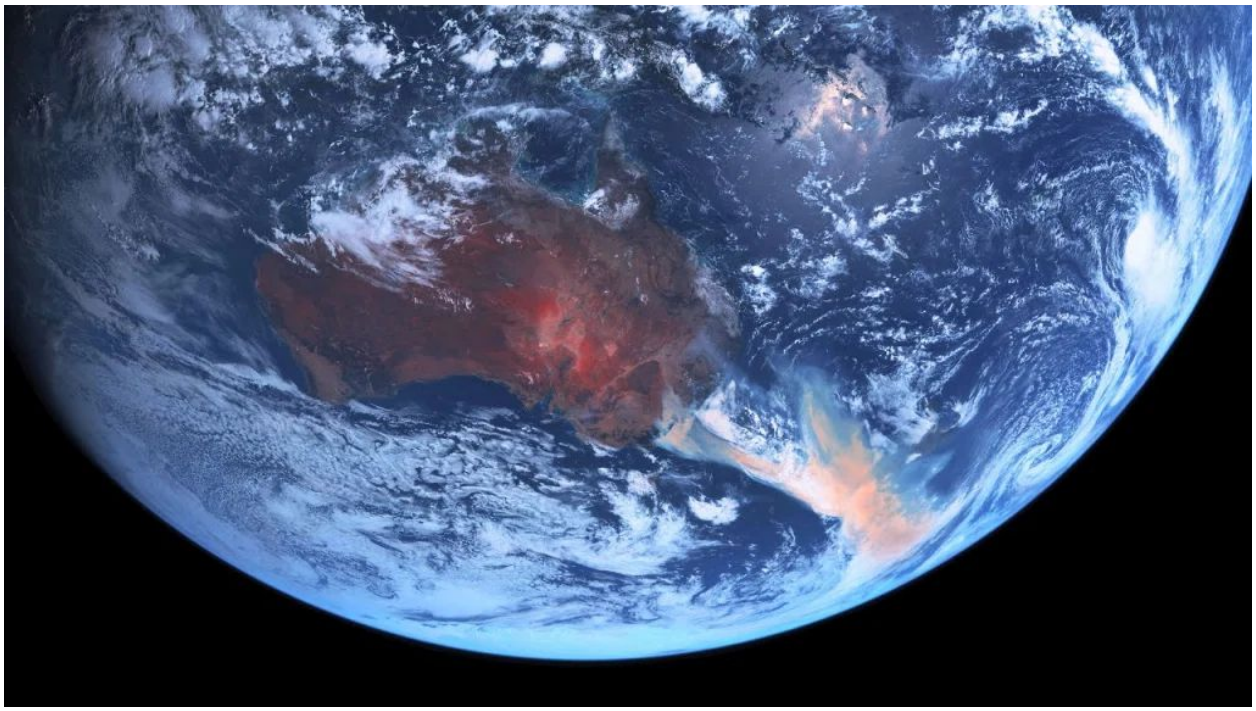
Machine Learning to detect Forest Fires

Australian Wildfires 2019

This year's fires in Australia have been worse than in many years before, because of dry, hot conditions. Climate change has been the reason for these fires, found in every state. New South Wales has been hit the hardest, with over 3,000 damaged homes and many of the 28 fatalities from around the country. The smoke from the fires settled over many big cities, including Sydney, and bringing the air to very toxic levels, not just for humans but also for animals living in burning areas, where habitat loss is also an issue.

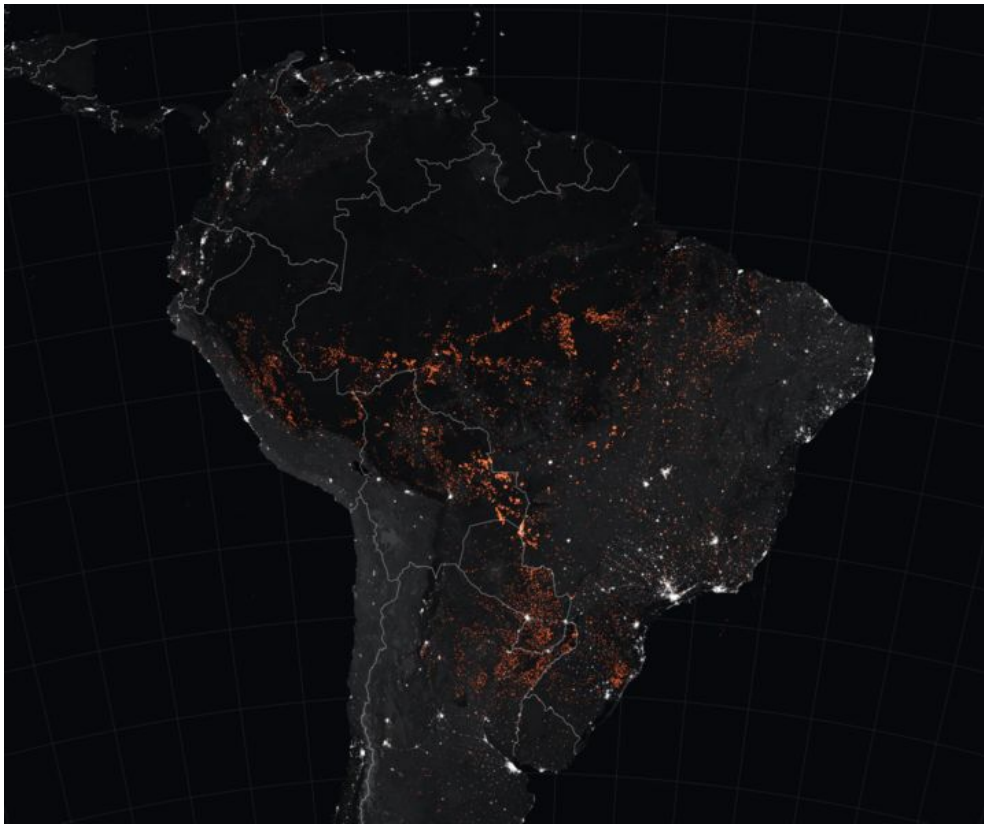
A billion wild animals have been caught in the fire according to some estimates and the ecology of the continent as well as of the world is in danger! There are species that might go extinct due to these wildfires!



NASA satellite images of Australian wildfires, courtesy of imgur.com

Amazon Rainforest Wildfires, 2019

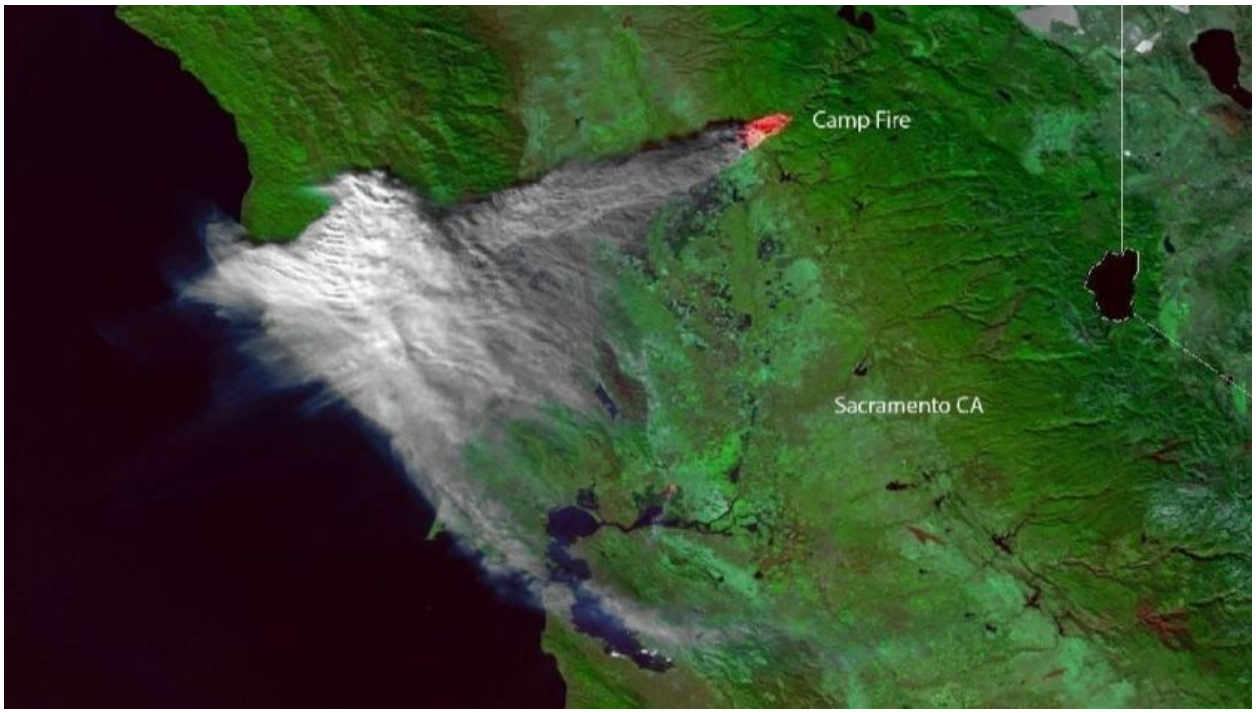
Fires in 2019 started in the Amazonian rainforests, caused by a slash and burn approach by farmers to deforest land for commercial use. The wildfires started on August 15, 2019 and persisted for months. Over 76,000 fires in the Amazon destroyed approximately 2,240,000 acres of forest.



MODIS satellite image of Amazonian fires, courtesy of earthobservatory.nasa.gov

California Forest Fires, 2018

Forest and bush fires are not uncommon in California, due to the arid climate. The worst of the many fires began in the middle of summer, and continued for months, only to have more big fires start in the months of October and November. In total, approximately 1.9 million acres of land were destroyed by the more than 8500 fires that burned throughout California, adding up to 3.5 billion dollars owed for forest fire prevention and property damage, not to mention the 103 civilian and firefighter fatalities. Air quality in the surrounding areas also reached toxic levels in the summer months restricting many residents to their homes.



Smoke from the Camp Fire spreading across Northern California, Image taken on November 8, 2018, courtesy of sandiegouniontribune.com

Greece, 2018

In mid-July 2018, two forest fires simultaneously began burning in the Administrative Region of Attica in Greece. The fires were located near multiple towns and cities, including Kineta, Rafina, and Mati, populated by tourists as well as locals, all of whom had to quickly evacuate and run or drive away from the fire. Some had to run to the ocean and swim, but many did not make it to

rescue boats brought to the shores. The cause of the fire has been labeled negligent arson, with one of the fires started by a man burning wood in his backyard, which ended up killing 102 people fleeing from the flames that grew massively in high wind speeds, tearing through every building and vehicle in the way.

People watching the fast moving blaze in Rafina, courtesy of nytimes.com

2019 Siberian Wildfires

The Siberian Wildfires started in July 2019. Since then, these massive wildfires in the most inaccessible northern part of Russia have burned more than 10 million acres of land. There have always been some fires in Siberia around this time in past years, but this most recent is much bigger than the rest. Russia ignored these fires at first, and didn't start fighting the fire until it had turned into a climate emergency. The carbon produced by the burning is having effects on the arctic glaciers, accelerating the melting of ice, and threatening the climate of earth.

Russian military aiding firefighters in Siberia, courtesy of spokesman.com

Bolivia 2010

The 2010 Bolivia Wildfires caused the country's government to declare a state emergency. Bolivia wasn't able to prevent the fires properly from spreading because they lacked water supply which is one reason why the fire ended up getting so big. There were over 25,000 fires in Bolivia that burned over 3,700,000 acres of land. The most affected region of the fire was the Amazonian province in the north of Bolivia.

SuperTanker aircraft aiding the fight against Bolivian wildfires, courtesy of riotimesonline.com

Reasons for forest fire:

Lightning is a common natural way that forest fires can start. These can be especially dangerous because if lightning strikes in a remote area, it can be hard to prevent and detect them quickly.

Vegetation can catch on fire in very dry and hot areas, so even the smallest spark can catch the vegetation on fire. The fire then spreads very quickly through all the surrounding vegetation.

Volcanic activity can cause the wilderness around the mountain to catch on fire as lava leaks out of the volcano. However, prevention measures can be implicated before eruption, so as do decrease the risk of a wildfire.

Careless disposal of cigarettes can easily start a fire. If not properly put out, it will keep burning on the forest floor, and can spark other fires.

Candles and campfires can be safely used in a forest, but must be properly controlled. If the candle tips over or an ember falls out of the fire, it can very quickly get out of control.

Also, arson can be the cause of a forest fire. People doing things like setting off fireworks in the forest, or attempting to set vegetation on fire using a lighter or matches, for example.

Ideas for forest fire prevention and control:

Prevention:

Forest fires nowadays are big threats to the environment and ecosystems and are extremely unpredictable but there are specific ways to prevent forest fires.

One way to prevent or reduce forest fires is to make more laws/rules for using forests as commercial use and other stuff, these are also one of the main reasons why the Amazon was on fire.

Another way is to heighten the level of security and censoring near or in forested areas to protect the forests from illegal loggers and farmers. Here are some more ways to prevent fires:

- Having firefighting tools in the area just incase an unexpected fire sparks in vegetation.
- Storing flammable objects away from vegetation so that nothing catches on fire.
- Using weather forecasting technology to predict lightning storms and other weather that could set the forest on fire.
- Having a security guard or guards checking the area once in awhile.
- Have a fire alarm that also alarms the fire station immediately whenever there is a fire.

- More laws for only commercial campfire use e.g. not made with stones and sticks
- Permits needed to have open or contained fire
- Thorough searching and protection of forests
- Networks of camouflaged cameras in high use and back-country areas with paths
- Looking for fire detection through cameras
- Close the forest trails for public use when camera detects fire danger or trespassing, privacy of people
- Cameras can also be used to track endangered and protected species, multiple uses
- Use heat detection, especially at campsites
- Letting underbrush naturally burn, and closing areas during this burning season to let nature do its thing
- Carefully extinguishing smoking materials
- Education is also important when it comes to forest fires. Lots of people don't know there are things everyone can do and help with to prevent major forest fires.

- Classes in schools should teach about Forest fire safety and what to do in case somebody spots forest fires

- Kids and teenagers should be taught about the danger of using fireworks etc. in forests especially in dry weather conditions. Quite a few brush fires in California got started this way

Control once fire starts:

- Early Detection is the key. The faster the fires are detected/discovered, the faster they can be put out causing less damage to property and life

- Shut off electricity if needed - if electrical lines go through forest and there are heavy winds, it might break the live electrical wires and cause more fires (Happened in California)

- Depending on the weather conditions, use technologies like drones to figure out how fast the fire is spreading and deploy firemen and aerial water drops.

Early Detection

A key aspect of controlling forest fires is detecting the fire before it becomes a big, fast moving blaze that uses many more resources and causes more damage to the environment and wildlife. If we think of more ways to react quickly when a small fire in an area of dense foliage starts, not as much effort is needed to control the fire that would be to come.

There are many different tactics that can be used to find these small fires in prone areas. Forest Service Rangers can use different tools to find and be aware of these fires in the area they are in charge of, and they can also help the visitors and campers in the forest to be on the watch for smoke and uncontrolled fire, especially around campsites.

Forest Ranger offices could have long distance scopes or sensors, that could see and detect smoke in the surrounding wilderness. Routine checking of the forest through the scope from buttes and small mountains surrounded by forest could reveal smoke, much like forest fire watchtowers, except that they would be portable and usable anywhere.

Actively checking campsites is very simple but can be an effective solution. It doesn't even need to be only people officially in charge of forest fire safety. ***People camping or hiking for the day could sign up for volunteering to watch the areas they stay in.*** If people started to sign up for this more, it could make a big impact on the early detection of fires.

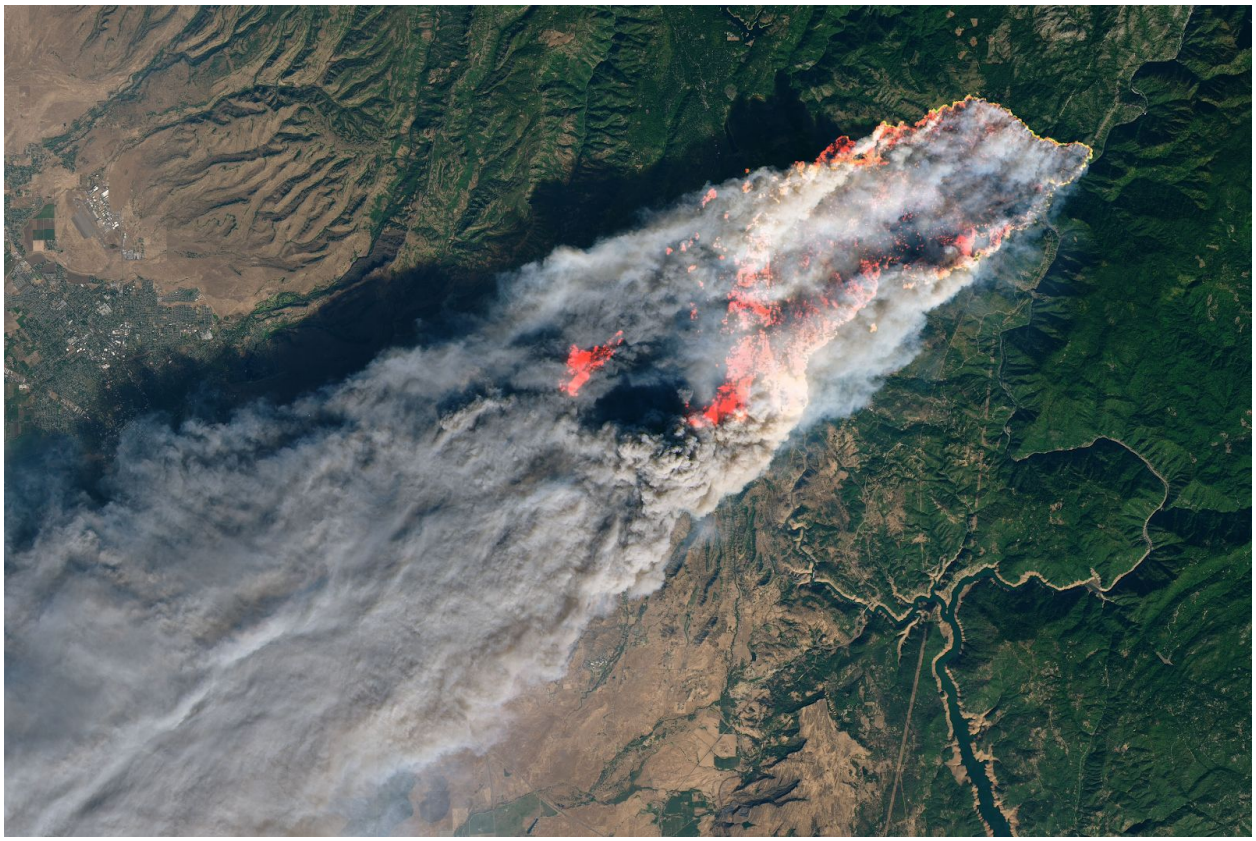
Thinking about ways to keep a better eye on forests overall is challenging, because it's very hard to watch so many millions of forests with accuracy at the same time. ***Then came the idea of using a satellite!*** Satellites can take numerous images on a large scale, which is the perfect thing for spotting fires and smoke.

The satellite can identify the locations of fires, which can be used in conjunction with historical data on forest fires. Then we can analyze the data found by the satellite images to find fires in areas more likely to be at risk. We could find patterns that would lead to predictions of future fires. This could be an efficient, accurate, and new way to detect fires before they cause more suffering for humans, animals, plants, and our environment.

Cameras that can detect smoke and fire could also be set up in high use areas of trails and campsites. They would be the first to notice and notify officials, because they are watching all the time. Cameras would be most useful in areas with many visitors, where most accidents happen. In the United States, about 84% of fires begin because of humans, whether intentional or not. These cameras could be positioned in places like treetops, and unlike satellites, they would be used more for immediate and early detection. Heat and smoke detecting cameras would be the short term alternative to analyzing satellite images or using a Convolutional Neural Network to identify fires.

Satellite Data:

Images:



Picture courtesy of Space.com of wild fire in Northern California

Pros:

- Images and/or videos are real time
- There are lot of satellites up there so at any given point of time, there will be some satellite(s) that can be used to take real-time pictures if needed
- Images show exactly where the fire is

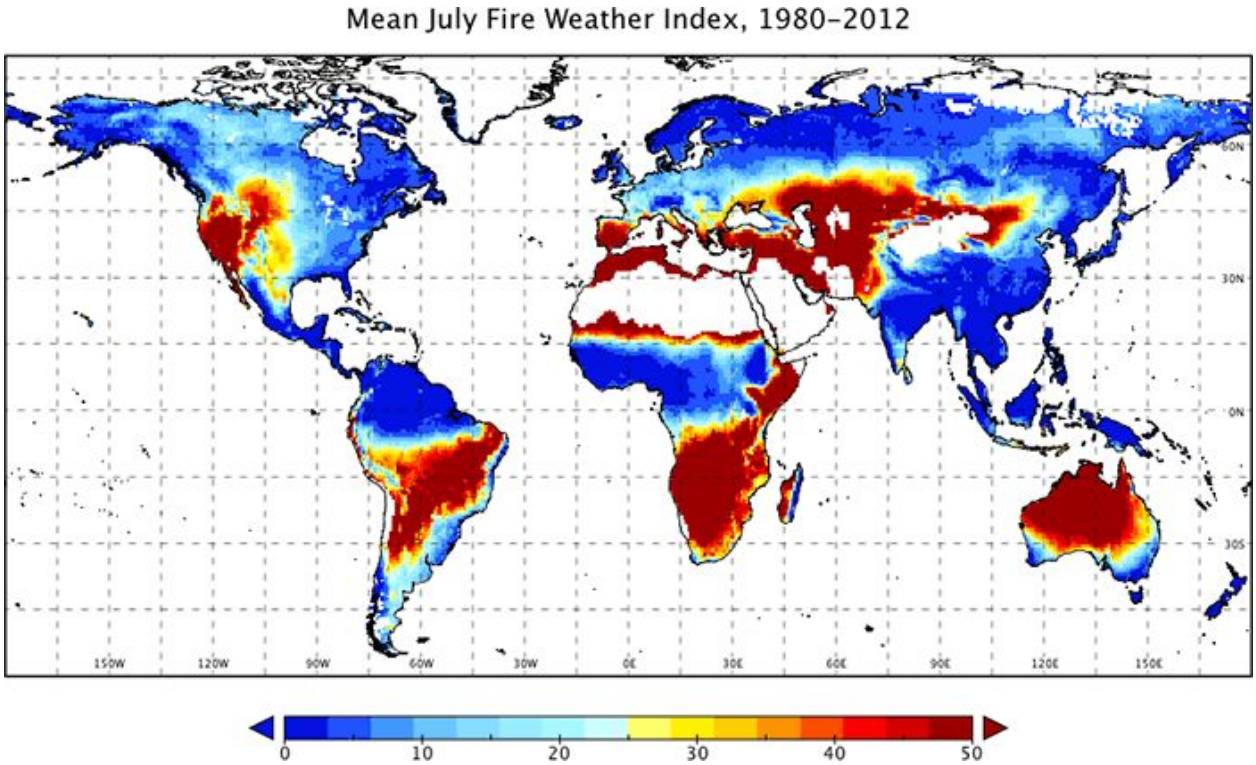
- Images can show how much fire already spread

- Images can give some detail about the surrounding areas of the fire to assess how fast the fire will spread
- Images can be downloaded easily. Even if they are videos, frames from videos can be taken

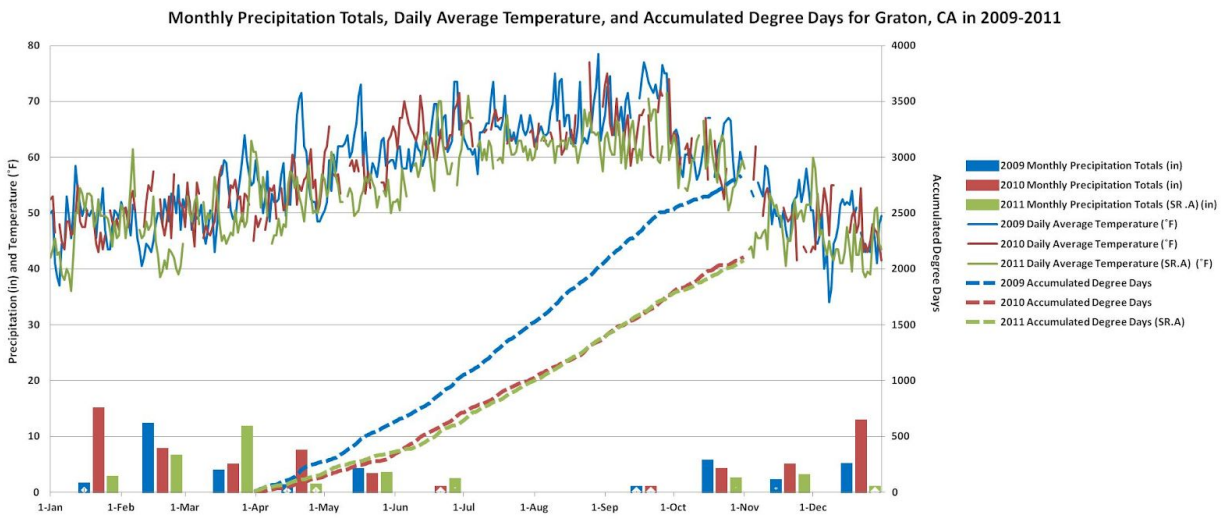
Cons:

- Takes lot of time to go through imagery to figure out if there is a fire. It could take days before a set of images can be used.
- Continuous monitoring is almost impossible to do around the world

Satellite weather data:



Picture courtesy of NASA Global Fire WEather Database



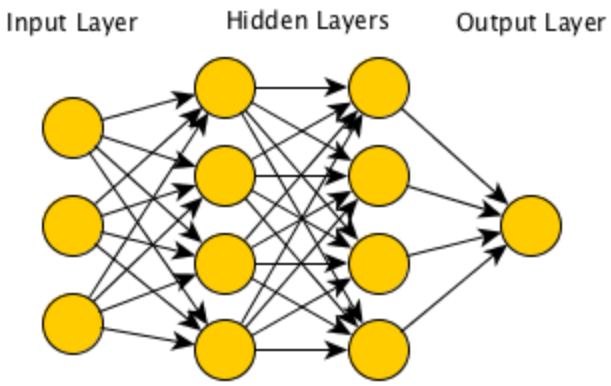
Picture courtesy of UCCE Sonoma county

Historical weather data can be used to check and predict the conditions of the forest. This data, then can be used to predict how fast the fires might spread and how much land it might burn. Even with faster computers, processing such information when an event happens takes a long time.

With the new developments in Machine Learning especially in the area of Neural Networks, it might actually be possible to detect the fires early and control them faster.

Neural networks

Neural networks are computers simulating neurons to construct an algorithm, based on inputs, outputs, and expected outputs. Each neuron is a node, taking a fixed amount of inputs, assigning weights, and then applying an activation function (For example, ReLU). The weights are assigned based on the loss function, the difference between the expected output and the output. They usually have multiple layers (stacked layers), starting with input layer, and ending with output layer. Hidden layers are in between input and output layers. The input layer takes as many inputs as specified, proceeds to the hidden layers, with each input node connected to each node in the hidden layer. Finally, it reaches the output layer, where it outputs based on the expected output.



(1) Deep Neural Network Model

Deep Neural Networks are basically Stacked Neural Networks and give us more sophisticated training opportunities. Deep Neural Networks have several advantages over simple regression analysis. Regression analysis is good for smaller databases, with less possible exceptions and no training. Larger databases have more exceptions, can adapt and train to create a more accurate model.

A branch of Deep Neural Networks, called Convolutional Neural Networks, are what is used for image classification. The network is identifying fire in photos given as inputs. This is precisely suited to identifying forest fires, as the network acts as a human brain trained only for identifying fire. It does not require breaks and can monitor different cameras simultaneously.

Neural Networks identify grouped pixels (Grouped using Max-Pooling) in images similar to the input training images. For sections of images close to the edge, a technique called zero-padding is applied. When the pixels are max-pooled, the edges leave gaps. These gaps are filled with zero, hence the name zero padding. With thousands of input images, the network learns to identify pixel values corresponding to fire and smoke. In this way, the network can identify fire in all images.

Various models can be used for image classification through transfer trait learning. Transfer trait learning is using a pre-trained network, and changing the inputs and outputs. The network still functions with similar accuracy, yet outputting the intended result. Image classification networks such as MobileNetV2, DenseNet, ResNet, and AlexNet.

The image dataset that was utilized in training this network was taken from . It contains images of fires in various environments and situations, as well as images without fire. This way the network can compare images of fire against images without fire and identify those pixels.

Files were imported to Google Colab using Google Drive. The code mounted at My Drive, and navigated to the database master file. The database master file was split into three sections, Train, Validation, and Test. Each file sectioned file was split into two categories, Fire and Nofire. The fire folder contained images of environments containing fire, while Nofire consisted of environments without fire.

```
[ ]  1 import pathlib
     2
     3 import matplotlib.pyplot as plt
     4 import numpy as np
     5 import pandas as pd
     6 import os
     7 import matplotlib.pyplot as plt
     8 from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
[ ]  1 from google.colab import drive
     2 drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
```

```
[ ]  1 base_dir = os.path.join(os.path.dirname('/content/gdrive/My Drive/'), 'data')
     2 train_dir = os.path.join(base_dir, 'train')
     3 validation_dir = os.path.join(base_dir, 'validation')
     4 print(train_dir)
     5 train_fire_dir = os.path.join(train_dir, 'fire')  # directory with fire pictures
     6 print(train_fire_dir)
     7 train_nofire_dir = os.path.join(train_dir, 'nofire')  # directory with no-fire pictures
     8 validation_fire_dir = os.path.join(validation_dir, 'fire')  # directory with ofire pictures
     9 validation_nofire_dir = os.path.join(validation_dir, 'nofire')  # directory with no-fire pictures
    10 print(train_nofire_dir)
```

```
/content/gdrive/My Drive/data/train
/content/gdrive/My Drive/data/train/fire
/content/gdrive/My Drive/data/train/nofire
```

Total number of training images is 1923. 955 of these training images were images of forests and vegetation on fire. 968 of the training files were images of forests without fire. The database included 102 validation Fire images, and 80 validation Nofire images. 178 test images were used to evaluate the performance of the network.

```
[ ]  1 num_fire_tr = len(os.listdir(train_fire_dir))
     2 num_nofire_tr = len(os.listdir(train_nofire_dir))
     3
     4 num_fire_val = len(os.listdir(validation_fire_dir))
     5 num_nofire_val = len(os.listdir(validation_nofire_dir))
     6
     7 total_train = num_fire_tr + num_nofire_tr
     8 total_val = num_fire_val + num_nofire_val
```

```
[ ]  1 print('total training fire images:', num_fire_tr)
     2 print('total training nofire images:', num_nofire_tr)
     3
     4 print('total validation fire images:', num_fire_val)
     5 print('total validation nofire images:', num_nofire_val)
     6 print("--")
     7 print("Total training images:", total_train)
     8 print("Total validation images:", total_val)
```

```
total training fire images: 955
total training nofire images: 968
total validation fire images: 102
total validation nofire images: 80
--
Total training images: 1923
Total validation images: 182
```

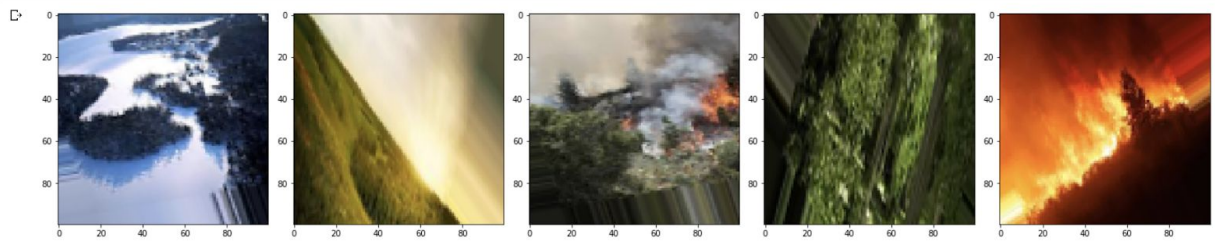
```
[ ]  1 BATCH_SIZE = 100 # Number of training examples to process before updating our models variables
     2 IMG_SHAPE  = 100  # Our training data consists of images with width of 150 pixels and height of 150 pixels
```

(2) Mounting Directories

```
[ ]  1 sample_training_images, _ = next(train_data_gen)
```

```
[ ]  1 # This function will plot images in the form of a grid with 1 row and 5 columns where images are placed in each column.
     2 def plotImages(images_arr):
     3     fig, axes = plt.subplots(1, 5, figsize=(20,20))
     4     axes = axes.flatten()
     5     for img, ax in zip(images_arr, axes):
     6         ax.imshow(img)
     7     plt.tight_layout()
     8     plt.show()
```

```
[ ]  1 plotImages(sample_training_images[:5])  # Plot images 0-4
```



(3) Sample images, including both Fire and Nofire

Modifications were made to the images so the network is more versatile in classifying images (4). A random rotation was added, with a range of 45°. A random zoom was also applied, as well as horizontal flip. An input pipeline was created with data in order to easily feed the data into the network. This was also done for validation data.

```
[ ]  1 train_image_generator      = ImageDataGenerator(rescale=1./255)  # Generator for our training data
     2 validation_image_generator = ImageDataGenerator(rescale=1./255)  # Generator for our validation data
```

```
[ ]  1 image_gen_train = ImageDataGenerator(
     2                     rescale=1./255,
     3                     rotation_range=45,
     4                     width_shift_range=.15,
     5                     height_shift_range=.15,
     6                     horizontal_flip=True,
     7                     zoom_range=0.5
     8                     )
     9
    10
    11 train_data_gen = image_gen_train.flow_from_directory(batch_size=BATCH_SIZE,
    12                                                      directory=train_dir,
    13                                                      shuffle=True,
    14                                                      target_size=(IMG_SHAPE,IMG_SHAPE),
    15                                                      class_mode='binary')
```

```
Found 1923 images belonging to 2 classes.
```

```
[ ]  1 val_data_gen = validation_image_generator.flow_from_directory(batch_size=BATCH_SIZE,
     2                                                               directory=validation_dir,
     3                                                               shuffle=False,
     4                                                               target_size=(IMG_SHAPE,IMG_SHAPE),
     5                                                               class_mode='binary')
```

```
Found 182 images belonging to 2 classes.
```

(4) Modifying Images and creating Input Pipeline

Model layers were defined in this segment of code (5). 5 convolutional layers and one output layer were defined. The first layer has 32 neurons, the second 64, the third, fourth, and fifth all have 120 neurons. All of the convolutional layers have the activation ReLu, split RGB, and Max-Pooling was performed. The output layer has two dense layers, the first containing 256 neurons, and the second containing 512 neurons. The dropout rate is 0.2 (Percentage of neurons to drop). The last layer has 2 neurons, and the activation Softmax (converts into a number between 1 and 2, essentially a boolean value) (6).

```
[ ]
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(100, 100, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),

    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(2,activation='softmax')
])
```

```
[ ]
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

(5) Model layers

```
[ ]  1 model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 98, 98, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 49, 49, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 47, 47, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 23, 23, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 21, 21, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 10, 10, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 8, 128)         147584
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 2, 2, 128)         147584
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 1, 1, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 256)               33024
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               131584
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 1026
=================================================================
Total params: 554,050
Trainable params: 554,050
Non-trainable params: 0
_________________________________________________________________
```

(6) Model summary

The model was trained with 50 epochs, with 20 training steps and 2 validation steps. It achieved an accuracy of 92.51% in training accuracy, and 89.01 in validation accuracy (7). It finished with a loss value of 0.3859 in training, and 0.4113 in validation.

```
[ ]  1 EPOCHS = 50
     2 history = model.fit(
     3     train_data_gen,
     4     steps_per_epoch=int(np.ceil(total_train / float(BATCH_SIZE))),
     5     epochs=EPOCHS,
     6     validation_data=val_data_gen,
     7     validation_steps=int(np.ceil(total_val / float(BATCH_SIZE))),
     8 )
```

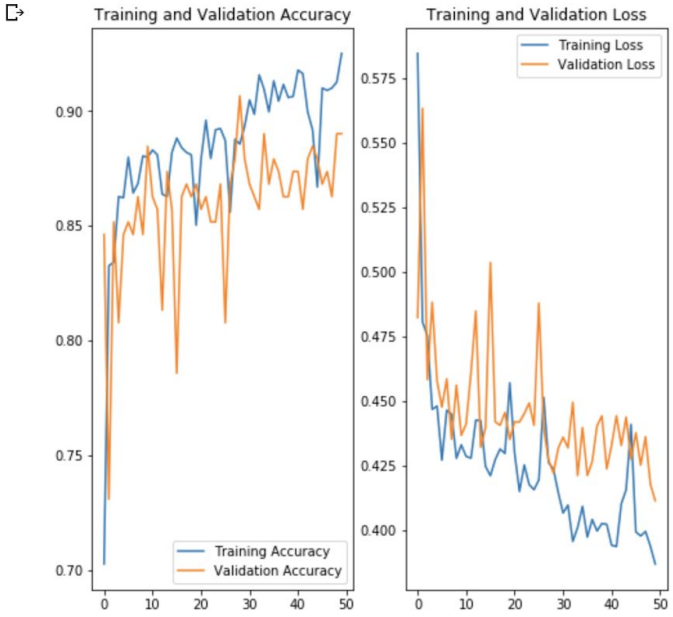
```
Train for 20 steps, validate for 2 steps
Epoch 1/50
20/20 [==============================] - 39s 2s/step - loss: 0.5855 - accuracy: 0.7025 - val_loss: 0.4822 - val_accuracy: 0.8462
Epoch 2/50
20/20 [==============================] - 39s 2s/step - loss: 0.4763 - accuracy: 0.8326 - val_loss: 0.5632 - val_accuracy: 0.7308
Epoch 3/50
20/20 [==============================] - 38s 2s/step - loss: 0.4725 - accuracy: 0.8341 - val_loss: 0.4582 - val_accuracy: 0.8516
Epoch 4/50
20/20 [==============================] - 39s 2s/step - loss: 0.4456 - accuracy: 0.8627 - val_loss: 0.4881 - val_accuracy: 0.8077
Epoch 5/50
20/20 [==============================] - 38s 2s/step - loss: 0.4464 - accuracy: 0.8622 - val_loss: 0.4577 - val_accuracy: 0.8462
Epoch 6/50
20/20 [==============================] - 38s 2s/step - loss: 0.4234 - accuracy: 0.8799 - val_loss: 0.4476 - val_accuracy: 0.8516
Epoch 7/50
20/20 [==============================] - 38s 2s/step - loss: 0.4431 - accuracy: 0.8643 - val_loss: 0.4584 - val_accuracy: 0.8462
Epoch 8/50
20/20 [==============================] - 38s 2s/step - loss: 0.4415 - accuracy: 0.8684 - val_loss: 0.4351 - val_accuracy: 0.8626
Epoch 9/50
20/20 [==============================] - 38s 2s/step - loss: 0.4255 - accuracy: 0.8804 - val_loss: 0.4560 - val_accuracy: 0.8462
Epoch 10/50
20/20 [==============================] - 38s 2s/step - loss: 0.4317 - accuracy: 0.8799 - val_loss: 0.4367 - val_accuracy: 0.8846
Epoch 11/50
20/20 [==============================] - 38s 2s/step - loss: 0.4259 - accuracy: 0.8830 - val_loss: 0.4411 - val_accuracy: 0.8626
Epoch 12/50
20/20 [==============================] - 38s 2s/step - loss: 0.4274 - accuracy: 0.8809 - val_loss: 0.4609 - val_accuracy: 0.8571
```

```
Epoch 13/50
20/20 [==============================] - 38s 2s/step - loss: 0.4394 - accuracy: 0.8638 - val_loss: 0.4847 - val_accuracy: 0.8132
Epoch 14/50
20/20 [==============================] - 38s 2s/step - loss: 0.4396 - accuracy: 0.8627 - val_loss: 0.4321 - val_accuracy: 0.8736
Epoch 15/50
20/20 [==============================] - 38s 2s/step - loss: 0.4221 - accuracy: 0.8820 - val_loss: 0.4395 - val_accuracy: 0.8571
Epoch 16/50
20/20 [==============================] - 38s 2s/step - loss: 0.4209 - accuracy: 0.8882 - val_loss: 0.5035 - val_accuracy: 0.7857
Epoch 17/50
20/20 [==============================] - 38s 2s/step - loss: 0.4262 - accuracy: 0.8840 - val_loss: 0.4418 - val_accuracy: 0.8626
Epoch 18/50
20/20 [==============================] - 38s 2s/step - loss: 0.4287 - accuracy: 0.8820 - val_loss: 0.4405 - val_accuracy: 0.8681
Epoch 19/50
20/20 [==============================] - 38s 2s/step - loss: 0.4289 - accuracy: 0.8809 - val_loss: 0.4455 - val_accuracy: 0.8626
Epoch 20/50
20/20 [==============================] - 38s 2s/step - loss: 0.4549 - accuracy: 0.8502 - val_loss: 0.4350 - val_accuracy: 0.8681
Epoch 21/50
20/20 [==============================] - 39s 2s/step - loss: 0.4270 - accuracy: 0.8788 - val_loss: 0.4418 - val_accuracy: 0.8571
Epoch 22/50
20/20 [==============================] - 38s 2s/step - loss: 0.4140 - accuracy: 0.8960 - val_loss: 0.4418 - val_accuracy: 0.8626
Epoch 23/50
20/20 [==============================] - 38s 2s/step - loss: 0.4243 - accuracy: 0.8794 - val_loss: 0.4450 - val_accuracy: 0.8516
Epoch 24/50
20/20 [==============================] - 38s 2s/step - loss: 0.4191 - accuracy: 0.8918 - val_loss: 0.4492 - val_accuracy: 0.8516
Epoch 25/50
20/20 [==============================] - 38s 2s/step - loss: 0.4140 - accuracy: 0.8924 - val_loss: 0.4405 - val_accuracy: 0.8681
Epoch 26/50
20/20 [==============================] - 38s 2s/step - loss: 0.4184 - accuracy: 0.8872 - val_loss: 0.4878 - val_accuracy: 0.8077
Epoch 27/50
20/20 [==============================] - 38s 2s/step - loss: 0.4497 - accuracy: 0.8560 - val_loss: 0.4394 - val_accuracy: 0.8681
Epoch 28/50
20/20 [==============================] - 38s 2s/step - loss: 0.4261 - accuracy: 0.8877 - val_loss: 0.4269 - val_accuracy: 0.8791
Epoch 29/50
20/20 [==============================] - 38s 2s/step - loss: 0.4213 - accuracy: 0.8856 - val_loss: 0.4221 - val_accuracy: 0.9066
Epoch 30/50
20/20 [==============================] - 38s 2s/step - loss: 0.4140 - accuracy: 0.8934 - val_loss: 0.4317 - val_accuracy: 0.8791
Epoch 31/50
20/20 [==============================] - 38s 2s/step - loss: 0.4058 - accuracy: 0.9048 - val_loss: 0.4359 - val_accuracy: 0.8681
Epoch 32/50
20/20 [==============================] - 38s 2s/step - loss: 0.4077 - accuracy: 0.8986 - val_loss: 0.4318 - val_accuracy: 0.8626
Epoch 33/50
20/20 [==============================] - 38s 2s/step - loss: 0.3974 - accuracy: 0.9158 - val_loss: 0.4494 - val_accuracy: 0.8571
Epoch 34/50
20/20 [==============================] - 38s 2s/step - loss: 0.4012 - accuracy: 0.9095 - val_loss: 0.4211 - val_accuracy: 0.8901
Epoch 35/50
20/20 [==============================] - 38s 2s/step - loss: 0.4081 - accuracy: 0.8996 - val_loss: 0.4396 - val_accuracy: 0.8681
Epoch 36/50
20/20 [==============================] - 38s 2s/step - loss: 0.3960 - accuracy: 0.9132 - val_loss: 0.4211 - val_accuracy: 0.8791
Epoch 37/50
20/20 [==============================] - 38s 2s/step - loss: 0.4036 - accuracy: 0.9043 - val_loss: 0.4264 - val_accuracy: 0.8736
Epoch 38/50
20/20 [==============================] - 38s 2s/step - loss: 0.3970 - accuracy: 0.9116 - val_loss: 0.4403 - val_accuracy: 0.8626
Epoch 39/50
20/20 [==============================] - 38s 2s/step - loss: 0.4007 - accuracy: 0.9059 - val_loss: 0.4441 - val_accuracy: 0.8626
Epoch 40/50
20/20 [==============================] - 38s 2s/step - loss: 0.4006 - accuracy: 0.9064 - val_loss: 0.4237 - val_accuracy: 0.8736
Epoch 41/50
20/20 [==============================] - 38s 2s/step - loss: 0.3917 - accuracy: 0.9178 - val_loss: 0.4327 - val_accuracy: 0.8736
Epoch 42/50
20/20 [==============================] - 38s 2s/step - loss: 0.3908 - accuracy: 0.9163 - val_loss: 0.4442 - val_accuracy: 0.8571
Epoch 43/50
20/20 [==============================] - 38s 2s/step - loss: 0.4093 - accuracy: 0.8996 - val_loss: 0.4327 - val_accuracy: 0.8791
Epoch 44/50
20/20 [==============================] - 38s 2s/step - loss: 0.4159 - accuracy: 0.8918 - val_loss: 0.4437 - val_accuracy: 0.8846
Epoch 45/50
20/20 [==============================] - 38s 2s/step - loss: 0.4379 - accuracy: 0.8669 - val_loss: 0.4275 - val_accuracy: 0.8791
Epoch 46/50
20/20 [==============================] - 38s 2s/step - loss: 0.3981 - accuracy: 0.9100 - val_loss: 0.4376 - val_accuracy: 0.8681
Epoch 47/50
20/20 [==============================] - 38s 2s/step - loss: 0.3962 - accuracy: 0.9090 - val_loss: 0.4251 - val_accuracy: 0.8736
Epoch 48/50
20/20 [==============================] - 39s 2s/step - loss: 0.3968 - accuracy: 0.9100 - val_loss: 0.4361 - val_accuracy: 0.8626
Epoch 49/50
20/20 [==============================] - 38s 2s/step - loss: 0.3931 - accuracy: 0.9126 - val_loss: 0.4178 - val_accuracy: 0.8901
Epoch 50/50
20/20 [==============================] - 38s 2s/step - loss: 0.3859 - accuracy: 0.9251 - val_loss: 0.4113 - val_accuracy: 0.8901
```

(7) Training Epochs data

Graph comparing training and validation accuracy and loss (8):

```
[ ]  1 acc = history.history['accuracy']
     2 val_acc = history.history['val_accuracy']
     3
     4 loss = history.history['loss']
     5 val_loss = history.history['val_loss']
     6
     7 epochs_range = range(EPOCHS)
     8
     9 plt.figure(figsize=(8, 8))
    10 plt.subplot(1, 2, 1)
    11 plt.plot(epochs_range, acc, label='Training Accuracy')
    12 plt.plot(epochs_range, val_acc, label='Validation Accuracy')
    13 plt.legend(loc='lower right')
    14 plt.title('Training and Validation Accuracy')
    15
    16 plt.subplot(1, 2, 2)
    17 plt.plot(epochs_range, loss, label='Training Loss')
    18 plt.plot(epochs_range, val_loss, label='Validation Loss')
    19 plt.legend(loc='upper right')
    20 plt.title('Training and Validation Loss')
    21 plt.savefig('./loss.png')
    22 plt.show()
```

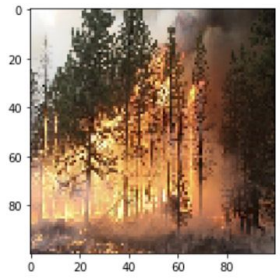


(8) Graph comparing Training and validation accuracy and loss

Testing the Neural Network

The test images used are not used in the training or validation dataset. Random forest fire and non-fire images are taken and converted into size 100x100. Below it shows the image and the result of the trained neural network.

```
[29] 1 from keras.preprocessing import image
     2 import matplotlib.pyplot as plt
     3 import matplotlib.image as mpimg
     4 test_image =image.load_img('/content/gdrive/My Drive/data/test/wildfire.jpg', target_size =(100,100))
     5 imgplot = plt.imshow(test_image)
     6
```

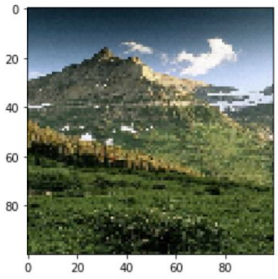



```
1 test_image =image.img_to_array(test_image)
2 test_image =np.expand_dims(test_image, axis =0)
3 result = model.predict(test_image)
4 if result[0][0] >= 0.5:
5     prediction = 'fire'
6 else:
7     prediction = 'No fire'
8 print(prediction)
```

```
fire
```

Image 1 - image has Fire and the Neural Network prediction (as shown at the bottom) “Fire”

```
[31]  1 from keras.preprocessing import image
      2 import matplotlib.pyplot as plt
      3 import matplotlib.image as mpimg
      4 test_image =image.load_img('/content/gdrive/My Drive/data/test/wallpapers_1.jpg', target_size =(100,100))
      5 imgplot = plt.imshow(test_image)
      6
```

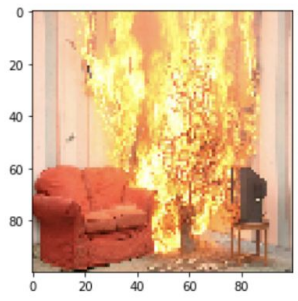



```
 1 test_image =image.img_to_array(test_image)
 2 test_image =np.expand_dims(test_image, axis =0)
 3 result = model.predict(test_image)
 4 if result[0][0] >= 0.5:
 5     prediction = 'fire'
 6 else:
 7     prediction = 'No fire'
 8 print(prediction)
```

```
No fire
```

Image 2 - Image has no fire and the Neural Network prediction is “No fire”

```
[33]  1 from keras.preprocessing import image
      2 import matplotlib.pyplot as plt
      3 import matplotlib.image as mpimg
      4 test_image =image.load_img('/content/gdrive/My Drive/data/test/tree-fire.jpg', target_size =(100,100))
      5 imgplot = plt.imshow(test_image)
      6
```




```
 1 test_image =image.img_to_array(test_image)
 2 test_image =np.expand_dims(test_image, axis =0)
 3 result = model.predict(test_image)
 4 if result[0][0] >= 0.5:
 5     prediction = 'fire'
 6 else:
 7     prediction = 'No fire'
 8 print(prediction)
```

```
fire
```

Image has fire and the Neural Network prediction is “Fire”

This convolutional neural network can be used in conjunction with satellite imagery, constantly monitoring forest for detection of wildfires. It can be used in conjunction with a separate neural network, which uses previous weather data during and before forest fires to estimate the size and intensity of the network.

Link to the Neural Network - https://colab.research.google.com/drive/1CW14peijYEgRodQOej82uwdicdYNxaLn

Sources

https://112.international/ukraine-top-news/forests-and-wildfires-top-five-largest-woods-fires-of-the-last-decade-42344.html

https://www.bustle.com/p/9-australia-fire-statistics-that-show-the-degree-of-its-impact-19777140

https://www.worldatlas.com/articles/largest-brush-and-forest-fires-in-recorded-history.htmlhttps://www.worldatlas.com/articles/largest-brush-and-forest-fires-in-recorded-history.html

https://www.bbc.com/news/world-latin-america-49450925

https://www.sandiegouniontribune.com/opinion/the-conversation/sd-california-2018-wildfires-burn-with-historic-impact-20181112-htmlstory.html

https://en.wikipedia.org/wiki/2019_Amazon_rainforest_wildfires

https://www.nytimes.com/2018/07/24/world/europe/greece-fire-deaths.html

https://www.aljazeera.com/programmes/faultlines/2019/11/amazon-burning-death-destruction-brazil-rainforest-191125232638261.html

https://www.nytimes.com/2018/07/24/world/europe/greece-wildfire.html

https://en.wikipedia.org/wiki/2010_Bolivia_forest_fires

https://www.bloomberg.com/graphics/2019-siberia-russia-wildfires/