THE CANADIAN SCIENCE FAIR JOURNAL

ARTICLE

Deep Learning Based Fire Recognition for Wildfire Drone Automation

Robin Yadav

Age: 16 | Surrey, British Columbia

BC Game Developers Innovation Award | Engineering and Geoscientists of BC Award, Placed in Top 5% of projects (BC Online STEM Fair) | Intact Financial Climate Change Resilience Award (Youth Science Canada Online STEM fair) | BC Youth Innovation Showcase Finalist

Wildfires are one of the most devastating and harmful natural disasters that occur in Canada every year. Firefighters require the best tools and equipment to fight these wildfires and limit their damage. Drones are becoming an increasingly useful asset to firefighters for wildfire monitoring and assessment. This project leverages deep learning to create a novel video-based fire detection system to add fire recognition and automation capability to drones. Several state-of-the-art deep learning models were trained and compared. Approximately three thousand images of fire were scraped from the web or found from smaller datasets and then labeled. Various amounts of data were augmented, and different types of augmentations were used to increase the size of the dataset. Additional experimentation was done with the training batch size, confidence threshold and intersection over union (IOU) threshold to obtain the greatest mean average precision (mAP) for fire detection. Unity Real-Time Development Platform was used to simulate a fire front and to automate drone movement. An automation algorithm was designed to assess recognized fire in video and output future movement. Specifically, the drone was tasked to fly parallel to a stabilized fire front by considering the distribution of fire across an image. In addition to this algorithm, a DJI Tello drone was automated using Arduino technology to fly between GPS coordinates in the real world and send alerts if fire is detected. The greatest mAP value (@IOU 0.5) was obtained by YOLOv3 at 89.5% and 98.7%. In the simulation, the drone was able to maintain a relatively close proximity to the stabilized fire line and GPS based navigation acts as a failsafe. Data preparation significantly impacts the performance of the model suggesting that certain labeling and augmentation techniques make patterns and features of fire more distinct and recognizable. In the simulation, the drone is able to handle general movement but cannot perform more intricate movements such as making tight turns in succession. Due to the high AP and fast inference speed of the model, this system is viable for real time fire detection. Testing demonstrates that automated drones may have the potential for increasing the efficiency of wildfire monitoring and providing firefighters with critical information.

INTRODUCTION

Every year, approximately eight thousand wildfires burn across Canada and damage over 2.5 million hectares of land (Statistics Canada, 2019). In British Columbia alone, an average of 300,000 hectares are burnt by wildfires causing 260 million dollars in destruction every year (BC Wildfire Service, 2020). Recently, drones have become useful tools for firefighters. (Shen, 2018) Wildfire drones survey the fire and provide critical monitoring information that firefighters need to reduce the damage a wildfire can cause. Tasks such as fire line monitoring, hot spot location, wildfire and destruction mapping and real time aerial video feed assessment can be conducted through wildfire drones (Savvides, 2018). However, human-operated drones are not the most efficient allocation of time and resources. Furthermore, they are difficult to operate in high stress situations and can put firefighters at risk. (Savvides, 2018)

In this paper, I investigate a potential system for wildfire drone automation using deep learning object detection models for fire recognition. Autonomous drones would be able to fly along a fire line, map the area and send data to firefighters. Automated drones have the possibility of improving wildfire monitoring efficiency and effectively providing firefighters with critical information on the fire.

Previous work in wildfire drone automation proposed the idea of GPS navigation of drone swarms guided by high altitude UAVs. (Afghah et al., 2019) However, high altitude UAVs are very expensive and drone swarms are very difficult to coordinate (Zhu et al., 2015). Some developments of autonomous drones for disaster relief reconstruct indoor settings in 3D and using those reconstructions for autonomous flight and performing intricate movements (Apvrille et al., 2014). My approach focuses on the automation of a single drone independent of other drones in the area. The movement of the drone is determined by the location of the surrounding fire and is not limited by obstacles and barriers which are present in an indoor setting.

It is critical to have an accurate fire detection system as a base for drone automation and wildfire monitoring. Most previous video-based fire detection systems use hand-crafted features such as spatial, temporal and color characteristics of fire to perform recognition (Phillips et al., 2002, Toulouse et al., 2015, Toulouse

et al., 2017). Although in recent years, there has been an interest in leveraging convolutional neural networks (CNN) for fire image classification.

A CNN adjusted from GoogleNet was used to classify images as fire or non-fire (Muhammed et al., 2018). Other developments include a CNN which performs patch identification on images of fire (Zhang et al., 2016). However, those CNNs lack localization functionality, so they cannot identify where the fire is within an image. I implemented fire detection using state-of-the-art object detection models which can identify the position of the fire within an image without the use of hand-crafted features. Several deep learning models were trained and compared to determine the most suitable model for fire detection. This approach allows for robust and accurate fire detection while still maintaining the ability to run in real time on low cost devices (Vidyavani et al., 2019). Also, the models are very versatile because the training data includes fires in many different contexts, from small localized fires to larger forest and bush fires. The use of deep learning object detection models for fire recognition and automating drones can offer an efficient system for wildfire monitoring.

METHODS

Digital images of fire were scraped from the web or collected from smaller pre-existing datasets. The FireSmoke, FireDetectionImage and FlickrFireSmoke (DeepQuestAI, 2019, Cair ,2017, Cazzolato T. Mirela et al., 2017) datasets were used. Images with low resolution (below 200 by 200 pixels), low fire visibility, and inaccurate representations of fire were discarded (supplementary figure 1). This includes images in which the fire was completely obstructed by smoke or was too small and blurred to be recognized distinctly as fire.

Negative examples that contained red objects (e.g. fire hydrants) were web scraped and added to the dataset. The dataset contained a total of 3057 images comprised of 2732 images of fire (totaling 8000 instances) and 325 images of no fire. A test set of 100 images of fire in high risk emergency situations and another test set of 50 single flame fire images (e.g. campfires) were compiled (supplementary figure 2).

The images were annotated using Microsoft VOTT (Visual Object Tagging Tool) to specify bounding boxes around the fire objects. Fire was annotated using 2 different labeling strategies (supplementary figure 3). In the first approach, individual flames of a whole fire were annotated separately. While in the second approach, fire was unsegmented and annotated as a whole regardless of the fact that it was composed of individual flames.

Labeling tools such as Microsoft VOTT can often produce corrupted data upon export. Due to this, the training and validation loss converged to NaN, which is an undefined datatype resulting from an error in a numerical calculation (e.g. divide by zero). A python script was created to locate and discard corrupted data.

Offline data augmentation was used to increase the size of the dataset. Image cropping, translating, rotating, reflecting, and hue, saturation and value (HSV) transformations were applied. Different variations of the dataset were created based on the amount and type of augmented images.

90% of the examples formed the training set and the remaining 10% formed the validation set. YOLOv3 (You Only Look Once), YOLOv3-Tiny, YOLOv3-SPP (Spatial Pyramid Pooling), YOLOv4, CSResNext50-Panet-SPP, and SSD-ResNet (Single Shot Detector with ResNet feature extractor) were trained on the dataset (Redmond and Farhadi, 2017, Bochkovskiy et al., 2020, Liu et al., 2018). All of the models were fine-tuned from pre-trained MSCOCO dataset weights. Since the training time for YOLOv3-Tiny was relatively shorter than the other models, it was used to perform experiments and refine the data. All of the YOLO models and CS-ResNext were trained with DarkNet while SSD was trained using TensorFlow. The images were resized to 416 by 416 and the batch size was varied between 16 and 128. An Intel i5-6200U CPU was used to perform inference and test the models using the mAP (mean Average Precision) metric. Due to the memory limitations of the GPUs used for training, some of the training trials were unable to be executed at a batch size 128.

In order to perform automation, the drone was tasked to fly parallel to the fire front or fire line. This task was performed by evaluating the distribution of fire within a frame of the drone's video feed. The frame was split into an n x n grid. If the center of a grid cell is contained within a bounding box, it is considered a "fire grid cell". The drone was parallel to the fire line when the column-based distribution of fire grid cells was concentrated on the extreme right or extreme left of the video frame (depending on the direction the drone is facing). If this scenario was not achieved, the drone rotated and changed its position slightly until it was orientated parallel to the fire line.

A threshold was applied at the ith column such that a grid cell in the jth column was considered on the "extreme" right of a video frame when $j \leq i$. The same reasoning was applied but in the opposite direction when the fire line was to the left of the drone (supplementary figure 4). Parameters such as the threshold, frames per detection and the rotation speed of the drone were varied to find the optimal configuration.

Unity Real Time Development Platform was used to create simulations for experimentation and testing of automation parameters. The object detection models were exported to Unity through serialized protocol buffers and used for inference via Net based TensorFlow and OpenCV. A fire line was generated which tested the drone's ability to move in all directions, turn right, left etc. The goal of the drone was to detect the fire and fly along the fire line from one end to the other. Z-axis (vertical movement) was ignored and the drone was set at constant height above the trees to avoid collision. The grid size was set to 20 by 20. Drone movement was evaluated by measuring the horizontal distance between the drone and fire line, the velocity of the drone, the amount of fire it is capturing with its camera and the total distance it trav-

elled. Ten iterations of the simulation were executed for each configuration of the parameters. The values of the parameters were set to reasonable values expected of drones in reality. Any configuration which resulted in the drone colliding with the fire line (horizontal distance to fire line was 0 m) or not being able to complete the path was discarded.

A secondary automation algorithm using GPS was implemented in real life. A GPS module, compass module, radio module and an Arduino Nano were configured to a DJI Tello drone. The positional information and video stream of the drone were sent to a Raspberry Pi which performed inference, displayed the results via a flask application and allowed the drone to fly between two GPS points (supplementary figure 5).

RESULTS

YOLOv3 Tiny was used to experiment with the data. The AP values were measured at a confidence threshold of 10% and the predictions were made on the 100-image test set. Figure 1 shows the difference in AP between segmented and unsegmented fire labeling with no data augmentation. The average AP was calculated from IOU thresholds varied between 10% and 50% with a step size of 10%. Unsegmented labeling performed better in every case.

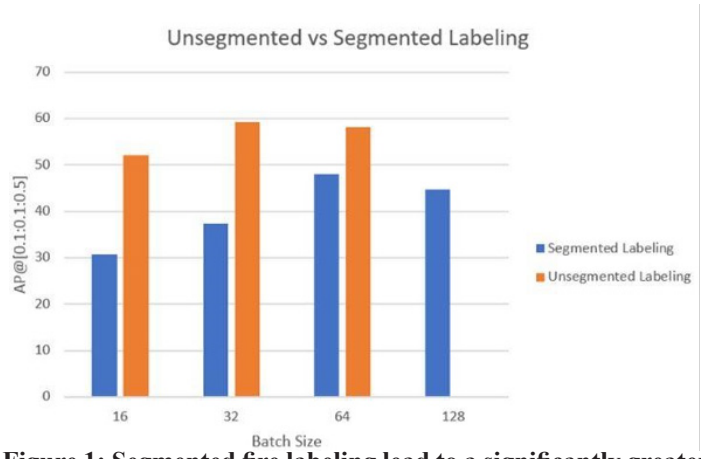


Figure 1: Segmented fire labeling lead to a significantly greater model performance at every measured batch size. The average difference in mAP between segmented and unsegmented fire labeling was 17.7%.

Further testing was done to see if data augmentation could improve the AP obtained from segmented fire labeling. Figure 2 compares the performance of YOLOv3 Tiny on different amounts of augmented data with HSV transformations. Also, performance is compared without HSV transformations (supplementary figure 6). Disabling HSV transformation increased model performance by 1.1% with segmented fire labeling. A more significant effect is observed with unsegmented fire labeling. Figure 3 highlights the difference in AP on unsegmented fire labeling that used data augmentation but without HSV transformation. The highest AP achieved with unsegmented fire labeling without HSV transformation was 64.3% which is an 8.4% increase from

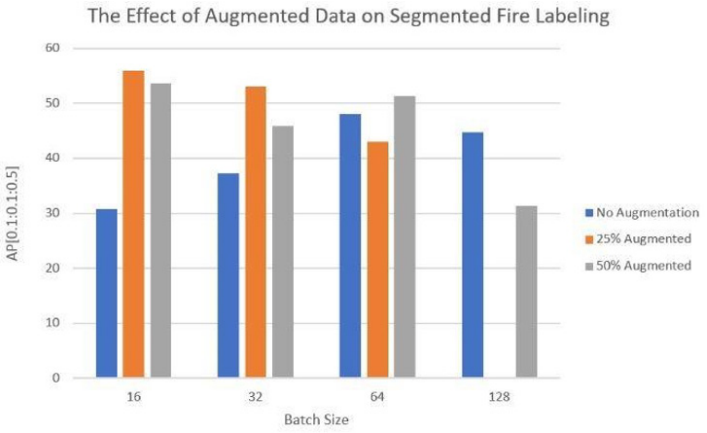


Figure 2: The highest performance on YOLOv3 Tiny with segmented labeling was obtained when 25% of the raw data was augmented with a batch size of 16. 25% augmentation of the raw data outperforms 50% augmentation by a slight margin in 2 out of the three tests. Having augmented data leads to higher mAP in almost every case. Note that this is when HSV augmentation is applied.

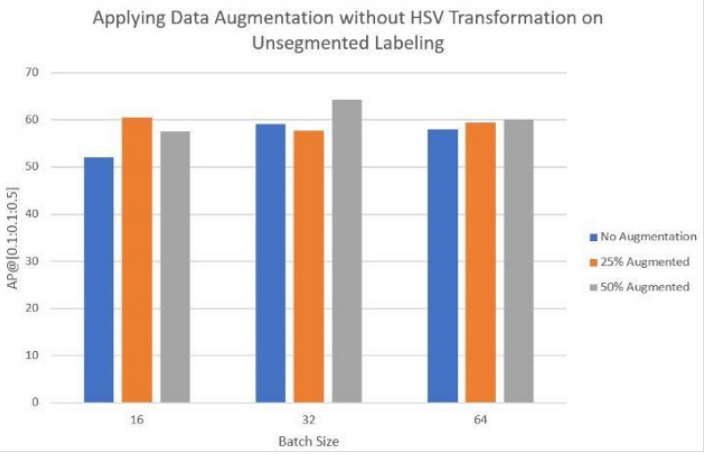


Figure 3: Data augmentation increases the performance of the YOLOv3 Tiny model with unsegmented fire labeling. Specifically, the highest model performance is obtained when 50% of the raw data is augmented.

segmented fire labeling. Interestingly, when the entire dataset was augmented, an AP of 64.6% AP@[0.1:0.1:0.5] was obtained which was on par with 50% of the dataset consisting of augmented images.

Around 400 additional images of fire were collected and labeled in the unsegmented format. The performance effect of the additional images is compared to the original amount of raw training data (supplementary figure 7).

YOLOv3 obtains the highest AP with unsegmented fire labeling and augmentation of the entire dataset without HSV transformations. The remaining models were trained with that dataset configuration.

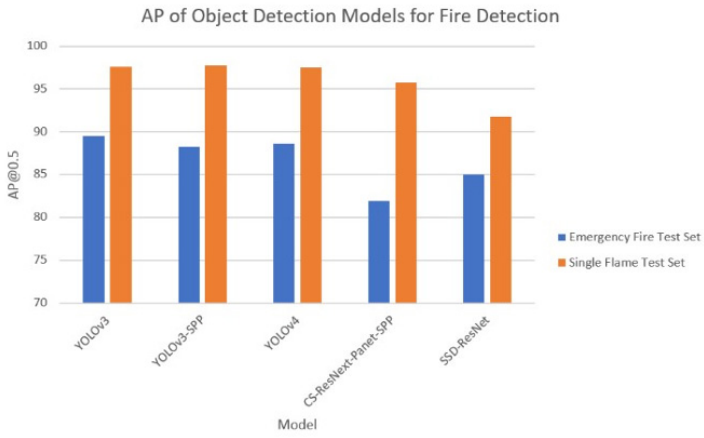


Figure 4: Performance on the emergency fire test set was consistently lower across all models than the single flame test set. YOLOv3 obtained the highest mAP on the emergency fire test set and YOLOv3-SPP obtained the highest mAP on the single flame test set.

Figure 4 shows the AP@0.5 of the all of the models on the emergency fire test set and the single flame test set.

All of the YOLO models perform similarly. YOLOv3 obtained the highest mAP on the emergency fire at 89.5% test while YOLOv3-SPP obtained a slightly lower AP of 88.3%. Also, YOLOv3-SPP performed slightly better on the single flame test with an AP of 97.81% which was 0.21% higher than YOLOv3. YOLOv3-Tiny had the lowest mAP out of all of the models. However, inference speed must also be considered. Figure 5 shows the average inference speed of the model per image tested on 100 images.

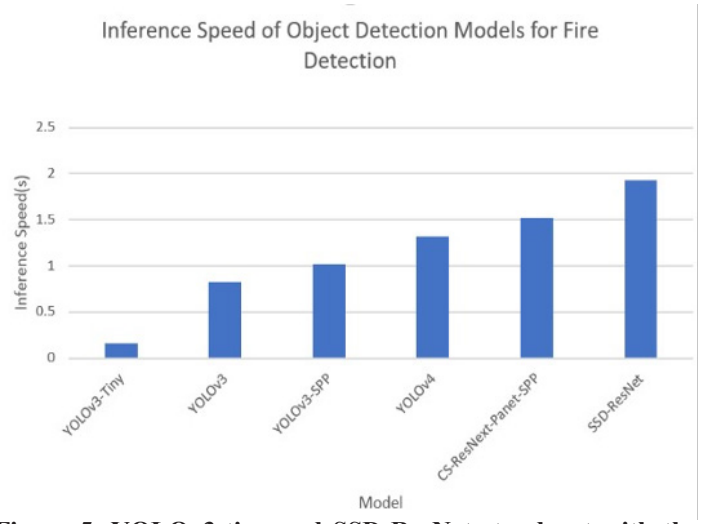


Figure 5: YOLOv3-tiny and SSD-ResNet stand out with the lowest and highest inference speed respectively. All of the other models have similar inference speeds.

Although YOLOv3 Tiny had the lowest mAP of all models, its inference speed was around 10 to 15 times less.

The optimal configuration for drone movement parameters consisted of detecting every 10 frames, a rotation speed of 0.45 degrees per frame and a threshold value at the 5th column. The average horizontal distance the drone maintains from the fire line was approximately 10.6 meters. The average velocity of the drone was 1.5 meters per second. On average, the number of grid cells it detected per frame was 8. The total distance of the fire line was 431.50 meters while the drone covered on average a distance of 368.2 meters per run. Figure 6 is a graph of how much the drone deviated from the fire line.

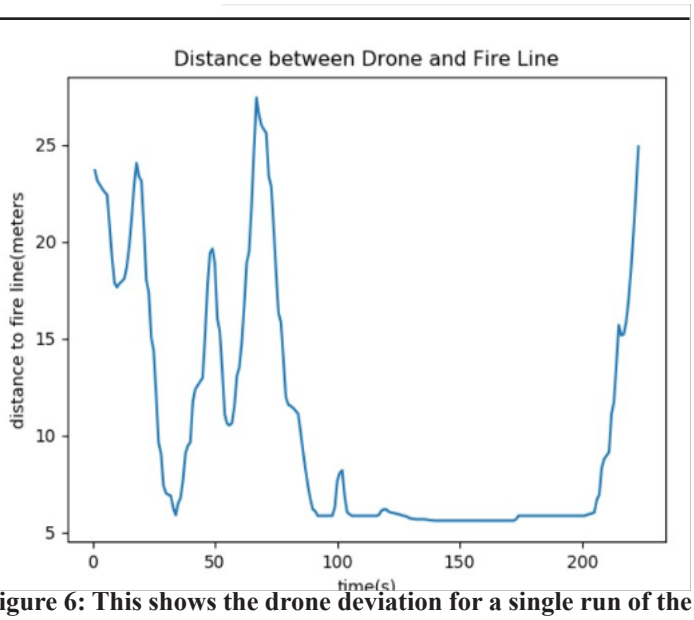


Figure 6: This shows the drone deviation for a single run of the simulation. While the drone movement was relatively stable for some periods of time, there are large increases indicating that the drone deviated before returning to its original position.

DISCUSSION

It was determined that the unsegmented labeling approach with augmented data (no HSV transformation) resulted in the greatest AP value. YOLOv3 had the highest performance with an AP of 89.5% on the test set of emergency fire images and 97.6% on the single flame images.

Perhaps annotating fire with bounding boxes was not the most suitable method for representing the features of fire and degraded model performance. Certain objects are well defined by a rectangular box e.g. cars, pedestrians, signs etc. but sometimes fire is very irregular. Polygon or semantic annotation could be used to eliminate ambiguities and difficult edge cases for fire labeling and they also provide a richer representation of objects (Endres et al., 2010). For example, some object detection applications for medical imaging show that performance is better for brain detection than lung detection because the shape of the brain can be better approximated by a bounding box. (Rajchl et al., 2016)

Many ambiguities arise when labeling fire with bounding boxes. For example, two flames which are connected at the base can be labeled as two independent flames or as one large flame. Segmenting fire creates additional patterns the object detector must recognize. There are more shapes, sizes and illumination and angles to consider. Notably, contextual information about the location and shape of the whole fire is required for segmentation. In contrast, labeling a region of fire in its entirety without flame segmentation can decrease the complexity of the task and allow the model to recognize consistent patterns in the data.

Usually, an increase in augmented data prevents the model from overfitting during training which leads to better performance (Shorten and Taghi, 2019). However, an increase in augmented data in the segmented fire dataset did not consistently improve the performance of YOLOv3 Tiny. This inconsistency might emerge from the fact that segmented fire detection is already a complex task. Increasing the amount of augmented data can result in unnecessary noise and irregularity making object detection more difficult. Furthermore, model performance increased when HSV transformations were removed. Since most of the features of fire are seen in the red spectrum, HSV transformations create unrealistic data by shifting the color of fire, which creates variance and leads to inconsistent results.

Interestingly, there is not a significant difference in AP even when data augmentation is applied to the unsegmented fire dataset. Possibly because there is not enough augmented data to create large changes in performance. The greatest ratio of augmented data to raw data was 1:1. Perhaps a larger ratio such as 2:1 or 3:1 is needed to observe significant effects of data augmentation. In addition, there was only one test for every batch size and ratio. Testing each batch size and ratio 3 to 4 times and averaging the results can be a better indication of the performance on the dataset.

There is a notable difference in AP between the emergency fire images test set and the single flame test set. Since a single flame has a much simpler structure and shape than a multi-flame fire, its patterns are more recognizable and distinct making it easier to detect. Also, fire in emergency situations is usually clouded by smoke and is less distinct against the background making it more difficult to detect. The ability to perform well on both test sets is representative of the difficulty and diversity of the training data.

The 3000 images in the dataset include fire in many different contexts and visibility settings. Large fires such as forest, bush, and house fires are included in the dataset with smaller, more localized fires. Therefore, the model is able to generalize and perform well on difficult and diverse test sets. Other object detection approaches either lack a large dataset (500-1000 images) or their dataset is too homogenous (Sucuoğlu et al., 2019, Barmpoutis et al., 2019). A 97% AP obtained by YOLOv3 on the single flame test set demonstrates that it could be used in real settings (e.g. homes, offices and other public areas) where fire usually begins as a small flame. Also, it is versatile enough to perform in settings where there is a large high-risk fire, such as forest fires.

Figure 7 shows some examples of the detections made on both test sets.

Figure 7: Images A, B and C show detections made on the single flame dataset. Images D, E and F show detections made on the high-risk fire emergency dataset. These detections demonstrate the model's ability to perform in many different settings and conditions.

Another important consideration is the speed and accuracy tradeoff between YOLOv3 and YOLOv3 Tiny. YOLOv3 is fast enough for real time detection but is not at the level of human visual cognition. YOLOv3 Tiny can operate well within that speed range although with a lower accuracy.

While YOLOv3 obtained an AP@0.5 in the 90% range, it can be improved for inference by applying some computationally inexpensive post processing steps. The confidence threshold of the detections could be changed dynamically when implemented in a video feed. If consecutive frames show a high confidence threshold, then the threshold could be lowered to detect more fire. If consecutive frames show a low confidence threshold then the threshold can be increased to eliminate those detections. Also, YOLOv3 could be used for initial detection and when the detections surpass a 90% confidence threshold (very high chance there is fire) then the model could be switched to the Tiny version for faster performance.

Further improvement could be made to the model by increasing the size of the dataset. Manual and automated web scraping can be used to find an additional 1000 to 2000 images. A pseudo-labeling process could be executed by having an already trained model run predictions on the new data. Any bad predictions would be relabeled manually, and the model would be retrained with the larger dataset.

While evaluating movement of the drone using the distribution of fire in the surrounding area offers more generalized movement it does not handle intricate movement very well. This was demonstrated in the large spikes in the graph for drone movement. The drone deviated from the path of the fire line when it encountered sharp turns or corners. Also, the distance the drone traveled is another indicator that it didn't perform with intricate movement. The total distance of the fire line was approximately 431 meters while the drone only traveled on average 370 meters. This means that the drone wasn't properly going around turns or corners, rather it was cutting across them which increased its distance from the fire line. Although, the drone did correct itself eventually, the situation isn't ideal.

Furthermore, the amount of fire detected by the model was low. The average number of fire grid cells per image was expected to be around 50-110. When the threshold is at the 5th column and the drone is flying parallel to the fire line, the number of grid cells on the right is 100. The number of detected fire grid cells should also be around that number. However, the amount of fire detected per image was around 8 grid cells. Some reasons for this include the fact that the models weren't trained with fire from Unity, and therefore didn't detect fire as well. Also, the results of the same model (YOLOv3-Tiny) in TensorFlow differ when it was implemented in Unity even though they should be the same (supplementary figure 8).

In Unity, the model tended to detect fire that was closer to the drone. This does offer some advantages because the direction the drone moves should be more influenced by the fire that is closer to the drone. But over time, the lack of detection for distant fires made the movement of the drone more imprecise and caused it to deviate.

Perhaps, instead of solely relying on a single algorithm (fire distribution) other techniques could be incorporated that handle specific scenarios (e.g. making a turn). GPS based navigation can always act as a failsafe if the main automation algorithm malfunctions. Since the drone would be conducting general monitoring and surveillance, it is not vital that the drone executes every movement to the exact GPS point and small errors in movement don't hinder the drone's ability to do its task.

It's important to consider that the simulation lacks details that are present in reality and is limited by the physics engine of the platform. Certain parameters such as the speed and height of the drone are kept constant and ideal weather conditions and visibility were assumed which is not necessarily consistent with real life. Smoke may also hinder the ability to detect fire, but its effect can be limited since the drone is viewing the fire line from the side and not directly above. Also, NIR (near-infrared) imaging might be able to reduce smoke hindrance and provide pictures that still retain the ground truth of the fire (Rossi et al., 2013). Real life testing of the drone with controlled fires needs to be done to validate the results of this project and to collect more image data on fires. Due to the limits of fire detection and automation, this project shows more potential for monitoring wildfires that are mostly under control and stabilized at the fire line rather than highly active wildfires. Future work can focus on improving the accuracy and speed of fire detection. One avenue for this is through knowledge distillation where a smaller model is trained to predict the inputs of a larger model (Hinton et al., 2015). Mapping functionality could be added to the drone in the form of triangulating the position of the fire based on the detections made in the video feed as the drone is moving. More details such as wind forces can be added to the simulation to better represent reality. Instead of explicitly programming instructions for the drone, a reinforcement learning model can be trained in the simulation. Further drone movement testing could be implemented through real life testing with candle flames.

CONCLUSION

In this paper, I have presented a lightweight, video-based fire detection system by leveraging the advantages of deep learning. This system was applied for drone automation through simulations and GPS navigation. I experimented with data preparation and model parameters to optimize the AP of object detection models for fire recognition. This system has the possibility of making fire detection more accurate and reliable for firefighting technology. The testing and evaluation done for drone automation shows its potential for practical application in wildfire monitoring. Automated drones may be able to provide critical information to firefighters so they can effectively control and reduce the environmental damage caused by wildfires.

ACKNOWLEDGMENTS

I would like to acknowledge the support provided by my family and teachers during the preparation of this project. I am very grateful and thankful for their support. Advice provided by my mentor, Matthew Emery, was greatly appreciated in refining and finalizing the deep learning aspect of this project.

REFERENCES

Apvrille, L., Tanzi, T., Dugelay, J.C. (2014). "Autonomous Drones for Assisting Rescue Services within the Context of Natural Disasters." 2014 XXXIth URSI General Assembly and Scientific Symposium (URSI GASS).

Analysis. Cair (2019). "Cair/Fire-Detection-Image-Dataset." GitHub, github.com/cair/FireDetection-Image-Dataset. DeepQuestAI (2019, June 28) "DeepQuestAI/Fire-Smoke-Dataset." GitHub. github.com/DeepQuestAI/Fire-Smoke-Dataset.

Barmpoutis, P., Dimitropoulos, K., Kaza, K., and Grammalidis N. (2019). "Fire Detection from Images Using Faster R-CNN and Multidimensional Texture Analysis." ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

BC Wildfire Service. (2020, Mar. 31) "Wildfire Averages." Statistics and Geospatial Data. https://www2.gov.bc.ca/gov/content/safety/wildfire-status/about-bcws/wildfirestatistics/wildfire-averages

Bochkovskiy, A., Wang, C.Y., Mark Liao, H.Y. (2020, April 23) "YOLOv4: Optimal Speed and Accuracy of Object Detection." ArXiv.

Cazzolato, M., Avalhais, L., Chino, D., Ramos, J., Souza, J., Rodrigues Jr, J., Taina, A. (2017). FiSmo: A Compilation of Datasets from Emergency Situations for Fire and Smoke

Endres, I., Farhadi, A., Hoiem, D., and Forsyth, D.A. (2010). "The Benefits and Challenges of Collecting Richer Object Annotations." IEEE Computer Society Conference on Computer Vision and Pattern Recognition.

Geoffrey, H., Vinyals, O., and Dean, J. (2015, Mar. 9). "Distilling the Knowledge in a Neural Network", ArXiv.

Jadon, A., Omama, M., Varshney , A., Ansari, S., Sharma, R. (2019, Sept. 4). "FireNet: A Specialized Lightweight Fire & Smoke Detection Model for Real-Time IoT Applications." ArXiv.

Microsoft (2019, Sept. 11). "Visual Object Tagging Tool." VoTT (Visual Object Tagging Tool), https://github.com/microsoft/VoTT.

Muhammad K., Ahmad, J., Mehmood I., Rho S., and Baik S. W. (2018, March 6). "Convolutional Neural Networks Based Fire Detection in Surveillance Videos." IEEE Access, 6, 18174–18183.

Phillips, W., Shah, M., and Lobo, N.V. (2002). "Flame Recognition in Video." Pattern Recognition Letters, 23(3), 319–327.

Rajchl, M., Lee, M.C.H., Oktay, O., Kamnitsas, K., Passerat-Palmbach, J., Bai, W., Damodaram,
M., Rutherford, M.A., Hajnal, J.V., Kainz, B., and Rueckert, D. (2017). "DeepCut: Object Segmentation From Bounding Box Annotations Using Convolutional Neural Networks." IEEE Transactions on Medical Imaging, 36(2), 674–683.

Ross, G., Donahue, J., Darrell, T., and Malik. J. (2014). "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation." 2014 IEEE Conference on Computer Vision and Pattern Recognition.

Rossi, L., Toulouse, T., Akhloufi, M., Pieri, A., and Tison, Y. (2013). Estimation of spreading fire geometrical characteristics using near infrared stereovision. Three-Dimensional Image Processing (3DIP) and Applications 2013.

Redmon, J., and Farhadi, A. (2017, Dec. 6). "YOLO9000: Better, Faster, Stron-

ger." 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
Savvides, L. "Fighting Fire with Fliers: How Drones Are Combating California's Record Wildfires." CNET, CNET, 27 Aug. 2018, www.cnet.com/news/californias-fires-face-anew-high-tech-foe-drones.
Shen, L. (2018, Nov. 30) "Here's How Drones Could Help With California's Wildfire Problems." Fortune. fortune.com/2018/11/29/drones-wildfires-california-drones/.
Shorten, C., and Taghi M. K., (2019, June) "A Survey on Image Data Augmentation for Deep Learning." *Journal of Big Data, 6*(1).
Statistics Canada. (2019, Oct. 30) "Wildfires." Get Prepared / Préparez-Vous. www.getprepared.gc.ca/cnt/hzd/wldfrs-en.aspx.
Sucuoğlu, S. H., Bogrekci, I., and Demircioglu P. (2019, Dec. 9). "Real Time Fire Detection Using Faster R-CNN Model." International Journal of 3D Printing Technologies and Digital Industry, 3(3), 220–226.
Toulouse, T., Rossi, L., Akhloufi, M., Pieri, A., and Maldague, X. (2018). "A Multimodal 3D
Framework for Fire Characteristics Estimation." Measurement Science and Technology, 29(2).
Toulouse, T., Rossi, L., Celik, T., and Akhloufi, M. (2015). "Automatic Fire Pixel Detection Using Image Processing: a Comparative Analysis of Rule-Based and Machine LearningBased Methods." *Signal, Image and Video Processing, 10*(4), 647–654.
Vidyavani , A., Dheeraj, K., Rama Mohan Reddy, M., and Naveen Kumar, KH. (2019, Oct.).
"Object Detection Method Based on YOLOv3 Using Deep Learning Networks." *International Journal of Innovative Technology and Exploring Engineering Regular Issue, 9*(1), 1414–1417.
Wei, L., Dragomir, A., Dumitru, E., Christian, S., Scott, R., Cheng-Yang, F., and Alexander,
B.C. (2016). "SSD: Single Shot MultiBox Detector." Computer Vision – ECCV 2016 Lecture Notes in Computer Science, 21–37.
Zhang, Q., Xu, J., Xu, L., and Guo, H. (2016). Deep Convolutional Neural Networks for Forest Fire Detection.
Zhu, X., Liu, Z., and Yang, J. (2015). "Model of Collaborative UAV Swarm Toward Coordination and Control Mechanisms Study." *Procedia Computer Science, 51*, 493–502.

ROBIN YADAV

Age: 16 | Surrey, British Columbia

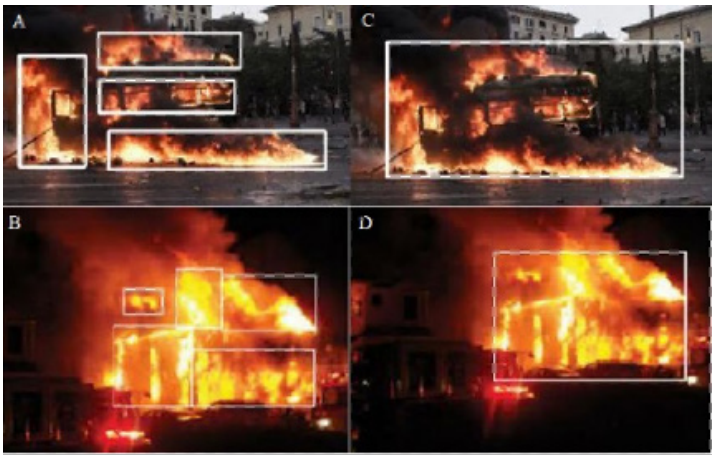
I'm very passionate about mathematics and computer science. I love solving challenging math problems and learning new skills in programming/computer science. Also, I enjoy volunteering in my community and reading about physics in my free time. I decided to pursue more advanced math and machine learning when I finished high school math and calculus in grade 9 and 10. This led me to my science fair project. I was inspired to work on an innovative solution that will integrate machine learning and wildfire technology after hearing how much damage wildfires cause in Canada.

SUPPLEMENTARY FIGURES

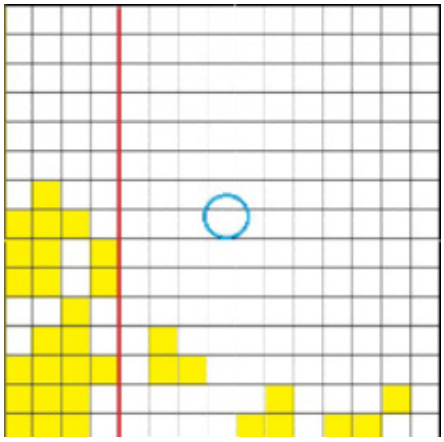
Figure 1: Example of a discarded image.
Figure 2: Examples of images in the test sets.
Figure 3: Example of the different methods of labeling.
Figure 4: A simple example demonstrating how the distribution of fire in a frame is analyzed.
Figure 5: Diagram of the components involved in the fire detection system.
Figure 6: Graph which shows the performance of YOLOv3 Tiny using data augmentation without HSV transformations.
Figure 7: Performance of YOLOv3 Tiny with varying amounts of raw data.
Figure 8: Detection example comparing the detection made in Unity and the detection made in TensorFlow.

Supplementary Figure 1: The only distinguishing feature of the fire in this picture is the light. However, the light emitted from the fire is a small portion of the image and is very similar to the other non-fire emitted light.

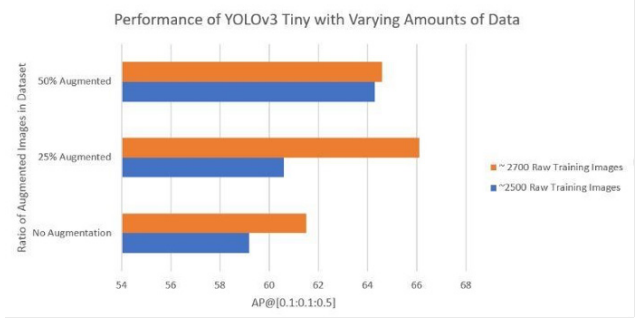
Supplementary Figure 2: The top 3 images are from the 50-image single flame test set. The bottom 3 images are from the high-risk emergency fire situations test set.



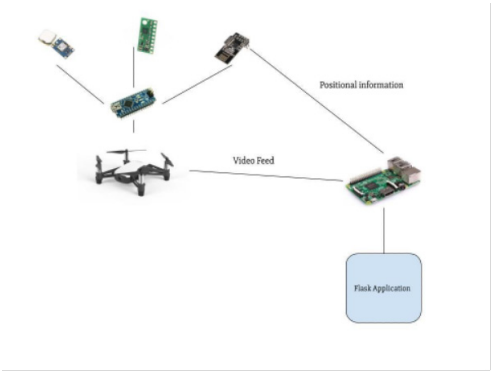
Supplementary Figure 3: Images A and B show segmented labeling where the fire is labeled according to the individual flames. Image C and D show their unsegmented counterpart where the whole fire is labeled as one object.



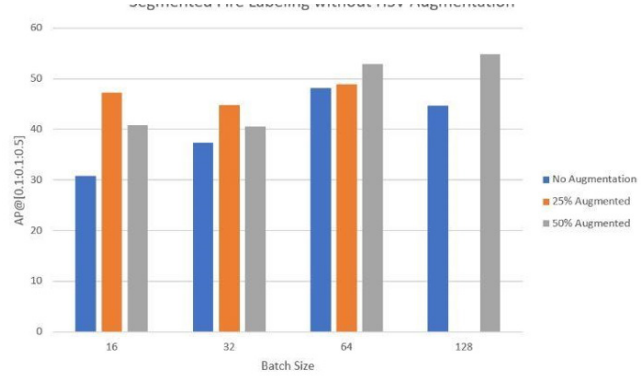
Supplementary Figure 4: As a simple example, consider a 15 by 15 grid. The red line represents the threshold and the yellow grid cells represent fire grid cells and white is background. Most of the fire is on the right side of the grid and only a few small patches are in the middle. This means that the drone is orientated parallel to the fire line (as most of the fire is on the right). The drone can move forward in the direction of the blue circle. With a larger grid the precision can be increased.



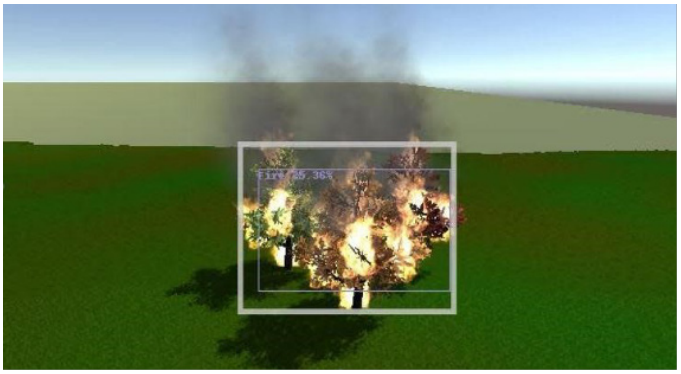
Supplementary Figure 7: YOLOv3 Tiny trained on 2700 raw images consistently has a higher performance than being trained on 2500 raw images.



Supplementary Figure 5: An Arduino Nano on the DJI Tello interfaces with a Neo6m GPS, LSM303 compass and NRF24 radio module. The positional information was sent to a Raspberry Pi, which was also connected to a radio module. The drone video feed was sent to the Raspberry Pi via a Wi-Fi connection. Inference was performed and results are displayed in a Flask application.



Supplementary Figure 6: Data augmentation without HSV transformations increases the performance of YOLOv3 Tiny compared to no augmentation across all batch sizes. The greatest performance is obtained when 50% of the raw data is augmented.



Supplementary Figure 8: The thicker grey bounding box shows the detection made by YOLOv3-Tiny in Unity and the thinner bounding box shows the detection when the same image was inputted into YOLOv3 in TensorFlow.