WILDFIRE DETECTION WITH YOLOV2

Ryan Beneduce[⋆], *Ryley Hill*[⋆], *and Caleb Schelle*[⋆]

[⋆]University of California San Diego, La Jolla, CA 92093-0238

ABSTRACT

Many artificial intelligence and machine learning models have been developed for smoke and fire detection in urban settings and are trained on man-made structures like buildings. This paper proposes a new dataset to train a YOLOv2 model to detect smoke and fire in natural environments, like the mountains and forests around California. The YOLOv2 model was chosen because of its high accuracy and ease of deployment onto an embedded system. The dataset was constructed using timelapses of fires recorded around southern California.

Index Terms— YOLOv2; smoke/fire detection; CNN; anchor boxes

1. INTRODUCTION

Wildfires are a dangerous threat to human life, the environment, and the economy of California. Detecting these fires in their infancy would provide responders valuable information about the location and reduce the resources needed to contain the fire. There currently exists an extensive network of cameras around California, Oregon, and Nevada that firefighters and volunteering citizens monitor for traces or smoke or fire in the camera feeds. This approach is expensive in human capital and monitoring is prone to human error.

There have been several approaches to detecting smoke and fire using novel machine learning approaches. Yuanbin et al. extracted smoke features from images using Gaussian Mixture Modeling (GMM) and input into a support vector machine (SVM) model that classifies the presence of smoke or not [8]. The images are preprocessed and enhanced with histogram equalization. The SVM was well suited for this project because of its simplicity, generalization ability, and nonlinear capabilities. The paper recorded a 96% detection rate with light shining near the smoke in a small area. A second paper, [3] focused on detecting smoke for wildfire applications using deep belief networks. This paper also used GMM to extract smoke features from images then fed the features into stacked layers of restricted Boltzmann machines (RBM) which make-up the deep belief network. The tuned model achieved a detection rate of 95%. The third paper proposed combining a Deep Convolutional Long-Recurrent Network with optical flow to detect fire and smoke. This method combines spatial learning through convolutional neural networks with temporal learning via long short term memory networks. Taking the difference between two sequential images creates an optical flow image showing the movement of smoke which is a feature that can be learned. This method achieved an accuracy of 93.3% [2].

The research conducted for this paper began with replicating the work found in [7]. The paper trained a YOLOv2 model on a smoke and fire image dataset from Kaggle [7]. YOLOv2 is a lightweight neural network with 21 layers that is ideal for deploying onto embedded systems. The model uses anchor boxes to extract features, in this case smoke and fire. The tuned YOLOv2 model achieved an accuracy of 96.8% in the paper.

Using cameras as opposed to traditional smoke detectors is preferable because of continuous sensing and faster detection capabilities. Traditional smoke detectors rely on smoke filling a room to the point where the sensor is triggered and an alarm sounds. The time between the fire starting and the sensor triggering may result in catastrophic damage to property or loss of human life. A camera looking at a room or building has a wide field of view where every pixel acts as a sensor and can detect smoke. With a fast model, a camera can identify the early stages of a fire just by the first wisps of smoke and notify the fire department. Applying the YOLOv2 model to time history footage of wildfires would reduce human resources and error.

The Kaggle dataset contained a wide variety of images ranging from stock images with black backgrounds to closed-circuit television (CCTV) footage of smoke in stores to traffic camera footage of vehicles combusting on interstates. Our research was created using the camera feeds from [4] posted on their Youtube channel. The camera network is extensive and creates a massive quantity of data that is ideal for a training machine learning model to be robust and accurate. Examples of the Kaggle dataset and sample images from the team's wildfire dataset are in Fig. 1.

2. DETAILS OF YOLOV2 MODEL

2.1. YOLOv2 Architecture

The YOLOv2 network consists of 21 total layers, beginning with an input image of 128x128x3 in size, and terminates from the YOLOv2 output layer with an image classification label. The hidden layers are composed of 3 cascades of convolution, batch normalization, and max pooling layers, utilizing a relu non-linearity function. Fig 2 displays the YOLOv2 architecture implemented to detect wildfire.

The YOLOv2 architecture is characterized by its subnetwork for object detection, which is initialized by the batch normalization feature extraction layer. Within the YOLO subnetwork a condensed image is divided into an equally spaced grid, where pre-sized anchor boxes predict bounding boxes to predict a label classification with a defined confidence score to display.

2.2. Feature Extraction

2.2.1. Ground Truth Labels

Ground truth labels are generated by the user. Image sequences were imported into the MATLAB application imageLabeler where a bounding box is drawn around the smoke or fire and is defined as the ground truth. The training, validation, and test datasets must all be labeled with ground truths. For the Kaggle dataset, the ground truth labeling was separated into two classification labels: smoke and fire, similar to Fig. 3.

The dual classification slowed the training and hindered the model's accuracy to classify either of the labels. The Kaggle model also failed to classify images with multiple labels at once. To counteract this, the group altered their approach towards classification labeling as well as created a training dataset with multiple labels at one time.

For the wildfire dataset, the ground truth labeling was compressed into a single classification label: wildfire. The motivation behind this fundamental change in labeling can be associated with the motivation behind this research. The objective was to classify the onset of a wildfire. For the purposes of labeling, the difference between smoke and fire was negligible, as both instances signaled a wildfire. This alteration in groundTruth label philosophy is highlighted in Fig. 3.

The classification labels also presented the opportunity to train the network how to distinguish between cloud coverage and wildfire smoke. As Fig. 3 depicts, a wildfire can be identified by a small plume of dark smoke emitting from ground level. These alterations provided the model the ability to detect multiple wildfires at once, or fires along the horizon.

2.2.2. Anchor Boxes

Anchor boxes are a set of predefined rectangles that best match the desired features. Their purpose is to reduce computation time by eliminating the requirement of scanning the entire image to predict the absolute size of bounding boxes. Instead, a predicted bounding box is scaled with respect to the anchors. Given ground truths prescribed in our training data we performed k-medoid clustering to estimate the preferred anchor boxes. Our model space and tunable hyperparameters included 4, 8, and 12 anchor boxes (Table 1).

Each predicted anchor box is defined by five parameters: X, Y, W, H, and Confidence, defined as the x and y coordinate of the centroid of the anchor box, the width and height of the anchor box, and the confidence score that the box contains an object. The confidence score is computed (equation 1) [1] and defines the probability of the class label within the model's anchor box. If there is an object detected within the anchor box, the confidence score is equal to the Intersection over Union (IoU) value between the ground truth labels and bounding boxes [7].

$$\begin{aligned} p(Class_i|Object) \cdot p(Object) \cdot IOU_{pred}^{truth} = \\ p(Class_i) \cdot IOU_{pred}^{truth} \end{aligned} \tag{1}$$

The loss equation, (equation 2), shows how YOLOv2 optimization has multiple parts. The first term calculates the distance between the ground truth bounding box and the anchor boxes of each cell. The second term is similar to the first and calculates the difference in width and height between ground truth and each anchor box. The third and fourth terms calculate objectness or lack thereof and update the error accordingly. The final term calculates classification loss [1].

$$\begin{aligned} Loss = \lambda_{coord} \cdot \sum_{i=0}^{S_x \cdot S_y} \sum_{j=0}^{B} 1_{ij}^{obj} (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \\ + \lambda_{coord} \cdot \sum_{i=0}^{S_x \cdot S_y} \sum_{j=0}^{B} 1_{ij}^{obj} (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \\ \sum_{i=0}^{S_x \cdot S_y} \sum_{j=0}^{B} (C_i - \hat{C}_i)^2 + \lambda_{noobj} \sum_{i=0}^{S_x \cdot S_y} \sum_{j=0}^{B} (C_i - \hat{C}_i)^2 \\ + \sum_{j=0}^{B} 1_i^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2 \end{aligned} \tag{2}$$

2.3. Wildfire Detection Methodology

2.3.1. Model Objective

Since the goal is to detect the earliest signs of a dangerous forest fire, both smoke and or fire are significant indicators worthy of notifying firefighters. Instead of classifying both fire and smoke similar to [7] we decided that a single feature was preferred which we chose to label as wildfire. This feature therefore encompasses any instance of smoke/fire. We made an effort to include inside bounding boxes the soil/canopy and source of the fire during the ground truth labeling so that the feature selection might differentiate itself better from clouds which are not regularly joined to the ground like fire smoke is.

Hyperparameter	Value
Anchor Boxes	4 ; 8 ; 12
Optimizer	SGDM ; ADAM
Learning Rate	1e-3 ; 1e-4
Mini Batch Size	8 ; 16
Max Epoch	80 ; 160

Table 1. Hyperparameter space. Green values represent preferred model.

Wildfire	No-Wildfire
TP = 433	FP = 30
FN = 0	TN = 229

Table 2. Confusion matrix of tuned model. Mean IoU = 38.5% ; Accuracy = 95.7%

2.3.2. Data Preprocessing

A suite of 4020 images consisting of 23 fires were first identified via the High Performance Wireless Research and Education Network (HPWREN) YouTube channel and downloaded using their online user portal [4]. Half of the fires which were older than 2020 were only available in .MP4 format and converted to individual .JPG images. We consciously tried to choose a wide selection of fires that were variable in distance to the camera- ie. near, mid, and far. Also, we consciously included fires that were variable in time of day- ie. day time, night time, and times during direct sunlight like sunset and sunrise. We split the images into 2,666 training, 662 validation, and 692 test images. These images correspond to [17,3,3] time lapse fires respectively. We considered normalizing our pixel values from [0,255] to [0,1] range, but the Stanford Lecture 6 (28:59) [6] indicates that it is unnecessary to normalize pixel values since "in images, at each location, we already have relatively comparable scale and distribution" which is appropriate for us as well. We also considered but did not implement zero-centering and pixel standardization. We think exploring these normalization would be valuable for future work.

Images were then resized to 128x128x3 pixels to accommodate the requirement of the YOLOv2 architecture. Then, ground truth labels were applied to each image to produce the arrays of labeled features. To expedite this process as opposed to drawing boxes one frame at a time we used the built in temporal interpolation feature of the MATLAB videoLabeler application [5]. It should be noted that the use of this feature is why our data set of fire images is not randomized and is instead sets of sequenced fires. We think this approach is also preferred since it keeps validation and test fires entirely independent from training.

2.3.3. Training

With our preprocessed images and drawn bounding boxes the training set was now ready for training the YOLOv2 detector. We explored a variety of hyperparameters in order to settle on our preferred model. We tuned the hyperparameters for the model based on 3 anchor box choices, 2 optimization techniques, 2 learning rates, 2 mini batch sizes, and 2 epoch sizes arriving at a total explorable hyperparameter space of 48. Our hyperparameter space is outlined in Table 1. We also explored 3 different middle layers early on but found that the default [7] was considerably better. Fig. 5 showcases a typical Loss/RMSE vs Epoch plot of one of our SGDM models.

3. RESULTS

The built-in MATLAB functions for precision and miss rate were inadequate for quantifying our model performance. We wrote a custom script to test the confusion matrix criteria of each of the generated YOLOv2 detectors on both the validation and test set. The detector will generate bounding boxes based on a prescribed confidence. We opted to use a threshold confidence of 20%- meaning that the detector would generate bounding boxes on the image when it was at aleast 20% confident that there was a wildfire. We chose this value lower than the default 50% value since we would rather potentially misclassify wildfire with a False Positive (FP) as opposed to entirely miss a wildfire with a False Negative (FN) (Fig. 4 - A). To classify an image as a True Positive (TP) the detector must generate a bounding box that is at least 20% IoU with the ground truth. This IoU threshold for classifying TP is also tuneable in our script but we thought this value was fair since the main goal is to simply detect the fires and not precisely locate them. Some images will have multiple bounding boxes detected (Fig. 4 - C). If at least one of these bounding boxes had at least 20% IoU with ground truth then it was classified TP, otherwise it was classified as FP. For example, in (Figure 4 - B) the two yellow squares of our detector were produced. If it was only the box outside the ground truth (green rectangle) it would be a FP, but because another box accounted for at least 20% of the ground truth it is a TP. For our dataset this approach seemed the most appropriate since often only one fire is in the image. However, we did notice that one of our wildfire time lapses in the test dataset contained two fires. A small number of times our detector correctly predicted one but not the other. Instances like this would require reevaluation of this confusion matrix criteria method as we currently classify that image as a TP since we correctly guessed one of the wildfires although half of it is a FN since we missed the other one. We suggest in the future a similar confusion matrix criteria based on the number of ground truth boxes as opposed to number of images would be more appropriate. With our evaluation metric decided we could then pass each of our detectors generated from our different hyperparameter

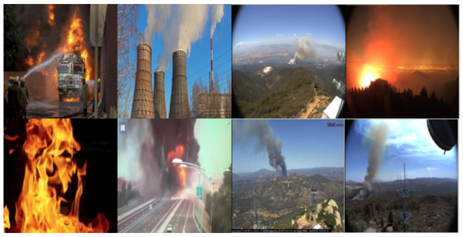

Fig. 1. Sample Images from Kaggle Dataset (left) and Wildfire Dataset (right)

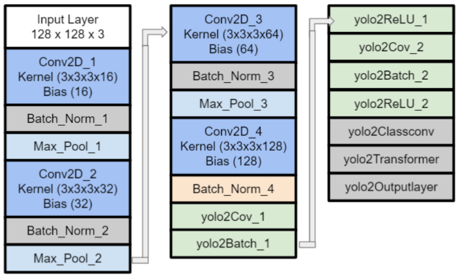

Fig. 2. Details on YOLOv2 Architecture

space through the script and evaluate its performance on the test dataset. Again, since we are keen to detect wildfire but not miss them we decided that the preferred model would be the one that maintained high accuracy but also had a low FN rate. Our preferred detector did extremely well. The results of the confusion matrix criteria are presented in Table 2. We found that this detector also outperformed other models on the validation data set as well.

4. CONCLUSION

We propose the use of YOLOv2 CNN for the use in real-time wildfire detection. By tuning a variety of hyperparameters we arrived at a robust detector capable of 0 FN and an accuracy of 95.7%. We demonstrate this approach with historic wildfire camera data from HPWREN. The proposed YOLOv2 detector is very appropriate for implementation in real-time for its low-latency and high performance compared with other region-based object detection networks. We foresee future work like increasing the number of wildfires in the database and a more detailed confusion matrix criteria is necessary to make affirmative conclusions, but our preliminary work shows great promise in object detection for wildfire response and mitigation.

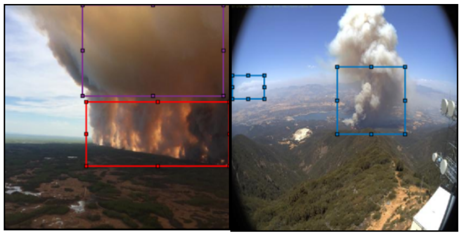

Fig. 3. Ground Truth Labeling for: Kaggle Dataset (Left) and WildFire Dataset (Right)

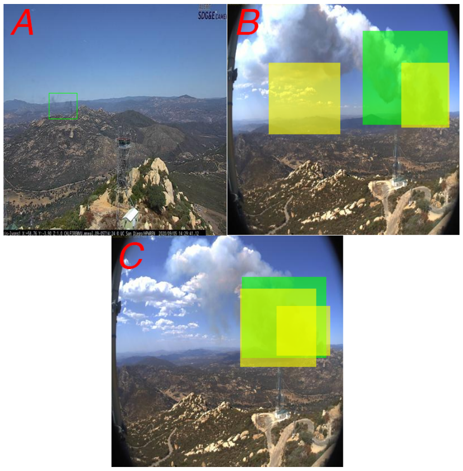

Fig. 4. Average IoU as a function of anchor boxes per grid cell

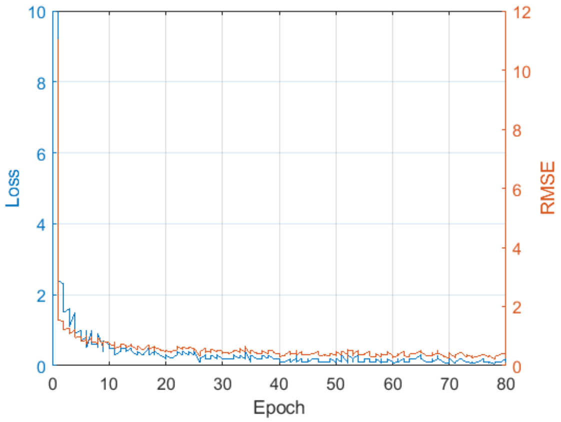

Fig. 5. Loss/RMSE vs Epoch for SGDM, 8 anchor boxes, learning rate 0.001, epoch 80, minibatch size 16

5. REFERENCES

[1] Gianluca Giuffrida, Gabriele Meoni, and Luca Fanucci. A yolov2 convolutional neural network-based human–machine interface for the control of assistive robotic manipulators. *Applied Sciences*, 9(11):2243, 2019.

[2] Chao Hu, Peng Tang, WeiDong Jin, ZhengWei He, and Wei Li. Real-time fire detection based on deep convolutional long-recurrent networks and optical flow method. In *2018 37th Chinese Control Conference (CCC)*, pages 9061–9066. IEEE, 2018.

[3] Rabeb Kaabi, Mounir Sayadi, Moez Bouchouicha, Farhat Fnaiech, Eric Moreau, and Jean Marc Ginoux. Early smoke detection of forest wildfire video using deep belief network. In *2018 4th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*, pages 1–6. IEEE, 2018.

[4] Nevada Seismological Lab. URL: `http://www.alertwildfire.org/sdge/index.html?v=7a7f1c3`.

[5] Matlab optimization toolbox, 2020. The MathWorks, Natick, MA, USA.

[6] Stanford University School of Engineering. Lecture 6 training neural networks. URL: `https://www.youtube.com/watch?v=wEoyxE0GP2M&t=200s`.

[7] Sergio Saponara, Abdussalam Elhanashi, and Alessio Gagliardi. Real-time video fire/smoke detection based on cnn in antifire surveillance systems. *Journal of Real-Time Image Processing*, 18(3):889–900, 2021.

[8] Wang Yuanbin. Smoke recognition based on machine vision. In *2016 International Symposium on Computer, Consumer and Control (IS3C)*, pages 668–671. IEEE, 2016.

6. CRITIQUES FROM OTHER TEAMS

Add some more information about any preprocessing of the data. (Cleaning of data, any features given to network other than raw image pixel data):

- *There was minimal preprocessing done to the data. There were some images that were not suitable for training or testing, and they were removed from the datasets. The images were all reduced in size to 416x416x3 before being input into the model. Images were all [0-255] scale- we did not normalize to [0-1] as explained in the report text and Stanford Lecture [8]. Future work may include centering and standardization.*

Adding more information about how the ground truths were generated or found. Was it labeled by the group members or was it found with the dataset that was used.

- *The groundTruth labels were generated by the group through the imageLabeler and videoLabeler applications in Matlab.*

It would be helpful to see the best and worst performing classification that the network made in comparison with the ground truth of where the smoke or fire is, i.e how much overlap there was. It should be shown both visually and in terms of the IOU measure that the group was using.

- *Refer to Figure 8: A,B, and C.*

You could add any other implementations you tried and those implementations' results. For example, any other network architectures used.

- *Before the final network was created, the group implemented several different iterations of middle layers. A few of the adjusted layers the group attempted were: adding dropout instead of batch normalization, creating a deeper network with multiple sets of cascade layers (conv, batch norm, maxpool) and adjusting the weight/bias initializers. The final results from the original network proved to be the most efficient and accurate so the group decided to stick with the original network.*

Please give more explanations on the anchor box and the bounded box. What's the difference between those two?

- *The bounding box we refer to is the ground truth box drawn around our fire and smoke. An anchor box is generated by the code and tries to "bound" the same ground truth object. One parameter our team tuned was the number of generated anchor boxes. Using more anchor boxes makes it more likely that the correct object is captured but is computationally expensive.*

In the plot of loss vs. iteration in page 10, it will be better to show both the training and validation results. In the video, you mentioned that there's no overfitting. No overfitting means that the training and validation loss do not have a significant difference. So it will be better to show both the training and validation loss to prove that there's no overfitting.

- *We understand that plotting the validation curve alongside the training curve may provide an easy visualization of the presence of overfitting, but we determined the lack of overfitting differently. Graphically, we see the training curve flatten and diminishing returns from each additional epoch. This phenomenon means that our model is not overfitting. If it continued to plummet with increasing epoch that would indicate it was overfitting. Additionally, upon testing the model on both the validation and test datasets, we saw similar accuracy values as the training data, indicating our model was not overfit.*

On page 11, what is the reason that the recall rate is lower in Kaggle dataset than in wildfire dataset? We guess the reason might be that the images have various different environment settings in the Kaggle dataset, but the images have very similar environment settings in the wildfire dataset.

- *Recall rate is the sensitivity of the model and it is true that our wildfire model had a higher recall rate than the Kaggle dataset. The increased sensitivity may be because the similarity between wildfire images was much higher than the Kaggle dataset. The wildfire model would probably be much less robust than the Kaggle model. One way to verify this would be testing the wildfire model on the Kaggle dataset and the Kaggle model on the wildfire dataset. Our team did aim to have maximum sensitivity to potential fires to reduce the possibility of false negatives.*

You mentioned a lot of hyperparameters, it would have been better to see them on the screen because there were a lot of them.

- *A table showing the tuned hyperparameters has been added to the report, see Table 1*

You mentioned wanting to use infrared images in the future, have you thought about how you will have to relabel everything to teach it how to identify fire in those types of images.

- *The infrared images might provide valuable information that traditional RGB images can't capture. We are not sure how easily this would be incorporated with our current model but would make an excellent future work project.*

It seems that you find your anchor boxes and then resize your images? Can you discuss how this won't mess up the location of the box. Do you draw it on before resizing or do you use padding?

- *The images are resized before the ground truth boxes are drawn on the images. .*

7. CONTRIBUTIONS

Each team member contributed to all aspects of the project. All three team members trained and tested models while exploring the hyperparameter space. Ryley created the wildfire dataset, Ryan tested additional layers within the YOLOv2 architecture, and Caleb characterized the hyperparameter space. Everyone contributed equally to each deliverable including the proposal, progress report, presentation, critiques of other teams, and this final report.