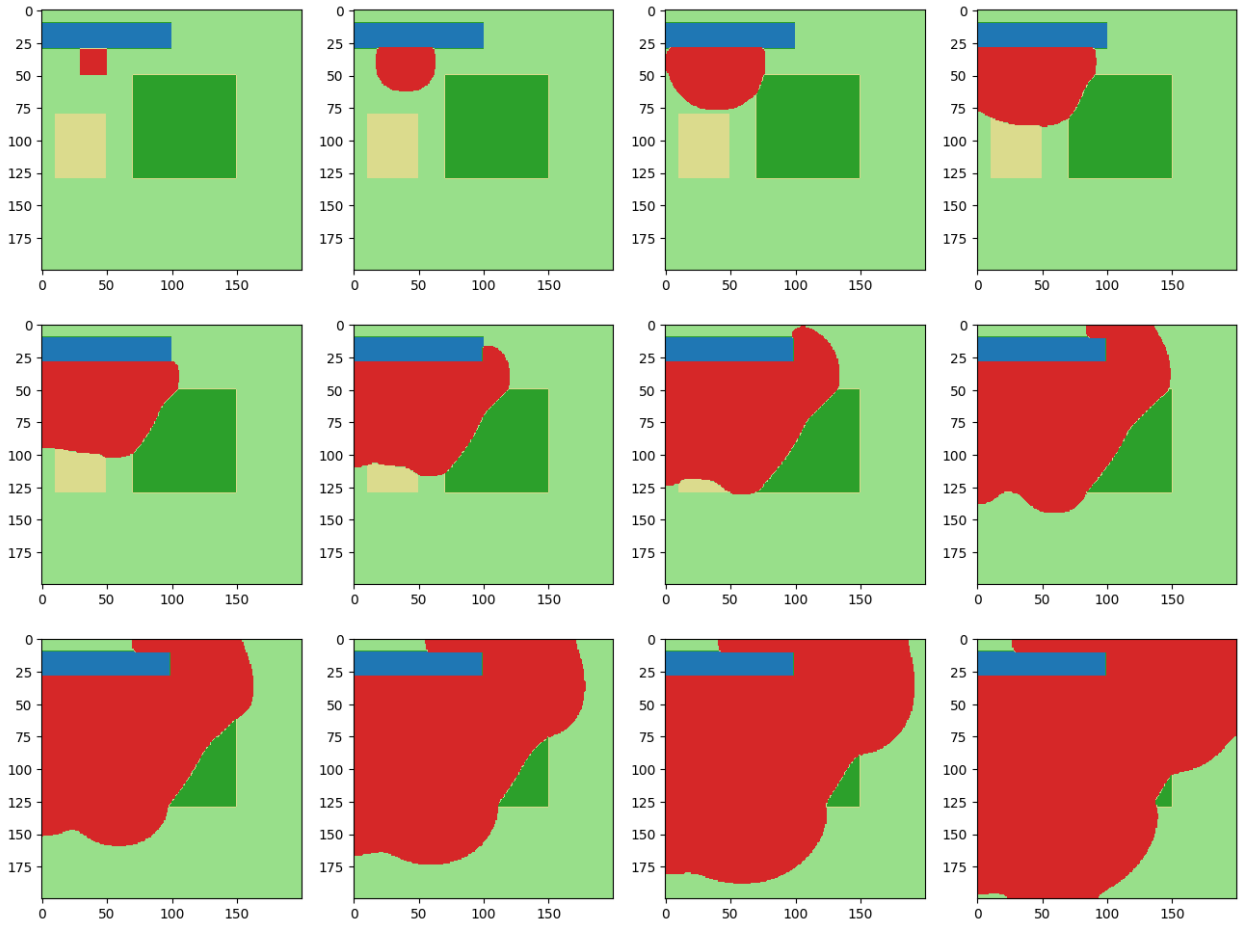

Figure 6.5: Example 2 Fire Perimeters

Upon performing data assimilation, we recover parameters $w = 0$, $B = 149.001811$, $C = 0.00019198904$, $k_1 = 0$, $k_2 = 0.045877042400000005$, $k_3 = 0.05734630300000001$, $k_4 = 0.1376311272$, $A_1 = 2097.74372$, $A_2 = 15733.0779$, $A_3 = 12586.46232$, $A_4 = 10488.7186$, $C_{s_1} = 1.77412965$, $C_{s_2} = 1.1827531$, $C_{s_3} = 1.478441375$, and $C_{s_4} = 1.77412965$. Using these parameters, we model and record predicted fire perimeters to compare to our true

perimeters. Once again, predicted perimeters are in red, while true perimeters are in white. The resultant pink-ish hue represents overlapping prediction and truth.

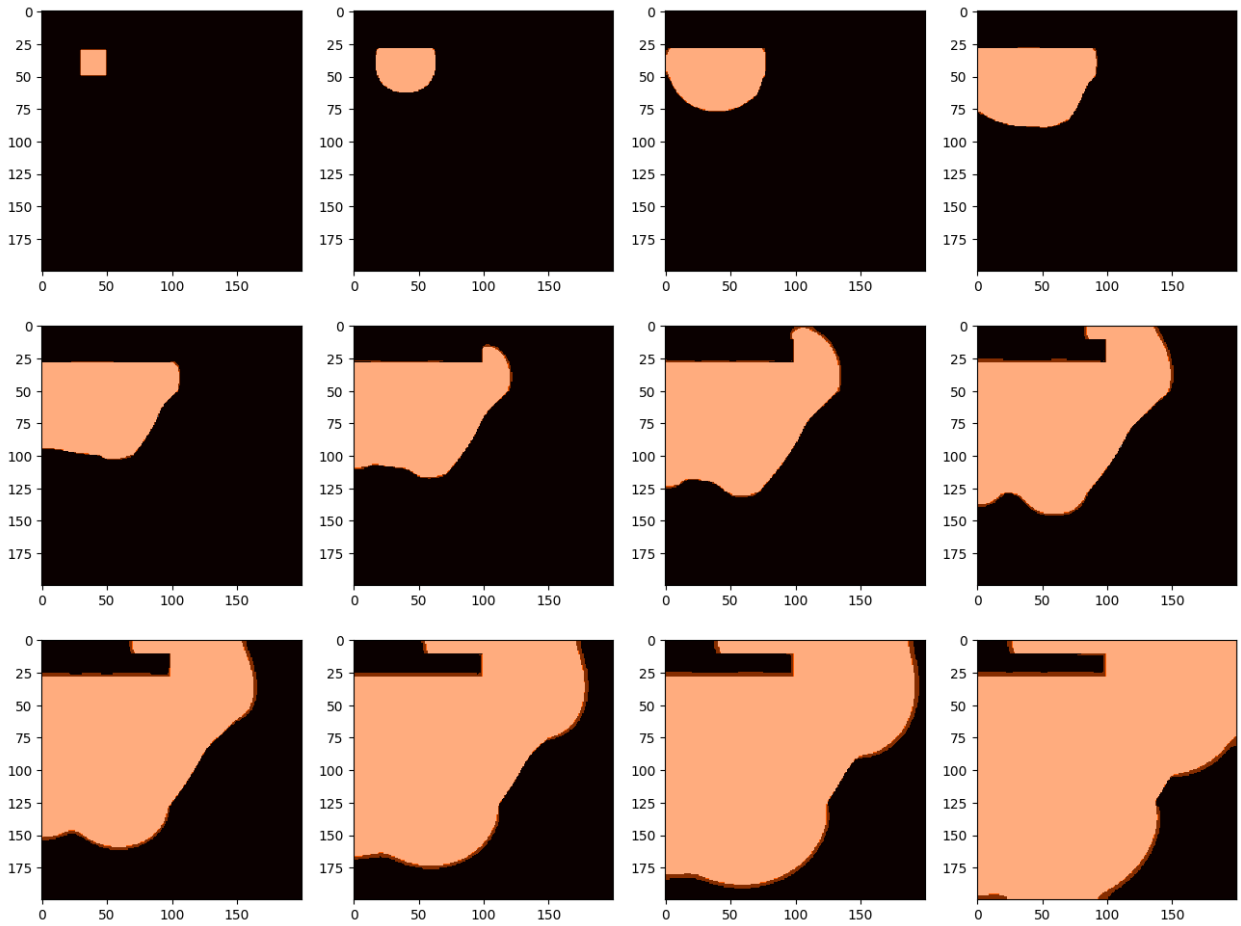

Figure 6.6: Example 2 Predictions

In Table 6.2, we can see the resulting IoU and weighted Kulczynski scores, which are again all close to 1.

Table 6.2: Example 2 Evaluation Metrics

Perimeter #	1	2	3	4	5	6
Kulczynski	1.000	0.986	0.996	0.997	0.996	0.995
IoU	1.000	0.981	0.989	0.991	0.987	0.983
Perimeter #	7	8	9	10	11	12
Kulczynski	0.993	0.992	0.992	0.991	0.990	0.993
IoU	0.972	0.970	0.967	0.963	0.960	0.970

6.2 On Data from the Dixie Fire (2021)

We now perform our approach on real fire data. While perimeters were predicted exactly on generated data, we know that our simplified model does not encapsulate every aspect of fire spread in the real world, so we should expect some significant error when transitioning to real data. There are a variety of other sources of error to consider as well.

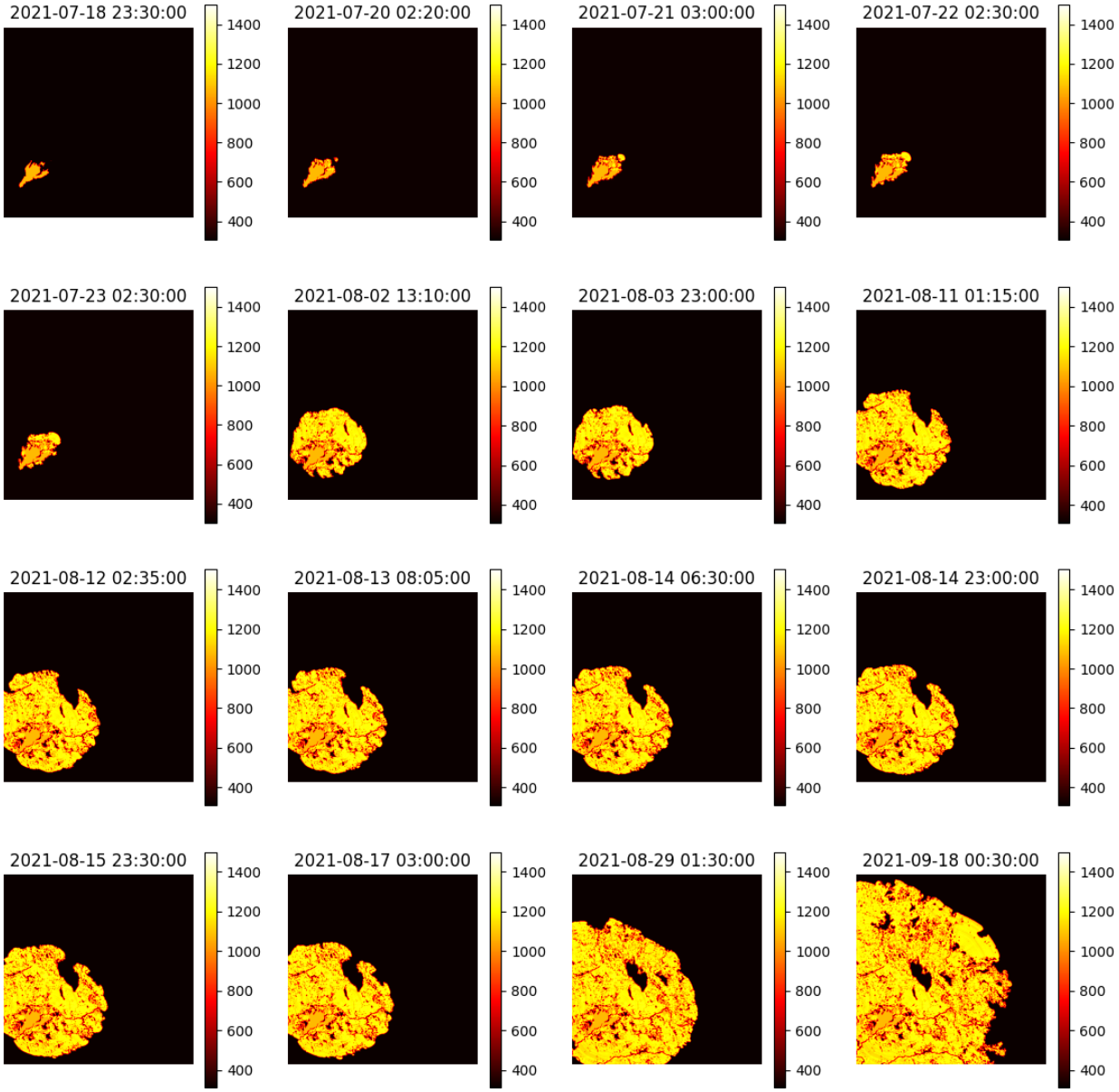

Figure 6.7: Predicted Temperature for the Dixie Fire

After some experimentation, we initialized our parameters to be $w = 0.001$, $B = 200$,

$C = 0.001$, $k_1 = 0$, $k_2 = 0.00005$, $k_3 = 0.00005$, $k_4 = 0.00005$, $A_1 = 800$, $A_2 = 2000$, $A_3 = 2000$, $A_4 = 2000$, $C_{s_1} = 0.05$, $C_{s_2} = 0.01$, $C_{s_3} = 0.01$, and $C_{s_4} = 0.01$. Despite running this process for several days, Nelder-Mead failed to find a minimum in that time frame. However, the best parameter set located during this period is $w = 0.00142528678$, $B = 202.087845$, $C = 0.000897715211$, $k_1 = 9.75 \times 10^{-6}$, $k_2 = 6.17728210 \times 10^{-6}$, $k_3 = 8.23637613 \times 10^{-6}$, $k_4 = 1.23545642 \times 10^{-5}$, $A_1 = 836.732853$, $A_2 = 2510.19856$, $A_3 = 2008.15885$, $A_4 = 1673.46571$, $C_{s_1} = 0.0428445756$, $C_{s_2} = 0.0142815252$, $C_{s_3} = 0.0285630504$, and $C_{s_4} = 0.0428445756$. This results in the temperature predictions found in Figure 6.7. Recall that the temperatures plotted here are the maximum temperatures recorded at each coordinate up to the time specified.

These temperature predictions (in Kelvin) are reasonable values to be expected in a Wildfire context. We use our cut-off value to produce masks of the predicted fire perimeters and plot them over the true fire perimeters in Figure 6.8. As before, the true perimeters are in white, while the predicted perimeters are in red.

While not perfect, our predicted fire perimeters approximate the size of the fire and the speed at which it is spreading well. The goal is to improve the accuracy through future work, but current qualitative results may help inform fire management decisions. In Table 6.3, we see the values of our metrics for each perimeter.

Table 6.3: Dixie Evaluation Metrics

Perimeter #	1	2	3	4	5	6	7	8
Kulczynski	1.00	0.74	0.67	0.65	0.58	0.72	0.72	0.60
IoU	1.00	0.64	0.56	0.54	0.45	0.50	0.48	0.41
Perimeter #	9	10	11	12	13	14	15	16
Kulczynski	0.60	0.59	0.59	0.59	0.59	0.57	0.70	0.66
IoU	0.41	0.39	0.38	0.38	0.38	0.36	0.42	0.34

We see in both metrics (and the figures) that the predicted perimeters are better for the first few weeks, and drop off in quality significantly after that. When using the model in real time, the computations could be restarted with new initial conditions each time an

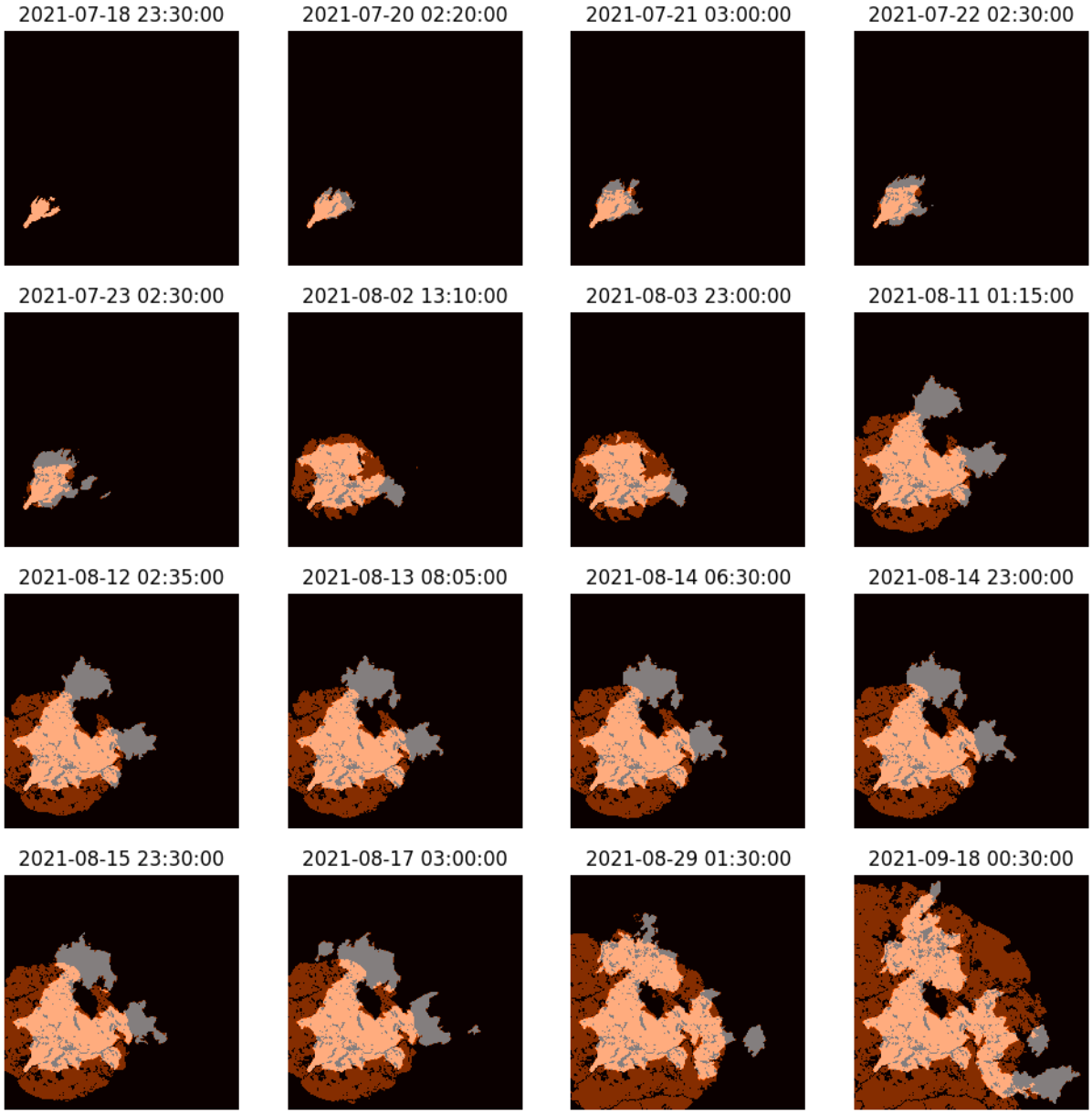

Figure 6.8: Predicted Fire Perimeters for the Dixie Fire

observation is made. This method results in the perimeters found in Figure 6.9. The metrics for these perimeters are found in Table 6.4.

As the Dixie Fire is the largest single fire in California history, it burned over a much larger area and for a longer time than the vast majority of wildfires [13]. We expect our method to produce even more accurate results on smaller, shorter-lived fires.

Table 6.4: Dixie Evaluation Metrics with Restarts

Perimeter #	1	2	3	4	5	6	7	8
Kulczynski	1.00	0.74	0.86	0.91	0.85	0.84	0.95	0.75
IoU	1.00	0.64	0.79	0.85	0.78	0.50	0.81	0.65

Perimeter #	9	10	11	12	13	14	15	16
Kulczynski	0.98	0.97	0.99	0.99	0.99	0.95	0.86	0.78
IoU	0.92	0.93	0.96	0.97	0.97	0.92	0.61	0.41

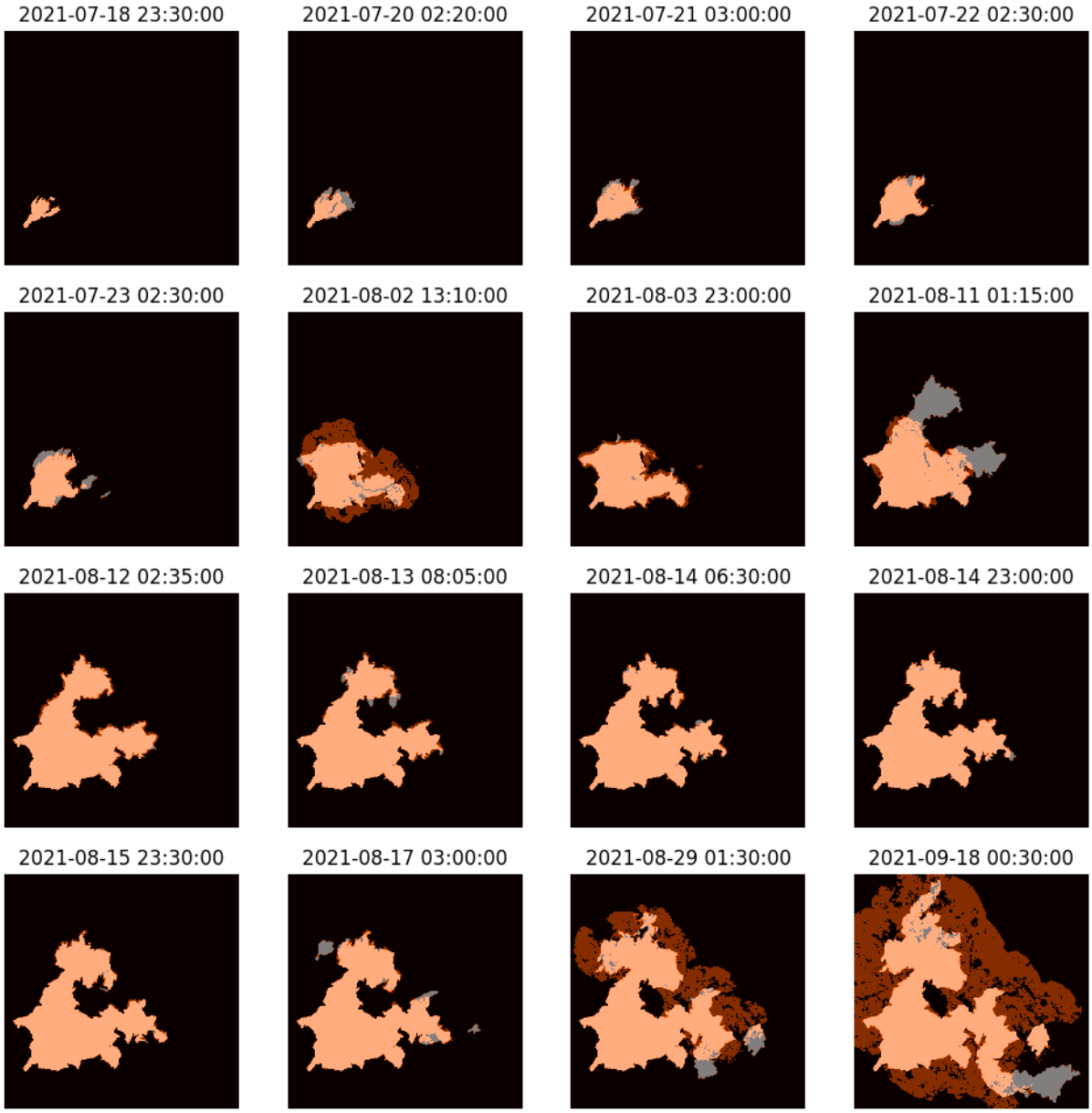

Figure 6.9: Predictions Restarting with Each Observation

Chapter 7. Conclusion

We have shown that our model, although much simpler than what is considered state-of-the-art, can effectively approximate the size and speed of an active wildfire. We have developed a novel deep learning application for recovering fuel distributions in real time and an efficient, parallelizable numerical method for evaluating our PDE model. In particular, our approach recovers accurate fire perimeters within the first few weeks of modeling the Dixie Fire. We expect our method to produce more accurate results on fires that are smaller and do not burn for as lengthy a time as the Dixie Fire. In the next section we present some ideas for potential improvements upon the method described in this paper.

7.1 Future Directions

There are many changes that could be made to immediately improve our approach, the most effective of which may be to add a term to the PDE, or a method to adjust temperature and fuel initializations, in order to describe the effect of topography on the change in temperature. For example, fire burns more quickly uphill, and will not travel across the boundary of a steep cliff face. Other improvements include an effort to measure vegetation volume and density, or to remotely estimate moisture content of fuel [47, 48, 49].

For quicker and more accurate parameter recovery, this method could be adapted using the idea behind the architecture of ProGAN [50]. We could begin with a low resolution and recover optimal parameters for the system. Then, we could transform these parameters to get initial parameters for the same problem at a higher resolution, and continue this process until we recover parameters at full resolution.

Appendix A. Selections from Code

All code for this project can be found on GitHub at: `https://github.com/drewjohnston13/wildfire_modeling`.

A.1 Crank-Nicholson Finite Difference Scheme Code

Below is a function for approximating the solution to our PDE model with the Crank-Nicholson scheme in two dimensions.

```
import numpy as np
import scipy.sparse as sp
import scipy.sparse.linalg as spla


def crank2D(k,A,B,C,Cs,T0,S0,g,Ta,v,a,b,tf,J,N):
    """
    Uses the Crank-Nicholson scheme to compute the solution to the PDE system:

        T_t = k(T_xx+T_yy) - (v1T_x+v2T_y) + A(Se^(-B/(T-T_a))-C(T-T_a))
        S_t = -C_SSe^(-B/(T-T_a)),  T > T_a

            on R = [a,b]×[a,b]

        Boundary Conditions:
            T(t,x,y) = g(x,y) on  dR (Dirichlet)
            S(t,x,y) = S(t-1,x+(-)dx,y+(-)dy) on dR (Numerical: No Flux)

        Initial Conditions:
            T(0,x,y) = T0(x,y)
            S(0,x,y) = S0(x,y)
```

```
    given N time steps and J space steps.
"""
# Initialize space and time grids
x,dx = np.linspace(a,b,J+1,retstep=True)
y,dy = np.linspace(a,b,J+1,retstep=True)
t,dt = np.linspace(0,tf,N+1,retstep=True)
X,Y = np.meshgrid(x,y)

# Initialize solution array
T = np.zeros((N+1,J+1,J+1))
S = np.zeros((N+1,J+1,J+1))
T[0] = T0(X,Y)
S[0] = S0(X,Y)

# Define matrix for Laplacian term
block = sp.diags(([1]*(J-2),[-4]*(J-1),[1]*(J-2)),offsets=(-1,0,1))
Dxxyy = sp.block_diag([block]*(J-1)) \
                + sp.diags(([1]*((J-1)**2-J+1),[1]*((J-1)**2-J+1)),
                offsets=(-J+1,J-1))
Dxxyy = Dxxyy*k*dt/(2*dx**2)

# Define matrix for single derivative term
v1, v2 = v
block = sp.diags(([-v1]*(J-2),[v1]*(J-2)),offsets=(-1,1))
Dxy = sp.block_diag([block]*(J-1)) \
                + sp.diags(([v2]*((J-1)**2-J+1),[-v2]*((J-1)**2-J+1)),
                offsets=(-J+1,J-1))
```

```
Dxy = Dxy*dt/(4*dx)


# Define matrix for T^n+1_i,j term.
I = sp.eye((J-1)**2)


# Collect "n+1 terms" on left side in matrix for solving
M = Dxxyy-Dxy
left = (I-M).tocsr()


# Solve at each time step.
for n in range(N):
    # Collect "n terms" for right side
    Tn = T[n,1:-1,1:-1].flatten()
    Sn = S[n,1:-1,1:-1].flatten()
    f = Tn+M@Tn+dt*A*(Sn*np.exp(-B/(Tn-Ta))-C*(Tn-Ta))

    # Add Boundary Conditions
    p = np.zeros((J-1,J-1))
    p[:,[0,-1]] = g(X[1:-1,[0,-1]],Y[1:-1,[0,-1]])
    p = p.flatten()/dx**2
    q = np.zeros((J-1)**2)
    q[:J-1] = g(X[0,1:-1],Y[0,1:-1])/dx**2
    q[-J+1:] = g(X[-1,1:-1],Y[-1,1:-1])/dx**2

    # Solve the system
    T[n+1] = g(X,Y)
    T[n+1,1:-1,1:-1] = spla.spsolve(left,f-p-q).reshape((J-1,J-1))
```

```
        S[n+1,1:-1,1:-1] = S[n,1:-1,1:-1] \
                    - dt*Cs*S[n,1:-1,1:-1]*np.exp(-B/(T[n,1:-1,1:-1]-Ta))
        # Numerical BCs:
        S[n+1,[0,-1],:] = S[n,[1,-2],:]
        S[n+1,:,[0,-1]] = S[n,:,[1,-2]]
    return T,S
```

A.2 Finite Element Code

Below is a function for solving the PDE model in two dimensions using FEniCS.

```
def solve_pde(k, A, B, C, Cs):
    # Define constants.
    a,b = 0,10
    x1,y1 = a,a
    x2,y2 = b,b
    Nx = 800
    Ny = 800
    tf = 3
    Nt = tf*10
    t,dt = np.linspace(0,tf,Nt+1,retstep=True)
    dt = fa.Constant(dt)
    Ta = fa.Constant(20.) # Ambient Temperature.
    v = fenics.as_vector([fa.Constant(0.), fa.Constant(0.)]) # Wind vector.


    # Define function space.
    mesh = fa.RectangleMesh(fenics.Point(x1, y1), fenics.Point(x2, y2), \
                    Nx, Ny, 'crossed')
    V = fenics.FunctionSpace(mesh, 'CG', 1)
```

```
# Define initial values.
T_0 = fenics.Expression('1200/cosh(pow(x[0]-b,2)+pow(x[1]-b,2))+Ta+0.1',
                        degree=2, b=4, Ta=Ta)
T_n = fa.interpolate(T_0, V)
funcstring = "(pow(x[0]-5,2)+pow(x[1]-5,2) <= 1.0)? 1.0 : 0.0;"
S_0 = Expression(funcstring, degree=2)
S_n = fa.interpolate(S_0, V)

# Define Trial and Test Functions.
T_n1 = fenics.TrialFunction(V)
S_n1 = fenics.TrialFunction(V)
q = fenics.TestFunction(V)
p = fenics.TestFunction(V)

# Define Boundary Conditions: BC = dot(grad(T_n),N)
# where N is the unit normal vector.
BC = fa.Expression("0.", degree=1) # Pure Neumann BCs.

# Define variational forms.
T_diff = T_n-Ta
linearized = exp(-B/T_diff) + B*exp(-B/T_diff)/(T_diff**2)*dt
F_T = (T_n1-T_n)*q*dx \
        + dt*(k*dot(grad(T_n1),grad(q)) + dot(v,grad(T_n1))*q \
        - A*(S_n*linearized*q - C*(T_n1-Ta)*q))*dx \
        + (k*BC*q)*ds
F_S = (S_n1-S_n)*p*dx + dt*Cs*S_n*linearized*p*dx
```

```
# Separate into bilinear and linear form.
a_T, L_T = lhs(F_T), rhs(F_T)
a_S, L_S = lhs(F_S), rhs(F_S)

# Step in time.
T = fa.Function(V)
S = fa.Function(V)
for i in range(len(t)):
    # Solve for T, update solution.
    fa.solve(a_T==L_T, T)
    T_n.assign(T)
    # Solve for S, update solution
    fa.solve(a_S==L_S, S)
    S_n.assign(S)
return T,S
```

Appendix B. Additional Figures

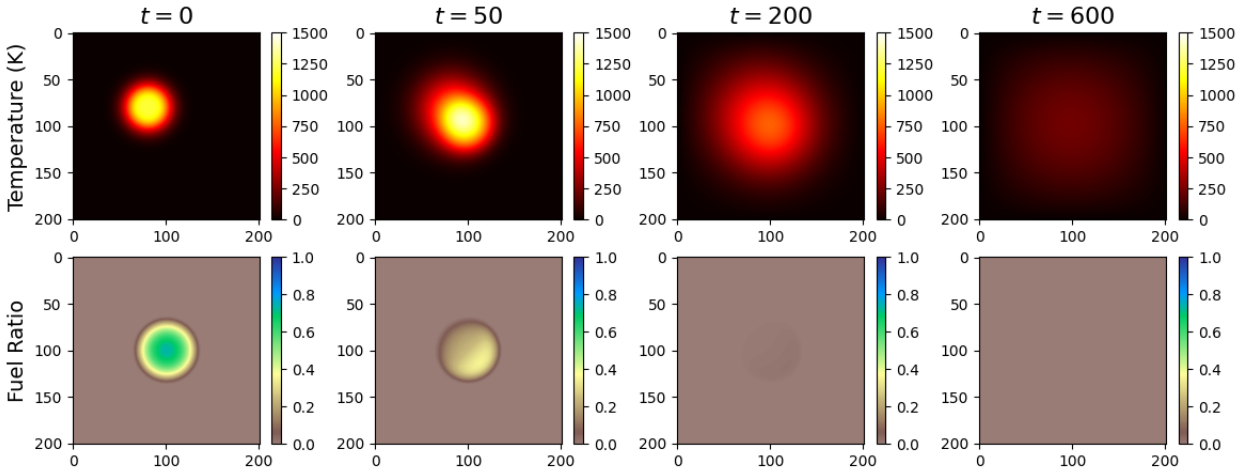

Figure B.1: 2-Dimensional Crank-Nicolson Example Heat Map (No Wind Present)

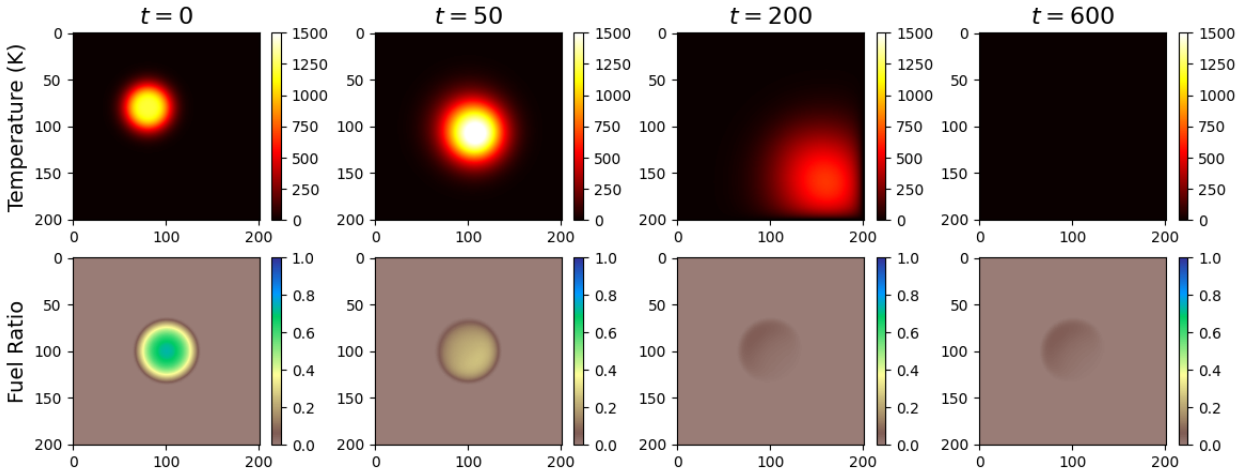

Figure B.2: 2-Dimensional Crank-Nicolson Example Heat Map with Wind

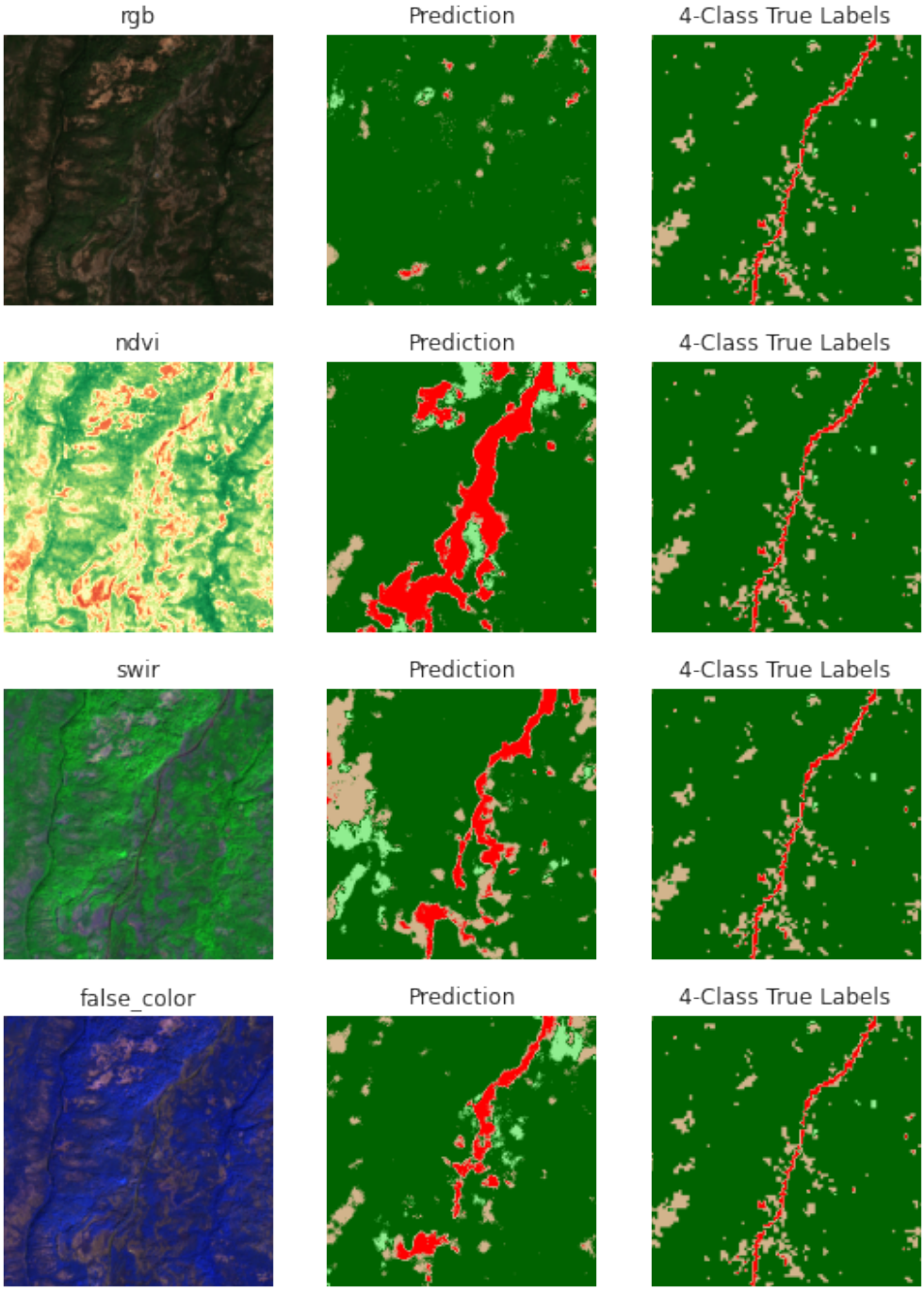

Figure B.3: Comparing Results Using RGB, NDVI, SWIR, and Agriculture Bands

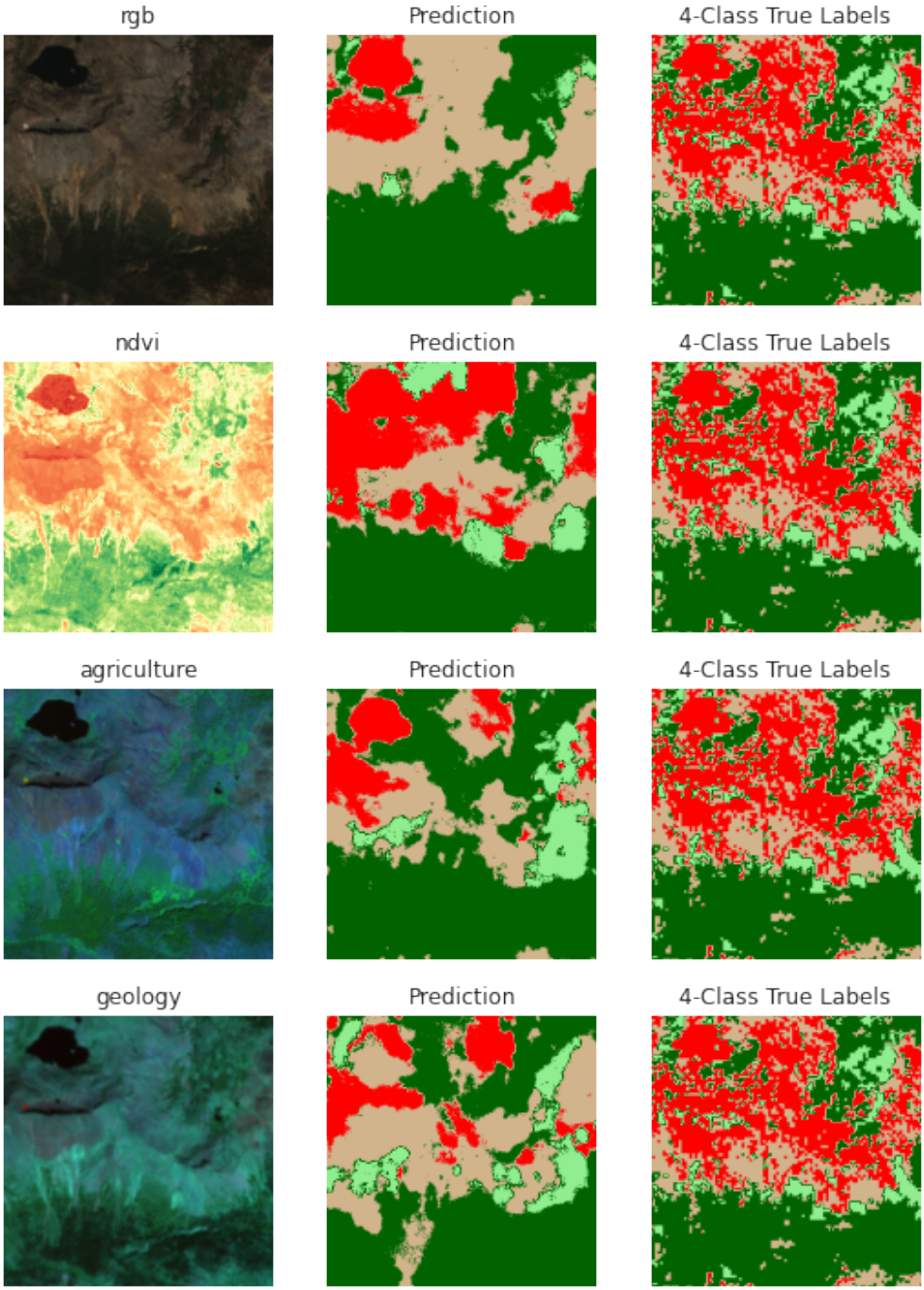

Figure B.4: Comparing Results Using RGB, NDVI, Agriculture, and Geology Bands

Bibliography

[1] Jan Mandel, Lynn S Bennethum, Jonathan D Beezley, Janice L Coen, Craig C Douglas, Minjeong Kim, and Anthony Vodacek. A wildland fire model with data assimilation. *Mathematics and Computers in Simulation*, 79(3):584–606, 2008.

[2] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[3] Alejandra Borunda. 'Megadrought' persists in western U.S., as another extremely dry year develops. `https://www.nationalgeographic.com/environment/article/megadrought-persists-in-western-us-as-another-extremely-dry-year-develops`, May 2021. Accessed: 2022-11-16.

[4] USDA Forest Service. Wildfire hazard potential. `https://www.firelab.org/project/wildfire-hazard-potential`, 2022. Accessed: 2022-11-14.

[5] Megan Healy. Cal poly researchers ready to tackle california's catastrophic fires with new institute, Mar 2019. Accessed: 2022-11-14.

[6] Mark A Finney and Patricia L Andrews. Farsite—a program for fire growth simulation. *Fire management notes*, 59(2):13–15, 1999.

[7] Alyson Kenward, Todd Sanford, and James Bronzan. *Western wildfires: A fiery future.* Climate Central, 2016.

[8] National Interagency Fire Center. 2020 National Large Incident YTD Report. `https://commons.wikimedia.org/wiki/File:2020_National_Large_Incident_YTD_Report.pdf`, December 2020. Accessed: 2022-11-18.

[9] National Interagency Fire Center. 2022 National Large Incident YTD Report. `https://gacc.nifc.gov/sacc/predictive/intelligence/NationalLargeIncidentYTDReport.pdf`, November 2022. Accessed: 2022-11-18.

[10] National Interagency Coordination Center. National fire news. `https://www.nifc.gov/fire-information/nfn`. Accessed: 2022-11-18.

[11] Matthew P Thompson, David E Calkin, Mark A Finney, Alan A Ager, and Julie W Gilbertson-Day. Integrated national-scale assessment of wildfire risk to human and ecological values. *Stochastic Environmental Research and Risk Assessment*, 25(6):761–780, 2011.

[12] A Park Williams, John T Abatzoglou, Alexander Gershunov, Janin Guzman-Morales, Daniel A Bishop, Jennifer K Balch, and Dennis P Lettenmaier. Observed impacts of anthropogenic climate change on wildfire in california. *Earth's Future*, 7(8):892–910, 2019.

[13] CAL FIRE and the State of California. Top 20 largest california wildfires. `https://www.fire.ca.gov/media/4jandlhh/top20_acres.pdf`, October 2022. Accessed: 2022-11-18.

[14] Diana Leonard and Andrew Freedman. Western wildfires: An 'unprecedented,' climate change-fueled event, experts say, Sep 2020.

[15] David Louie. Damage from california's wildfires estimated at $10 billion, experts say. *San Francisco, CA, USA*, 2020.

[16] CAL FIRE and the State of California. Dixie fire incident. `https://www.fire.ca.gov/incidents/2021/7/13/dixie-fire/`, October 2022. Accessed: 2022-11-18.

[17] Colby Bermel. Dixie fire becomes largest single wildfire in california history. `https://www.politico.com/states/california/story/2021/08/06/dixie-fire-becomes-largest-single-wildfire-in-california-history-1389651`, Aug 2021.

[18] LANDFIRE 1.1.0, U.S. Department of the Interior, U.S. Geological Survey, and U.S. Department of Agriculture. Landfire mosaic data products. `https://landfire.gov/version_download.php`, 2022. Accessed: 2022-11-18.

[19] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[20] Anders Logg and Garth N Wells. Dolfin: Automated finite element computing. *ACM Transactions on Mathematical Software (TOMS)*, 37(2):1–28, 2010.

[21] Matthew W Scroggs, Jørgen S Dokken, Chris N Richardson, and Garth N Wells. Construction of arbitrary order finite element degree-of-freedom maps on polygonal and polyhedral cell meshes. *ACM Transactions on Mathematical Software (TOMS)*, 48(2):1–23, 2022.

[22] Martin S Alnæs, Anders Logg, Kristian B Ølgaard, Marie E Rognes, and Garth N Wells. Unified form language: A domain-specific language for weak formulations of partial differential equations. *ACM Transactions on Mathematical Software (TOMS)*, 40(2):1–37, 2014.

[23] Drew Johnston. Wildfire modeling. `https://github.com/drewjohnston13/wildfire_modeling`, 2022.

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[25] Richard C Rothermel. *A mathematical model for predicting fire spread in wildland fuels*, volume 115. Intermountain Forest and Range Experiment Station, USDA Forest Service, US Department of Agriculture, 1972.

[26] CE Van Wagner. Conditions for the start and spread of crown fire. *Canadian Journal of Forest Research*, 7(1):23–34, 1977.

[27] Richard C Rothermel. *Predicting behavior and size of crown fires in the Northern Rocky Mountains*, volume 438. Intermountain Forest and Range Experiment Station, USDA Forest Service, US Department of Agriculture, 1991.

[28] Frank A Albini. *Spot fire distance from burning trees: a predictive model*, volume 56. Intermountain Forest and Range Experiment Station, USDA Forest Service, US Department of Agriculture, 1979.

[29] Ralph M Nelson Jr. Prediction of diurnal change in 10-h fuel stick moisture content. *Canadian Journal of Forest Research*, 30(7):1071–1087, 2000.

[30] Frank A Albini and Elizabeth D Reinhardt. Modeling ignition and burning rate of large woody natural fuels. *International Journal of Wildland Fire*, 5(2):81–91, 1995.

[31] Elsa Pastor, Luis Zárate, Eulalia Planas, and Josep Arnaldos. Mathematical models and calculation systems for the study of wildland fire behaviour. *Progress in Energy and Combustion Science*, 29(2):139–153, 2003.

[32] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[33] John Salvatier, Thomas V Wiecki, and Christopher Fonnesbeck. Probabilistic programming in python using pymc3. *PeerJ Computer Science*, 2:e55, 2016.

[34] Edgar Gabriel, Graham E Fagg, George Bosilca, Thara Angskun, Jack J Dongarra, Jeffrey M Squyres, Vishal Sahay, Prabhanjan Kambadur, Brian Barrett, Andrew Lumsdaine, et al. Open mpi: Goals, concept, and design of a next generation mpi implementation. In *European Parallel Virtual Machine/Message Passing Interface Users' Group Meeting*, pages 97–104. Springer, 2004.

[35] P. R. Amestoy, I. S. Duff, J. Koster, and J.-Y. L'Excellent. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2001.

[36] P. R. Amestoy, A. Guermouche, J.-Y. L'Excellent, and S. Pralet. Hybrid scheduling for the parallel solution of linear systems. *Parallel Computing*, 32(2):136–156, 2006.

[37] Earth Resources Observation and Science (EROS) Center. Sentinel. `https://www.usgs.gov/centers/eros/science/usgs-eros-archive-sentinel-2?qt-science_center_objects=0#qt-science_center_objects`, 2017. Accessed: 2021-9-1.

[38] LANDFIRE 1.1.0, U.S. Department of the Interior, U.S. Geological Survey, and U.S. Department of Agriculture. Landfire 2020 fuel vegetation type layer. http://landfire.cr.usgs.gov/viewer/, 2020. Accessed: 2021-10-1.

[39] Sha Huang, Lina Tang, Joseph P Hupy, Yang Wang, and Guofan Shao. A commentary review on the use of normalized difference vegetation index (ndvi) in the era of popular remote sensing. *Journal of Forestry Research*, 32(1):1–6, 2021.

[40] National Interagency Fire Center. FTP Server CALFIRE Data. https://ftp.wildfire.gov/public/incident_specific_data/calif_n/!CALFIRE/2021_Incidents/CA-BTU-009205_Dixie/, 2021. Accessed: 2021-10-12.

[41] Jeffrey Chao. *Evaluating the Use of Various Distance Metrics For Assessing a Model's Wildfire Prediction Performance*. University of California, Los Angeles, 2018.

[42] George Gaylord Simpson. Mammals and the nature of continents. *American Journal of Science*, 241(1):1–31, 1943.

[43] Josias Braun-Blanquet et al. Plant sociology. the study of plant communities. *Plant sociology. The study of plant communities. First ed.*, 1932.

[44] S Kulczynski. Plant complexes in the pieniny. *Int. Newslett. Porlish Acad. Sci. Lett., Class Math. Natural Sci.*, 2:203–257, 1927.

[45] National Centers for Environmental Information, National Oceanic and Atmospheric Administration, U.S. Air Force, and Federal Aviation Agency of the U.S. Department of Transportation. U.S. local climatological data. https://www.ncei.noaa.gov/metadata/geoportal/rest/metadata/item/gov.noaa.ncdc:C00684/html#.

[46] Mike Wotton. Grass fire behavior and flame. http://www.firelab.utoronto.ca/behaviour/grass_fire.html. Accessed: 2022-11-19.

[47] Jonathan P Dandois and Erle C Ellis. Remote sensing of vegetation structure using computer vision. *Remote sensing*, 2(4):1157–1176, 2010.

[48] Krishna Rao, A Park Williams, Jacqueline Fortin Flefil, and Alexandra G Konings. Sar-enhanced mapping of live fuel moisture content. *Remote Sensing of Environment*, 245:111797, 2020.

[49] Guangxiong Peng, Jing Li, Yunhao Chen, Abdul Patah Norizan, and Liphong Tay. High-resolution surface relative humidity computation using MODIS image in Peninsular Malaysia. *Chinese Geographical Science*, 16(3):260–264, 2006.

[50] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.