AUTONOMOUS FLIGHT WILDFIRE DETECTION DRONE

Ilkhom Karimov

ilkhom.karimov@dronedetects.com

https://www.linkedin.com/in/ilkhomkarimov/

DroneDetects

ABOUT ME

Masters Degree in Computer Science in Seoul, South Korea

Enterprise System Integrator IBM Solutions

Master of Business Administration

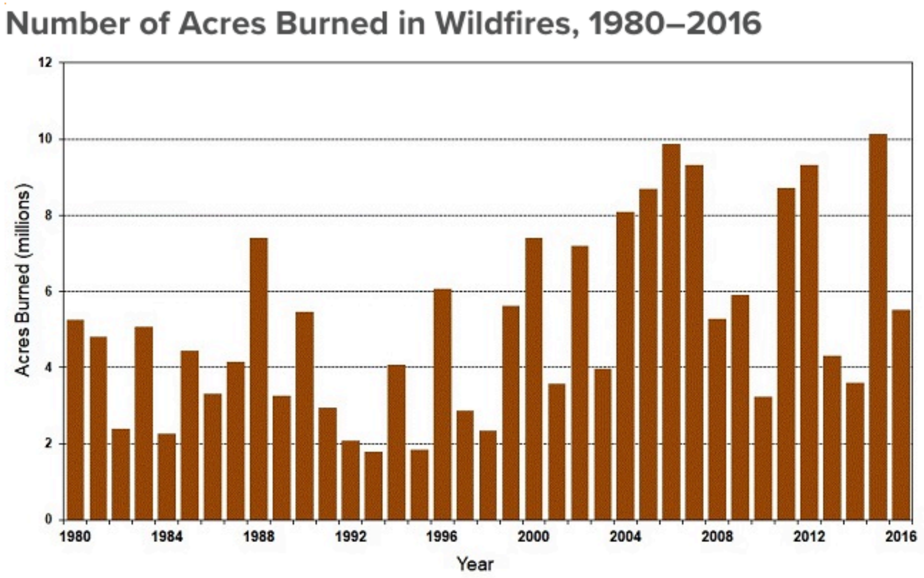
Founder Institute Startup Incubator Program Graduate

Software Engineer – Machine Learning

DroneDetects Startup Founder

INTRODUCTION

- Since 2000 6.9 million acres are burned in wildfires
- Double of average annual acreage burned than in the 1990s (on average 3.3 million burned in 1990)
- Losses of wildfires added up to 5.1 billion dollars over the past ten years



Source: National Interagency Fire Center.

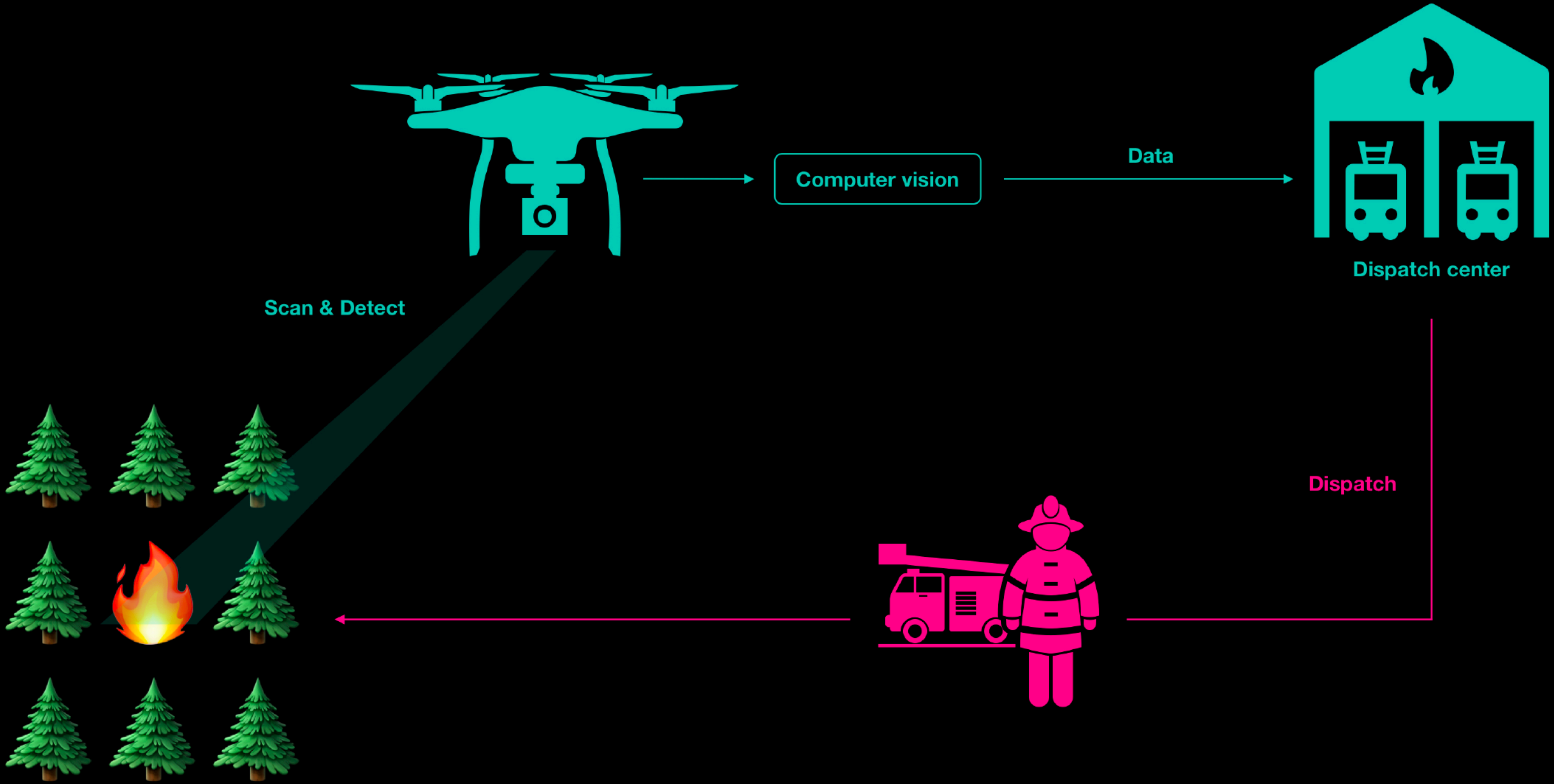
PROBLEM

- Very Difficult to put out fire at uncontrollable late stages
- Difficult to monitor large forest areas or mountain areas all the time
- People cause 90 percent of wildland fires in the United States
- Ground stations 2-5 Km coverage area
- Satellite Systems Alerts too late

Source - Insurance Information Institute

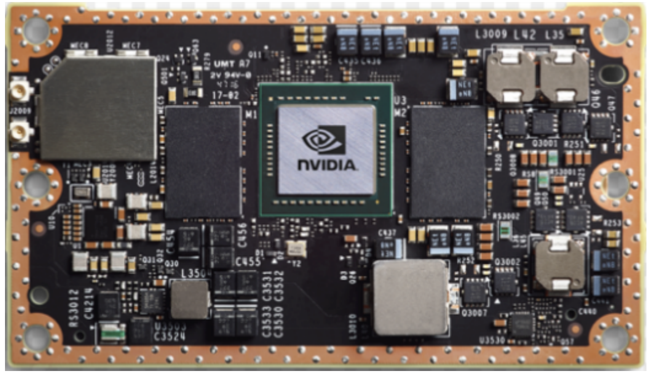
SOLUTION

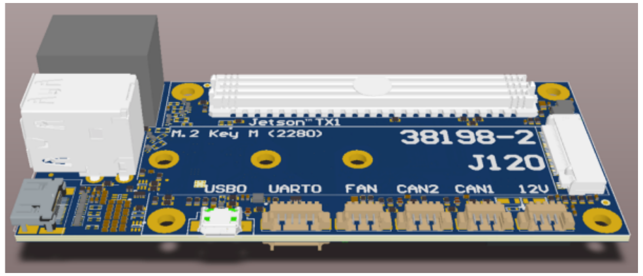
- Fight the fire at the earliest stages
- Autonomous Flight Drone with Computer Vision Fire
- Detection System
  - Can fly over 10 miles
  - Monitor designated area autonomously
  - Detect Fire Autonomously
  - Real-Time Monitoring System
  - Cost-effective


Computer vision
Data
Dispatch center
Scan & Detect
Dispatch

HARDWARE

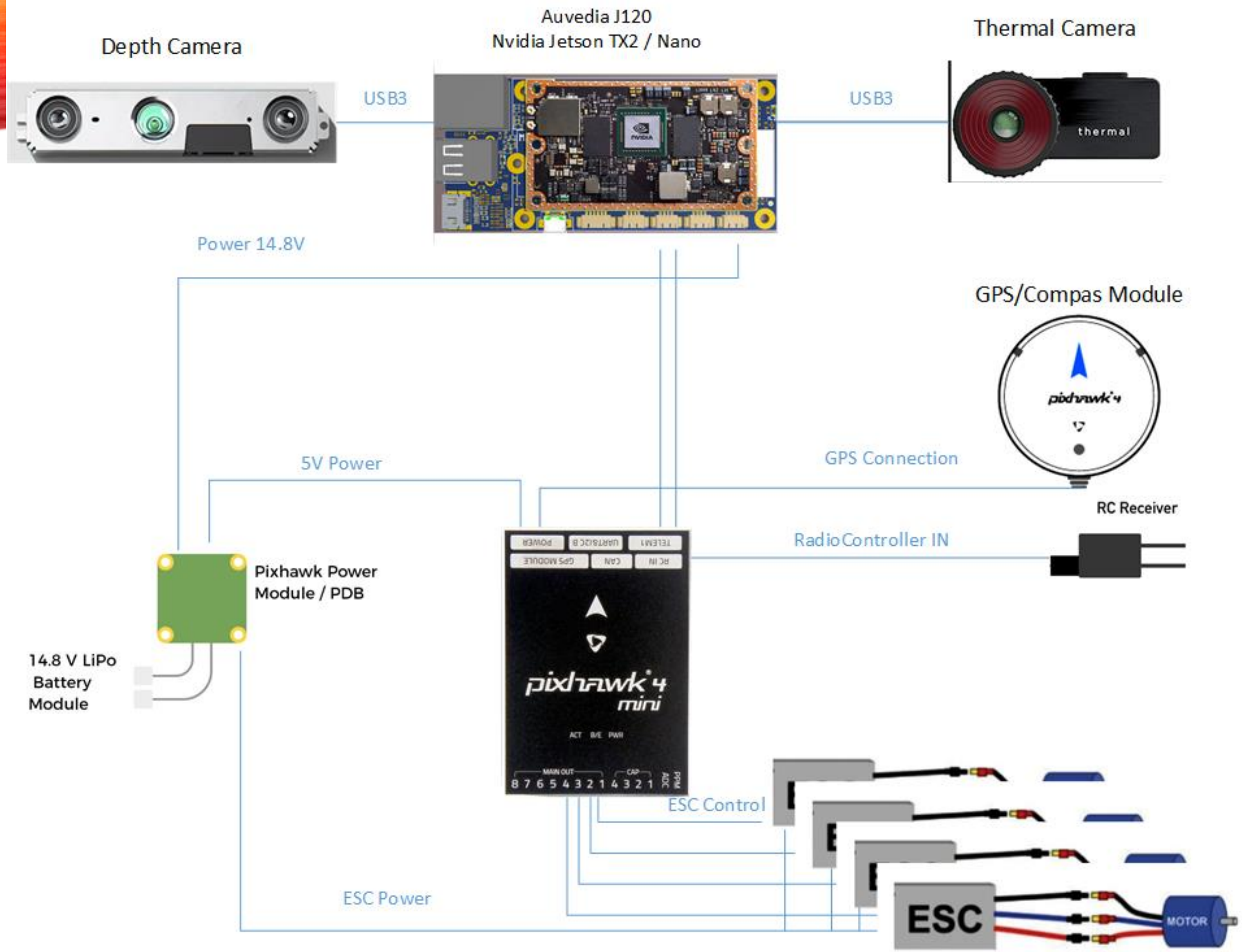
GPU	**NVIDIA Pascal™, 256 CUDA cores**
CPU	HMP Dual Denver 2/2 MB L2 + Quad ARM® A57/2 MB L2
Video	4K x 2K 60 Hz Encode (HEVC) 4K x 2K 60 Hz Decode (12-Bit Support)
Memory	8 GB 128 bit LPDDR4 59.7 GB/s
Display	2x DSI, 2x DP 1.2 / HDMI 2.0 / eDP 1.4
CSI	Up to 6 Cameras (2 Lane) CSI2 D-PHY 1.2 (2.5 Gbps/Lane)
PCIE	Gen 2 | 1x4 + 1x1 OR 2x1 + 1x2
Data Storage	32 GB eMMC, SDIO, SATA
Other	CAN, UART, SPI, I2C, I2S, GPIOs
USB	USB 3.0 + USB 2.0
Connectivity	1 Gigabit Ethernet, 802.11ac WLAN, Bluetooth
Mechanical	50 mm x 87 mm (400-Pin Compatible Board-to-Board Connector)



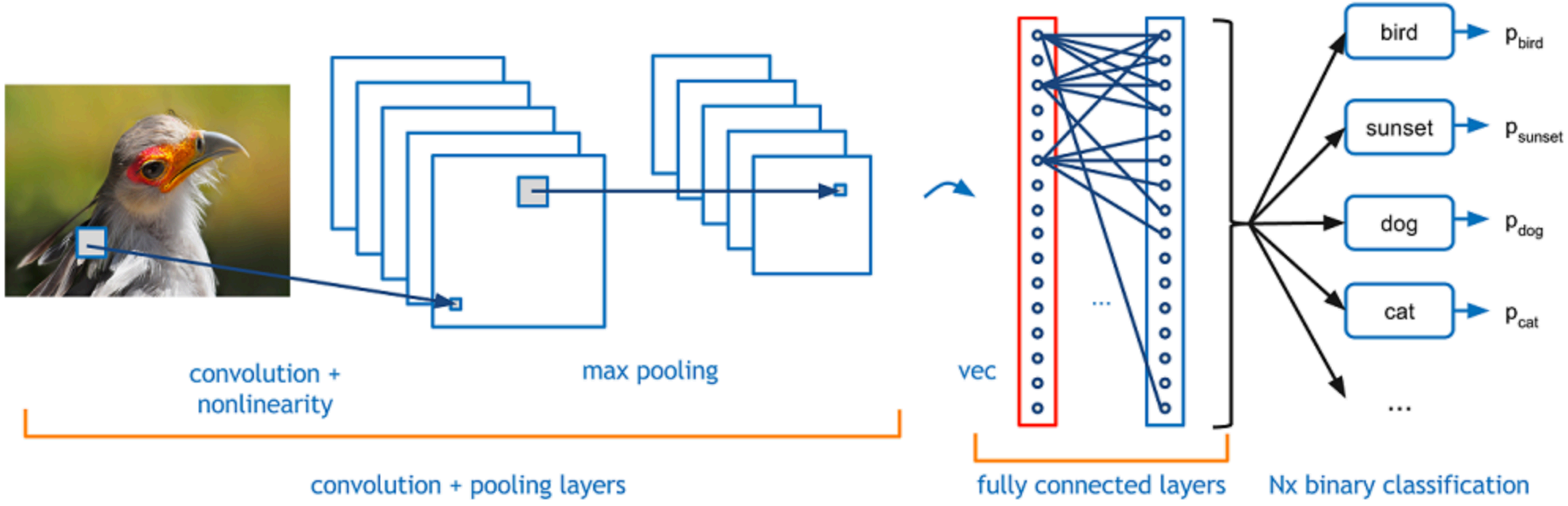


PROTOTYPE DRONE

HARDWARE DETAILS

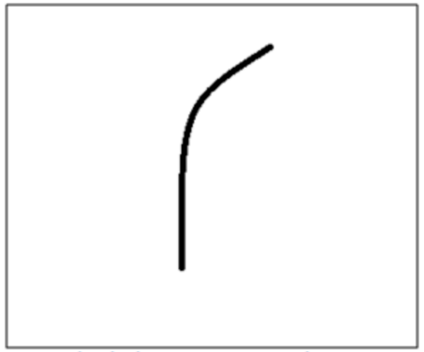


CONVOLUTIONAL NEURAL NETWORKS

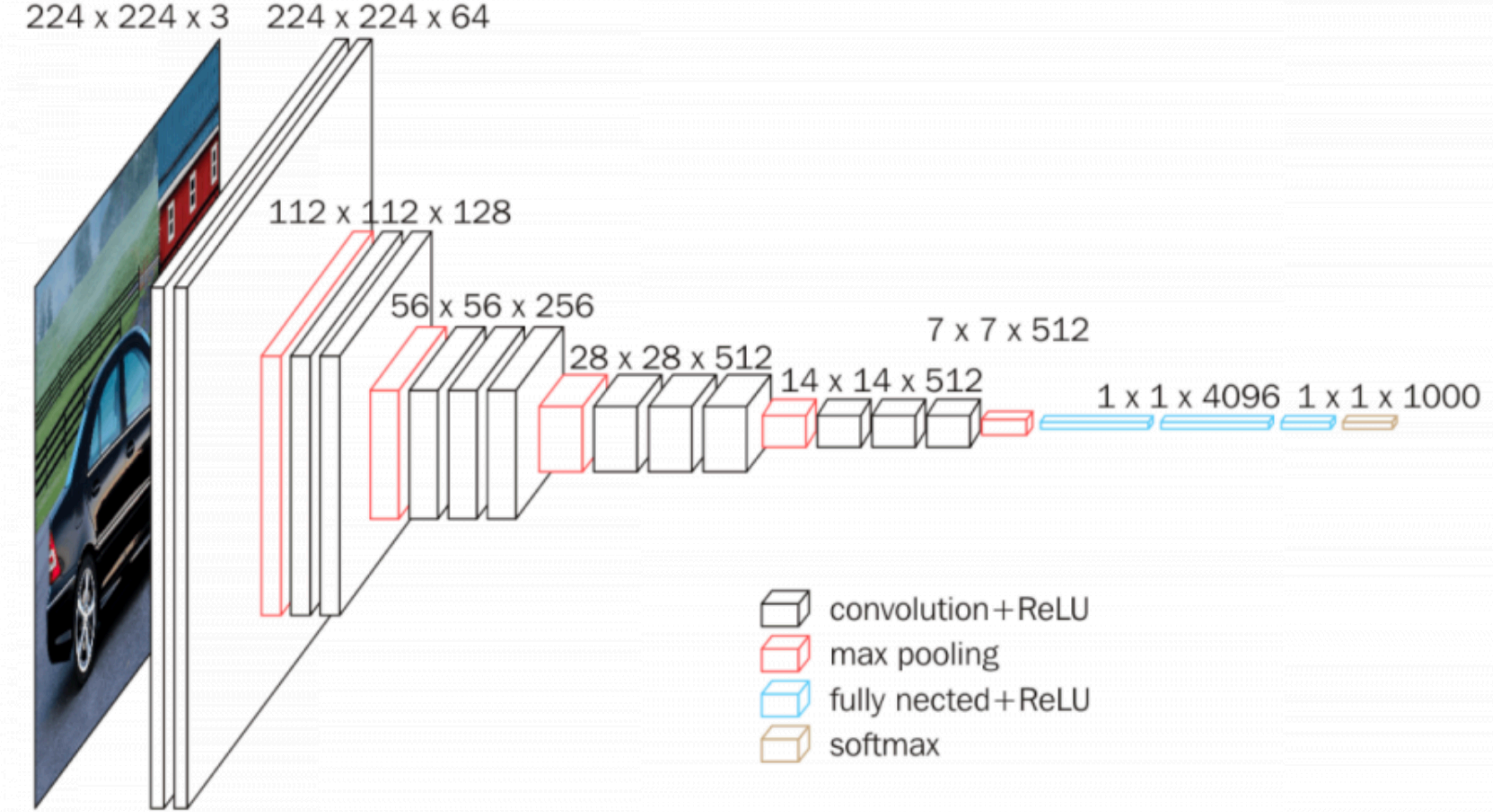


0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter



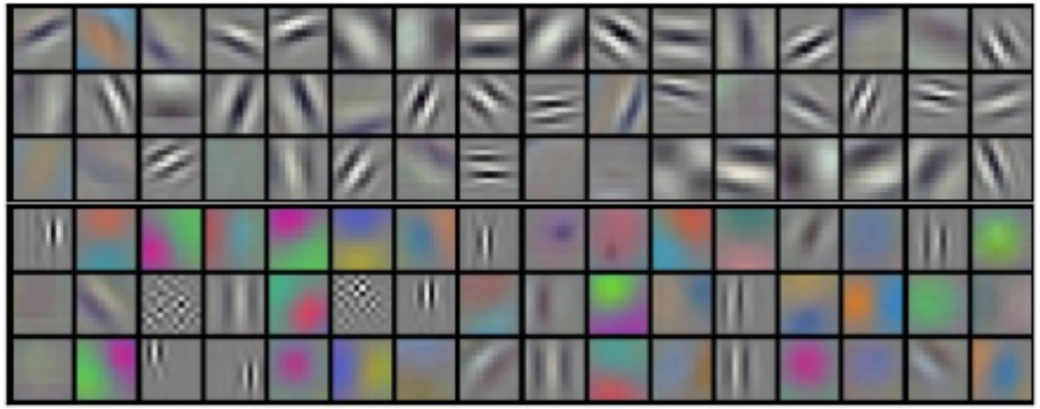
Visualization of a curve detector filter



224 x 224 x 3
224 x 224 x 64
112 x 112 x 128
56 x 56 x 256
28 x 28 x 512
14 x 14 x 512
7 x 7 x 512
1 x 1 x 4096
1 x 1 x 1000
convolution+ReLU
max pooling
fully nected+ReLU
softmax

TRANSFER LEARNING

- Transfer learning is a machine learning technique where a model trained on one task is re-purposed on a second related task
  - Self Driving Cars / Drones (Simulators)
  - Robots (Simulator)
  - Computer Vision (ex. ImageNet)



Example: Filters learned by AlexNet (Krizhevsky et al.,2012)

DATASET

1. Fatkun Batch Download Image – To Download all search results on Google Images
2. Manual Cleanup – around 2000 usable images
3. Download Drone flight footage videos (Manually from Vimeo/ Youtube)
4. Get images from video (Frames which are different)

```
import cv2
import imutils
import glob
```

```
video_files = glob.glob("./videos/*")
video_counter =0
```

```
for video in video_files:
    video_counter += 1
    capture = cv2.VideoCapture(video)

    first_frame = None  # initialize first frame
    change_count = 0
    # Threshold to know the frame changed or movement occured
    abs_difference_threshold = 1500000
    frames_allowed = 300
    previous_frame = None

    while True:
        (grabbed, frame) = capture.read()
        if not grabbed:  # if no frame break
            break
        original_frame = frame

        # Resize the frame
        frame = imutils.resize(frame, width=500)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        gray = cv2.GaussianBlur(gray, (21, 21), 0)

        if first_frame is None:
            first_frame = gray

        # Compute the absolute difference between this frame and first frame
        frame_delta = cv2.absdiff(first_frame, gray)

        # Count frame change or movement and update current frame
        if frame_delta.sum() > abs_difference_threshold:
            change_count += 1
            first_frame = gray
            print(frame_delta.sum())
            cv2.imwrite("./forest_images/frame-%d-%d.jpg" % (video_counter, change_count), original_frame)
```

Fire Detection Implementation Based VGG-16

In [1]:
```
# Import Helper Libraries
import numpy as np
from glob import glob
import cv2
import matplotlib.pyplot as plt
```

In [2]:
```
#Import VGG16 and DL Tools
from sklearn.datasets import load_files
from keras.utils import np_utils
from keras.applications.vgg16 import VGG16, preprocess_input
```

```
Using TensorFlow backend.
```

In [3]:
```
from keras.preprocessing import image
from tqdm import tqdm

def path_to_tensor(img_path):
    # load RGB image as PIL.Image.Image type
    img = image.load_img(img_path, target_size=(img_width, img_height))
    # Converting PIL.Image.Image type to 3D tensor with shape (224, 224, 3)
    x = image.img_to_array(img)
    # convert 3D tensor to 4D tensor with shape (1, 224, 224, 3) and return 4D tensor
    return np.expand_dims(x, axis=0)

# Pull dataset(Labeled Images) from folders
def paths_to_tensor(img_paths):
    list_of_tensors = [path_to_tensor(img_path) for img_path in tqdm(img_paths)]
    return np.vstack(list_of_tensors)
```

In [4]:
```
# Change image size to 224, 224
img_width, img_height = 224, 224

class_number = 2

# define function to load train, test, and validation datasets
def load_dataset(path):
    data = load_files(path)
    fire_files = np.array(data['filenames'])
    fire_targets = np_utils.to_categorical(np.array(data['target']), class_number)
    return fire_files, fire_targets

# load train, test, and validation datasets
train_files, train_targets = load_dataset('fireImages/train')
valid_files, valid_targets = load_dataset('fireImages/valid')
test_files, test_targets = load_dataset('fireImages/test')
```

In [38]:
```
# Freeze the layers which are not required to re train
for layer in model.layers:
    layer.trainable = False
```

In [39]:
```
# add a global spatial average pooling layer
x = model.output
x = GlobalAveragePooling2D()(x)
# x = Dropout(0.45)(x)
# and a logistic layer we have num_classes classes
predictions = Dense(num_classes, activation='softmax')(x)
```

In [40]:
```
from keras.models import Model
# this is the model we will train
model = Model(inputs=model.input, outputs=predictions)
```

In [41]:
```
# compile the model (should be done *after* setting layers to non-trainable)
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
#model.compile(loss='categorical_crossentropy', optimizer='adadelta', metrics=['accuracy'])
```

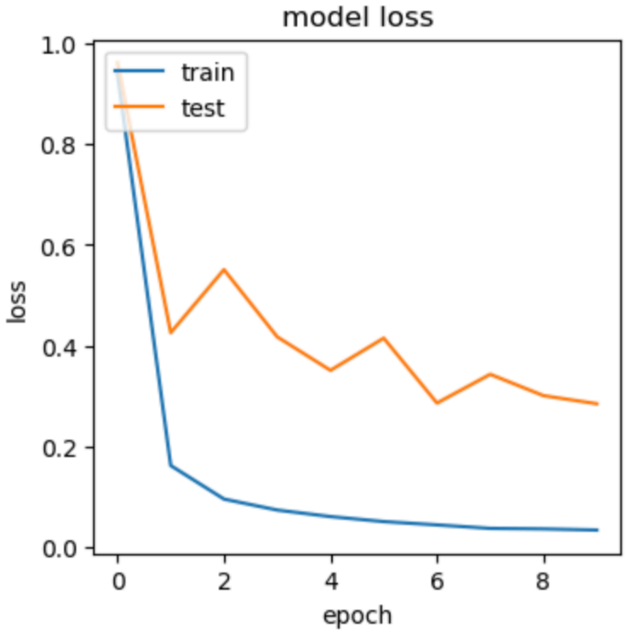
In [14]:
```
model.summary()
```

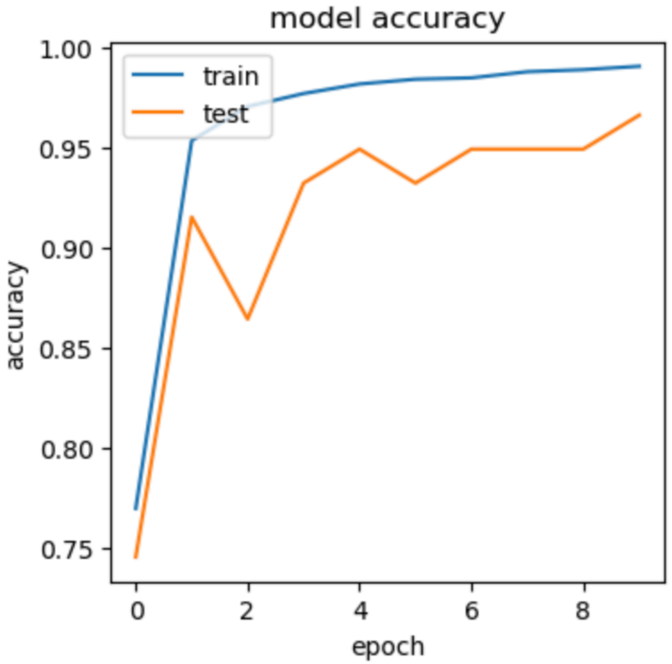
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         (None, 224, 224, 3)       0         
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0         
_________________________________________________________________
global_average_pooling2d_1 ( (None, 512)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 1026      
=================================================================
Total params: 14,715,714
Trainable params: 1,026
Non-trainable params: 14,714,688
_________________________________________________________________
```

In [16]:

```
from keras.callbacks import ModelCheckpoint
from keras.callbacks import TensorBoard

tbCallback = TensorBoard(log_dir='./Graph', histogram_freq=0, write_graph=True, write_images=True)

import keras.backend.tensorflow_backend as K

# train the model
checkpointer = ModelCheckpoint(filepath='firemodel.weights.best.hdf5', verbose=3, save_best_only=True)
hist = model.fit(train_tensors, train_targets, batch_size=64, epochs=10,
      validation_data=(valid_tensors, valid_targets), callbacks=[checkpointer, tbCallback],
      verbose=2)  #, shuffle=True)
```

RESULTS





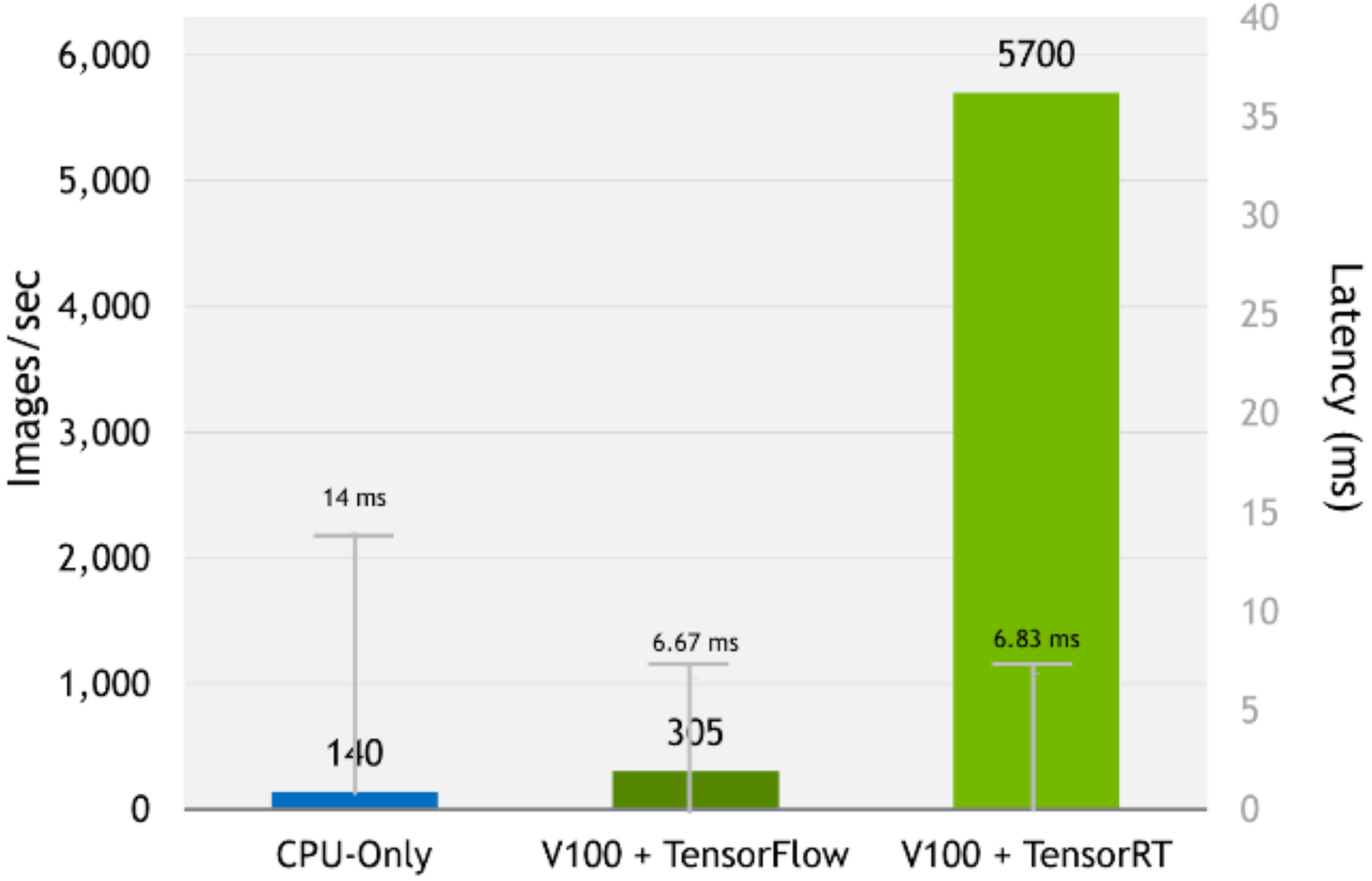
In [21]:
```
# report test accuracy
test_accuracy = 100*np.sum ( np.array( predictions)==np.argmax(test_targets, axis=1) ) / len(predictions )
print('Test accuracy: %.4f%%' % test_accuracy)
```

```
Test accuracy: 91.6667%
```

In [22]:
```
# test validation accuracy
predictions = [np.argmax(model.predict(np.expand_dims(feature, axis=0))) for feature in valid_tensors]
valid_accuracy = 100*np.sum ( np.array( predictions)==np.argmax(valid_targets, axis=1) ) / len(predictions )
print('Validation accuracy: %.4f%%' % valid_accuracy)
```

```
Validation accuracy: 96.6102%
```

Up to 40x Faster CNNs on V100 vs. CPU-Only Under 7ms Latency (ResNet50)



Inference throughput (images/sec) on ResNet50. **V100 + TensorRT**: NVIDIA TensorRT (FP16), batch size 39, Tesla V100-SXM2-16GB, E5-2690 v4@2.60GHz 3.5GHz Turbo (Broadwell) HT On **V100 + TensorFlow**: Preview of volta optimized TensorFlow (FP16), batch size 2, Tesla V100-PCIE-16GB, E5-2690 v4@2.60GHz 3.5GHz Turbo (Broadwell) HT On. **CPU-Only:** Intel Xeon-D 1587 Broadwell-E CPU and Intel DL SDK. Score doubled to comprehend Intel's stated claim of 2x performance improvement on Skylake with AVX512.

Freeze the graph

In [26]:
```
from keras.models import load_model
from tensorflow.python.framework import graph_io
from tensorflow.python.tools import freeze_graph
from tensorflow.core.protobuf import saver_pb2
from tensorflow.python.training import saver as saver_lib
```

In [27]:
```
def freeze_session(session, keep_var_names=None, output_names=None, clear_devices=True):
    """
    Freezes the state of a session into a pruned computation graph.

    Creates a new computation graph where variable nodes are replaced by
    constants taking their current value in the session. The new graph will be
    pruned so subgraphs that are not necessary to compute the requested
    outputs are removed.
    @param session The TensorFlow session to be frozen.
    @param keep_var_names A list of variable names that should not be frozen,
                          or None to freeze all the variables in the graph.
    @param output_names Names of the relevant graph outputs.
    @param clear_devices Remove the device directives from the graph for better portability.
    @return The frozen graph definition.
    """
    graph = session.graph
    with graph.as_default():
        freeze_var_names = list(set(v.op.name for v in tf.global_variables()).difference(keep_var_
names or []))
        output_names = output_names or []
        output_names += [v.op.name for v in tf.global_variables()]
        input_graph_def = graph.as_graph_def()
        if clear_devices:
            for node in input_graph_def.node:
                node.device = ""
        frozen_graph = tf.graph_util.convert_variables_to_constants(
            session, input_graph_def, output_names, freeze_var_names)
        return frozen_graph
```

In [28]:
```
from keras import backend as K
import tensorflow as tf
frozen_graph = freeze_session(K.get_session(),
                              output_names=[out.op.name for out in model.outputs])
```

```
WARNING:tensorflow:From <ipython-input-27-0d92ab84a915>:26: convert_variables_to_constants (from t
ensorflow.python.framework.graph_util_impl) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.compat.v1.graph_util.convert_variables_to_constants
WARNING:tensorflow:From /home/solid/anaconda3/envs/GPU_Keras/lib/python3.7/site-packages/tensorflo
w/python/framework/graph_util_impl.py:245: extract_sub_graph (from tensorflow.python.framework.gra
ph_util_impl) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.compat.v1.graph_util.extract_sub_graph
INFO:tensorflow:Froze 34 variables.
INFO:tensorflow:Converted 34 variables to const ops.
```

In [29]:
```
tf.train.write_graph(frozen_graph, "./", "frozen_fire_detector.pb", as_text=False)
```

Out[29]:
```
'./frozen_fire_detector.pb'
```

Convert to UFF

In [30]:
```
!convert-to-uff --input-file frozen_fire_detector.pb -l
```

```
/bin/sh: 1: convert-to-uff: not found
```

Convert to uff

In [31]:
```
!convert-to-uff -o fire_detector.uff --input-file frozen_fire_detector.pb
```

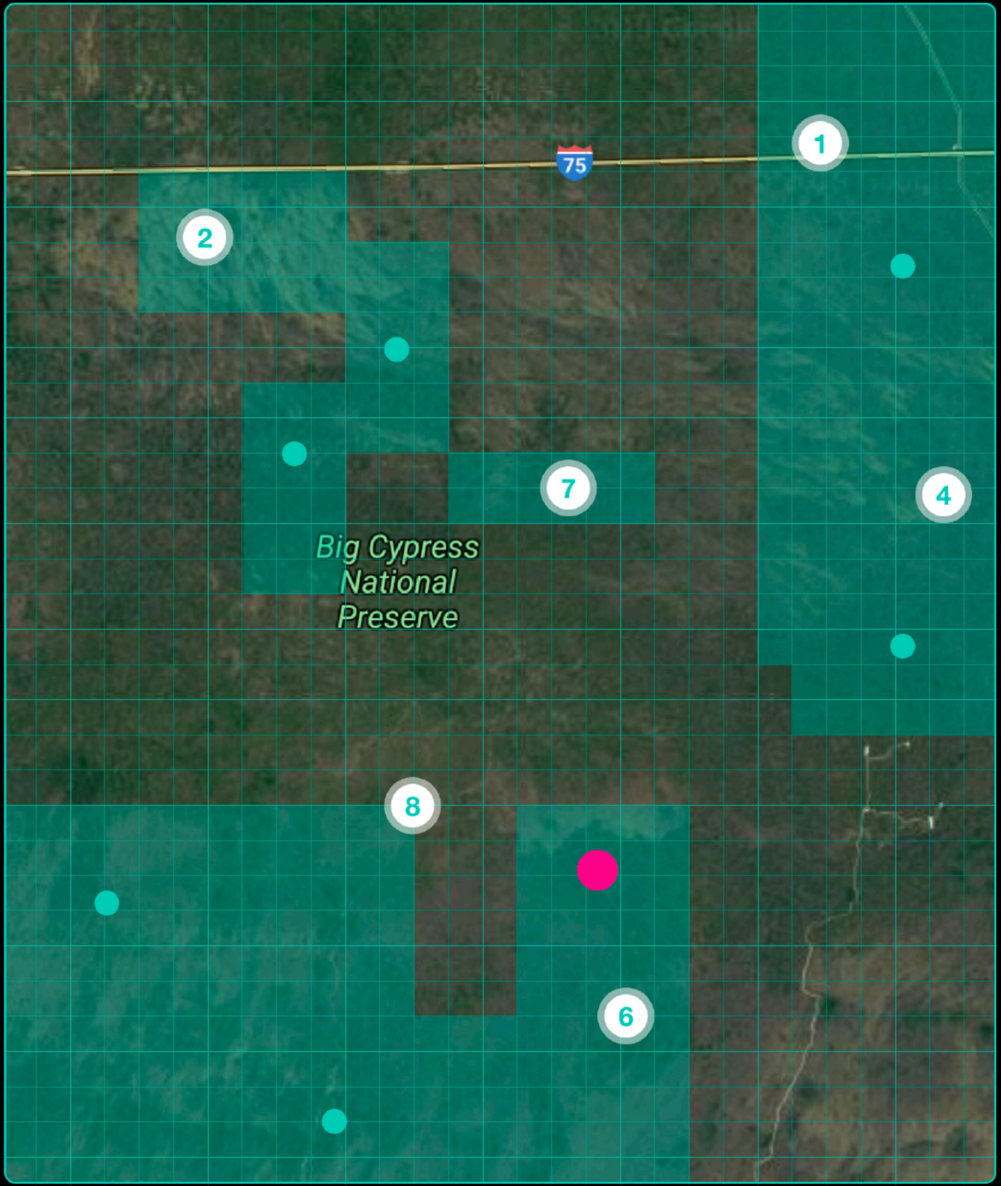
AUTONOMOUS FLIGHT

- Dronekit
- DroneKit-sitl
- Mavproxy
- Ardupilot (QGroundControl)

```python
from dronekit import connect, VehicleMode, LocationGlobalRelative

print('Connecting to vehicle on: udp:127.0.0.1:14551')
vehicle = connect('udp:127.0.0.1:14551', wait_ready=True)

def arm_and_takeoff(aTargetAltitude):

    print("Basic pre-arm checks")
    # Don't try to arm until autopilot is ready
    while not vehicle.is_armable:
        print(" Waiting for vehicle to initialise...")
        time.sleep(1)

    # Copter should arm in GUIDED mode
    vehicle.mode = VehicleMode("GUIDED")
    vehicle.armed = True

    # Confirm vehicle armed before attempting to take off
    while not vehicle.armed:
        print(" Waiting for arming...")
        time.sleep(1)

    print("Taking off!")
    vehicle.simple_takeoff(aTargetAltitude)  # Take off to target altitude

    while True:
        print(" Altitude: ", vehicle.location.global_relative_frame.alt)
        # Break and return from function just below target altitude.
        if vehicle.location.global_relative_frame.alt >= aTargetAltitude * 0.95:
            print("Reached target altitude")
            break
        time.sleep(1)

arm_and_takeoff(10)

vehicle.airspeed = 3

# Going towards first point for 30 seconds
point1 = LocationGlobalRelative(-35.361354, 149.165218, 20)
vehicle.simple_goto(point1)

# sleep so we can see the change in map
time.sleep(30)

print("Going towards second point for 30 seconds (groundspeed set to 10 m/s) ...")
point2 = LocationGlobalRelative(-35.363244, 149.168801, 20)
vehicle.simple_goto(point2, groundspeed=10)

# sleep so we can see the change in map
time.sleep(30)

print("Returning to Launch")
vehicle.mode = VehicleMode("RTL")

# Close vehicle object before exiting script
print("Close vehicle object")
vehicle.close()
```

DASHBOARD DRONE HISTORY SETTINGS

REAL-TIME SCANNING AREA



SCAN TASK

Big Cypress National Preserve - 01

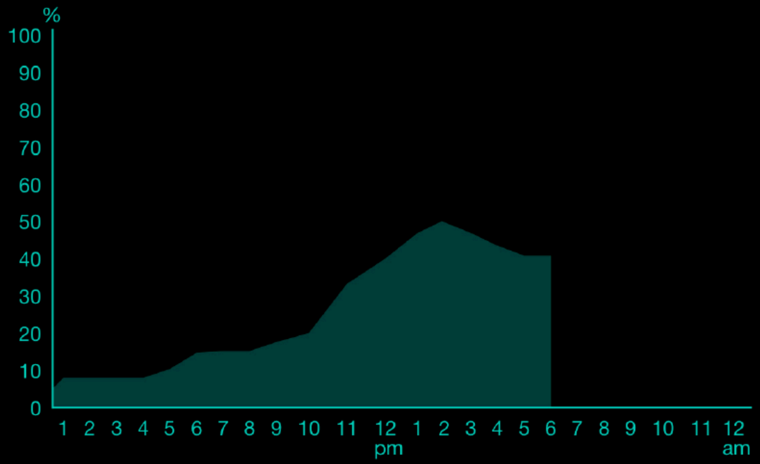
AREA SCANNED
57%

SQMILE
635

FIRE DETECTED
12

DISPATCH
4

FIRE PREDICTION

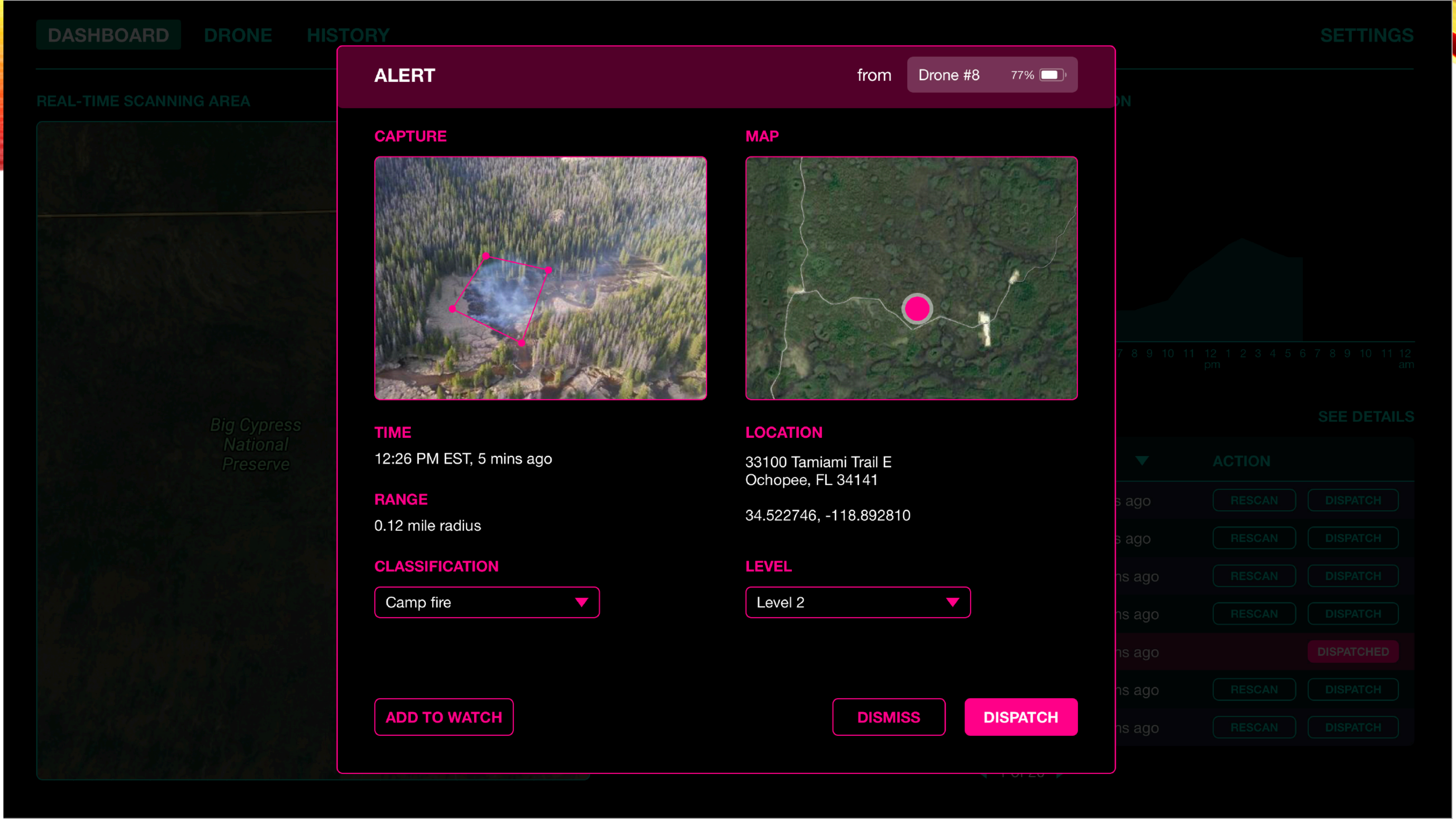


WATCH LIST

SEE DETAILS

CLASSIFICATION	RANGE ▼	LEVEL ▼	TIME ▼	ACTION
Camp fire	0.01 mi	2	3 mins ago	RESCAN DISPATCH
Smoke	0.03 mi	1	5 mins ago	RESCAN DISPATCH
Smoke	0.08 mi	1	10 mins ago	RESCAN DISPATCH
Camp fire	0.01 mi	2	15 mins ago	RESCAN DISPATCH
Fire	0.15 mi	5	16 mins ago	DISPATCHED
Smoke	0.01 mi	1	22 mins ago	RESCAN DISPATCH
Camp fire	0.01 mi	2	45 mins ago	RESCAN DISPATCH

◀ 1 of 20 ▶


DASHBOARD
DRONE
HISTORY
SETTINGS
REAL-TIME SCANNING AREA
Big Cypress National Preserve
ALERT
from
Drone #8
77%
CAPTURE
MAP
TIME
12:26 PM EST, 5 mins ago
RANGE
0.12 mile radius
CLASSIFICATION
Camp fire
LOCATION
33100 Tamiami Trail E
Ochopee, FL 34141
34.522746, -118.892810
LEVEL
Level 2
ADD TO WATCH
DISMISS
DISPATCH
SEE DETAILS
ACTION
RESCAN
DISPATCH
DISPATCHED

SUMMARY

This is ongoing project

Still working on the prototype fine tuning

Model has high accuracy but need to be live tested