EY Challenge 1: Fire Mapping

Fire mapping from airborne imagery

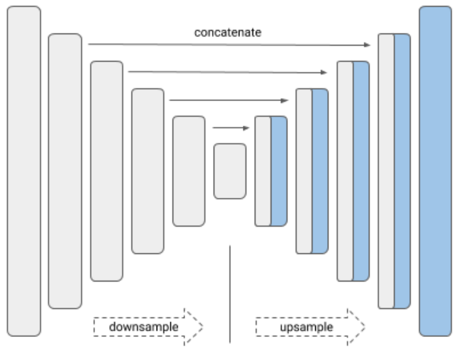
Yash JAIN, Yash RAJ, Chandrashish PRASAD, Arpan BISWAS

KEY ELEMENTS

Our approach involved employing deep learning methods to train a model for the task of semantic segmentation by classifying each pixel as either 1 (denoting on fire) or 0 (denoting not on fire). Different matching techniques were used to map the polygons to their corresponding linescan images and to deal with smaller training set, augmentation was used.

TRAINING METHOD USED

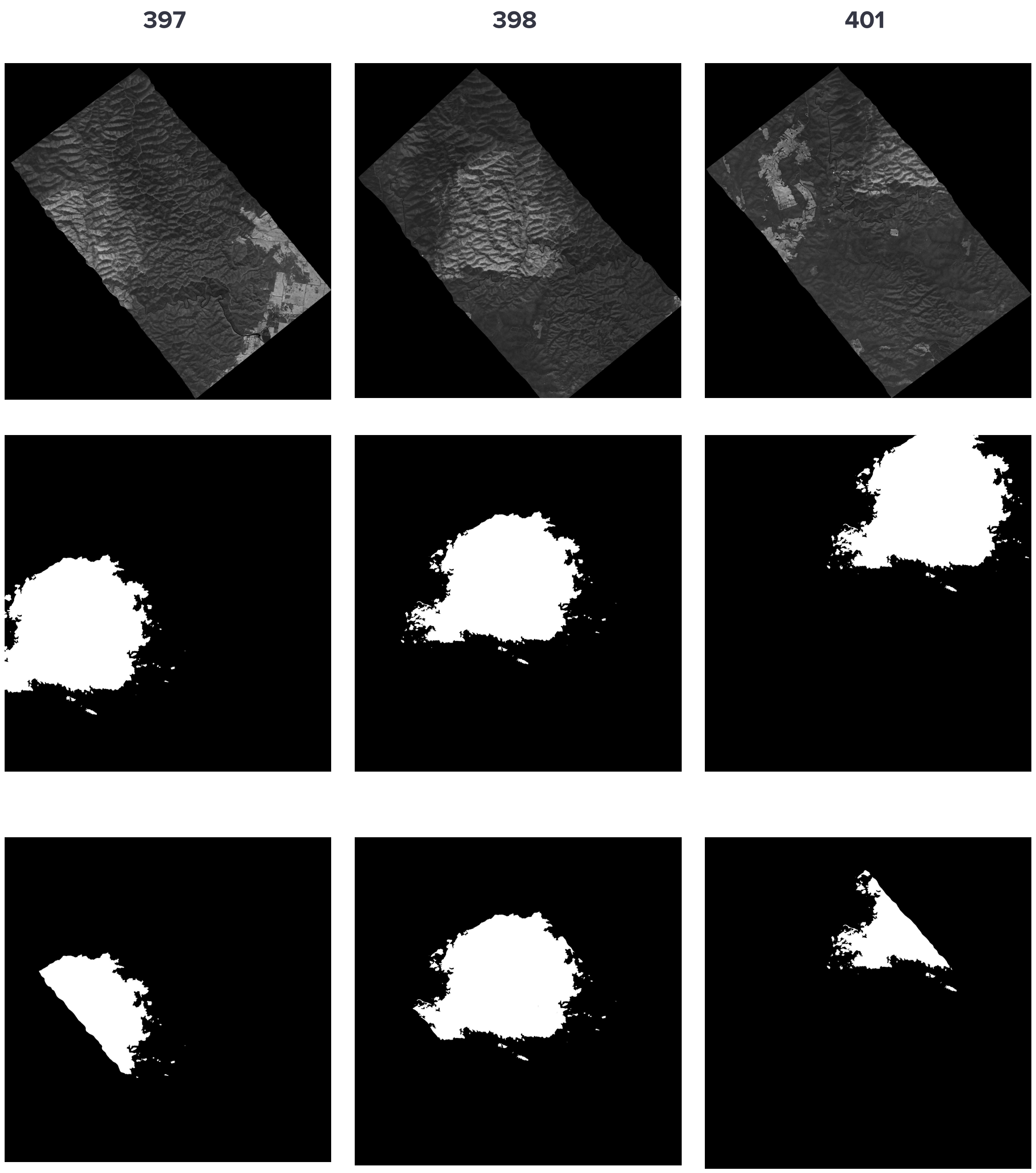
UNet with ResNet34: The UNet architecture contains two paths. The first path does the encoding (contraction) with a series of convolutional and max pooling layers. It captures the context in the image. The second path does the decoding (expansion). We add the subsequent layers and the pooling operations are replaced by upsampling operators which increases the resolution of the output images from the encoder. This enables precise localization using transposed convolutions. It only contains Convolutional layers and does not contain any Dense layer because of which it can accept images of any size.



ResNet34: It is a 34 layer convolutional neural network model that has been pre-trained on the ImageNet dataset- a dataset that has 100,000+ images across 200 different classes. However, it is different from traditional neural networks in the sense that it takes residuals from each layer and uses them in the subsequent connected layers.

APPROACH TO FEATURE ENGINEERING

The foremost thing to begin with the challenge was cleaning the polygon dataset. We identified that every composite polygon's source name had some clue for matching it to the linescan image. We termed these identifiers as “Fire_ID. For example: `Composite wallhalla 397, 398 & 401 20190225 (1311 to 1342hrs)` gave us the Fire_IDs **397**, **398** and **401**.

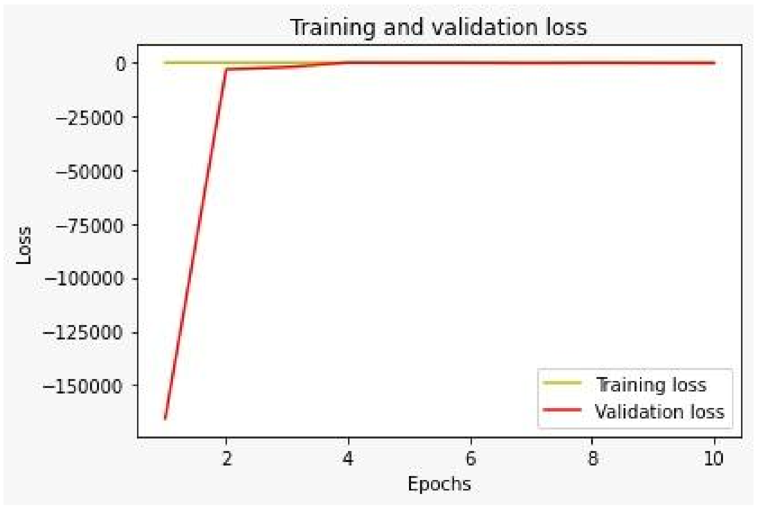


After extracting the same from the linescan labels, an Inner Join was used to merge the two dataframes. We used these to save .jpeg images and their corresponding masks. To increase the size of the training data, augmentation (albumentations) was used by applying transformations, flips and rotations on the saved images.

APPROACH TO MODEL VALIDATION

As the number of images (original) for training was small, the validation is not prefered on a large fraction of training images. So, taking 10% of training data as the validation data was a viable option.

The optimum number of training epochs for the model came out to be five empirically as training on more epochs resulted in similar results. After using different loss functions, Binary Cross Entropy achieved more promising results as it converged faster. The metric used against the loss function was MSE (Mean-Squared Error)



HIGHEST PERFORMING FEATURES

The training of the model was done on different batch sizes and number of epochs. The batch size of 4 gave encouraging results. The performance deteriorated as we gradually increased the batch size from 4 to 8 and then 32.

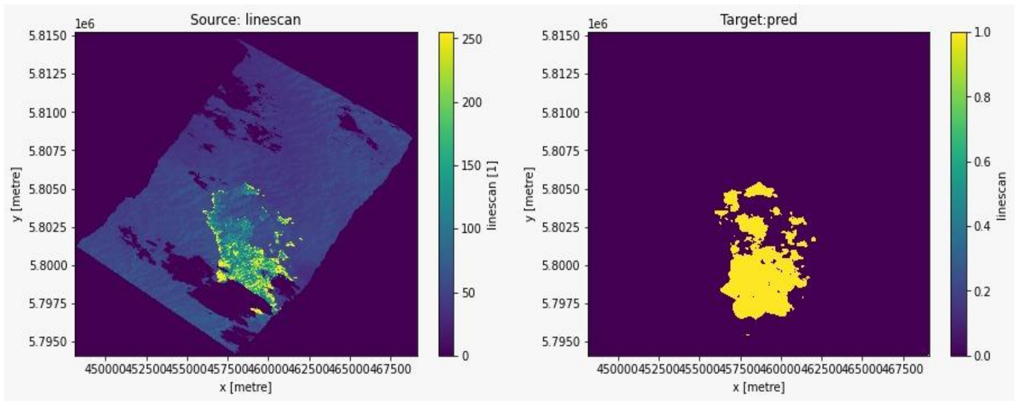
After training, three different models with 256x256, 512x512, 1024x1024 input image size, the one with 1024x1024 as input image size gave the best prediction. It takes more time per epoch to train but needs a lesser number of epochs.

KEY DECISION POINTS

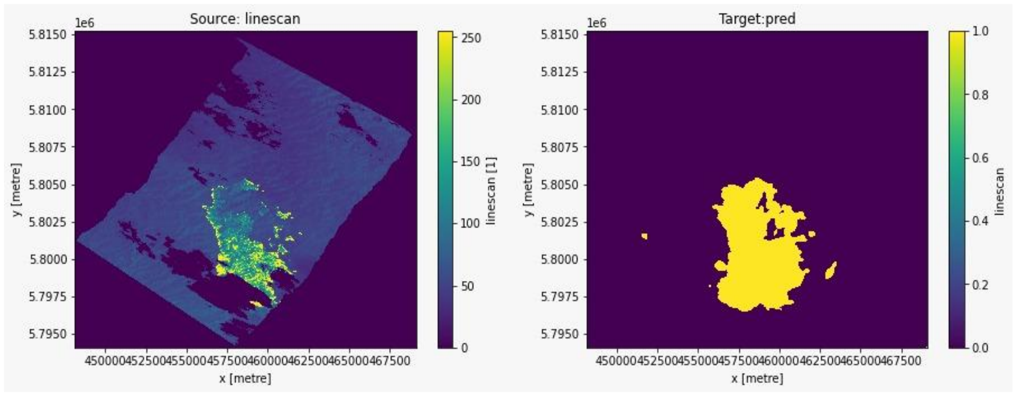
Streamlining our focus towards deep learning methods from the other methods like XGBoost and Ruptures was a key decision point. Initially, using Ruptures (a python library) to find spikes in linescan values and classifying the pixels within that region as fire gave us a 0.75+ score but the model was trained with some underlying assumptions which may not have been the best for the problem in hand.

The model was trained on 1024x1024 size images but while predicting, 512x512 images were used to get a more solid polygon-like region which improved the polygon continuity at each pixel by adhering to its surrounding pixels.

1024x1024

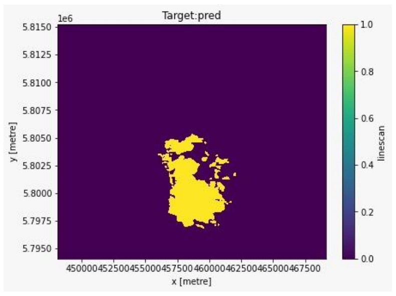


512x512

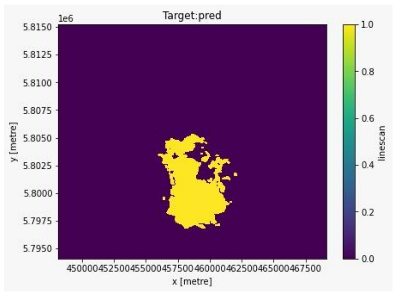


Another key decision point was fixing the optimum probability threshold for predictions which was done more in a trial and error fashion.

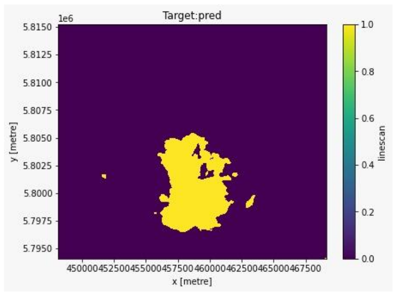
0.5



0.2



0.01



Masks for different probability thresholds

UNIQUENESS IN APPROACH

1. Since data cleaning was an integral part of the challenge, our approach for matching polygons does not require any time buffer and is in a lot of ways immune to human error as labels are auto generated by the device as per the event. Since, what we termed as Fire_ID is a part of the linescan label, it will always be present in both the gdf dataframe and training dataframe. Things like time and date are more susceptible to human inaccuracies.
2. We built the model using UNet which is mostly used in Biomedical and microscopic images. It is because of this we didn't require several thousands of training images as it adapts to class imbalance well and gives quite good results on smaller training datasets too. It is also quite fast.

LIMITATIONS

The variability in the dimension of the image is an issue. Although our model works for a wide range of input image dimensions, images with a much higher dimension (more than any image in the training set) can give poor results.

Given the nature of the problem, we made sure that our model does not underfit the data. In pursuit of that, we built a model that gives slightly more false positives.

Train time is somewhere between 5-8 hours (which is not unheard of for a deep learning model) and it requires a machine with at least 32 gbs of RAM.

CAN THE MODEL BE USED OUTSIDE AUSTRALIA?

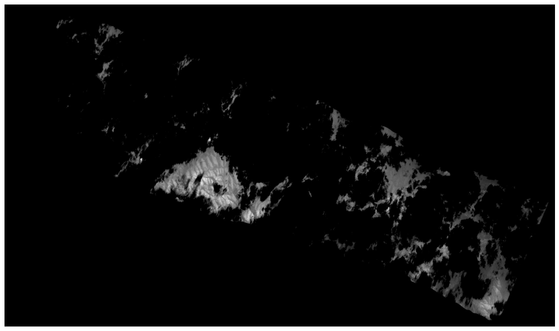
We didn’t make any strong assumptions regarding the specific type of data required by us, while approaching the problem. So, yes the model can indeed be used outside Australia. We made conscious efforts to build our model such that it generalizes better over a wide range of situations.

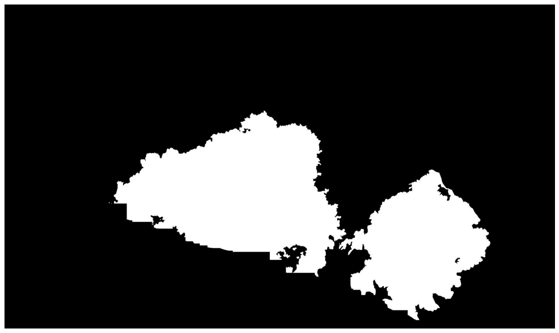
OTHERS

Our method can be implemented to work on the images of much higher resolution by generating smaller crops of the image and training on that. It gave us promising results but due to paucity of time, it could not be executed completely. Trying Dice Coefficient Loss instead of Binary Cross Entropy may also improve the result.

One of the difficult parts of the challenge was to come up with a suitable model for this task as the number of training images was quite less.

Due to the abundance of clouds in some fire events the corresponding masks can give misleading predictions for a pixel since the linescan values for clouds are zero.



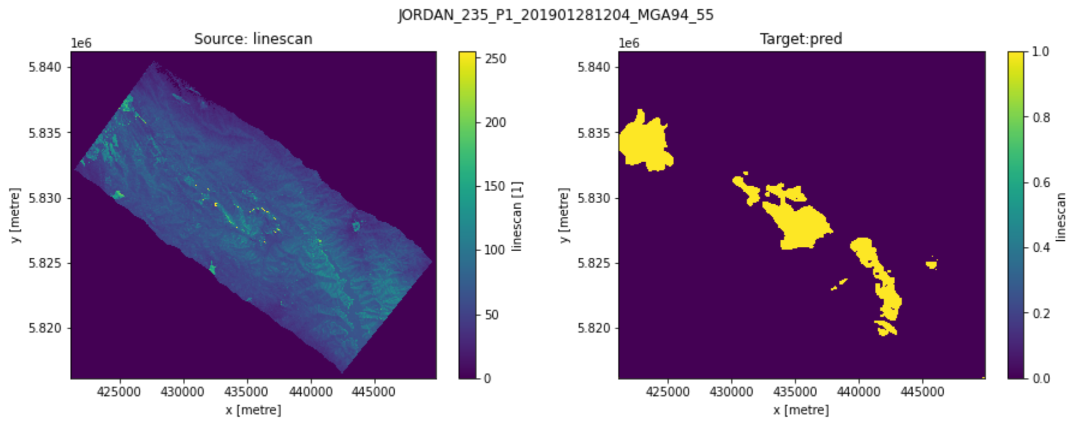


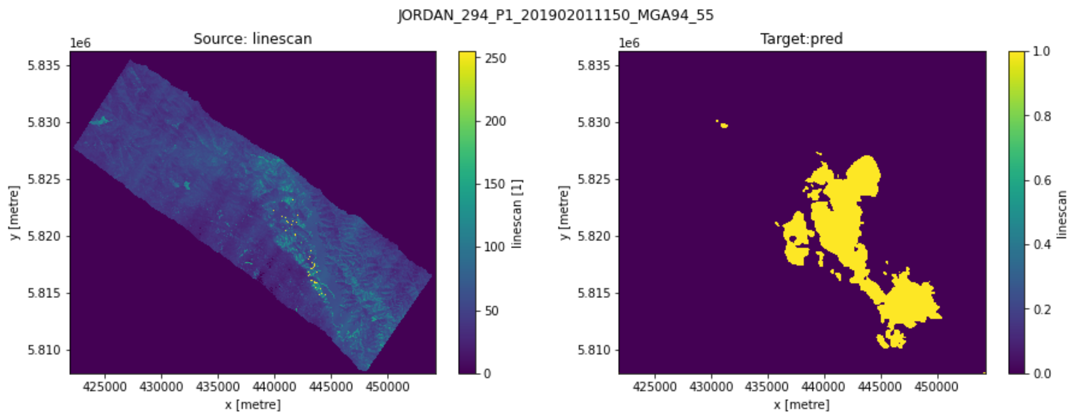
Our model took 3-4 minutes to run the predictions for all the five test images and 5-10 sec to identify whether each coordinate pair was on fire or not in the given linescan images.

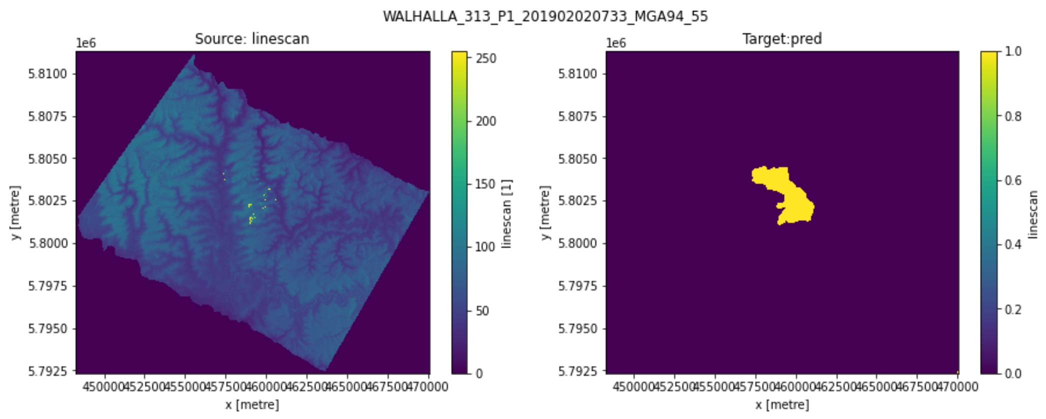
Prediction Summary

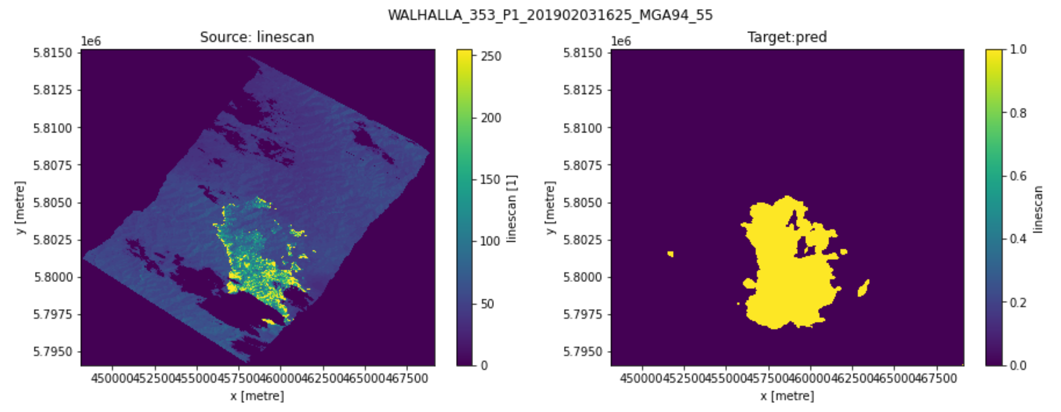
Best score on a fraction of data (as per submission rules and metric) was 0.75+

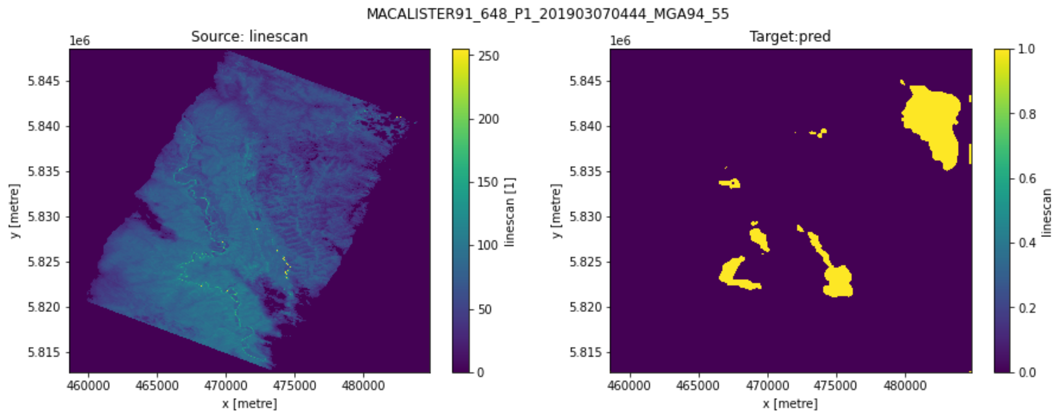
<u>Predictions (on five test images)</u>











Summary of prediction in **test.csv**:

```
0.0    3607
1.0    1393
Name: target, dtype: int64
```