**Approach**

The key elements of this approach are the modified [challenge 1 jupyter notebook](https://drive.google.com/file/d/1LqP9p2ELGigTtJIfRqHV6DSWk2Yk80GM/view?usp=sharing), [ArcGIS Pro](https://drive.google.com/drive/folders/1FdFG7R9p8drVDiUziY7wcmWER7_jJfgH?usp=sharing) and careful selection of less noisy data for training.

* **Use of Challenge 1 Jupyter Notebook and ArcGIS Pro**

The data set polygon extraction is the first headache to this problem. To extract these polygons, ArcGIS Pro plays a very important role. The shape files are loaded into ArcGIS Pro. The coordinates of the images are taken from the open data cube jupyter notebook, plotted in ArcGIS Pro and the coordinates’ demarcation are used to extract the polygons. The selected polygons are rasterized using the open data cube functions and the image is extracted using the same boundaries of the original image coordinates. This is illustrated in the [video](https://drive.google.com/file/d/1oYaQZjNZHL-pCo-uj7I0C6whXebGIeMj/view?usp=sharing).

* **Dataset**

Out of the total 129 images that were provided for training. 23 were images (id-0 to id-22) did not have any shape files for extraction using ArcGIS Pro. Coincidentally, these 23 images are the images for the second challenge. Out of the remaining 98 images, 24 are very noisy and there are intentionally not extracted.

Noisy images fall under these categories:

1. Linescans that had a lot of black areas in the image and were not clear enough.
2. Linescans that had their mask only in black of the image after polygon extraction.
3. Linescans that did not match their masks after side-by-side comparison.

These images are avoided to prevent the model from always learning that black areas are areas of fire.

Less noisy images fall under these categories:

1. Images with smoke around fire
2. Images with unusual surfaces after satellite image taken by satellite

Of the remaining 74 images, 49 images are used for data augmentation for training and testing. The remaining 25 are used validation of the model.

After getting prediction from the trained model, the images are then loaded to ArcGIS to selects points that fall within white areas. After getting points from ArcGIS Pro using this [excel file](https://drive.google.com/file/d/1rlaKLxl5X52juGMHtEetSrX7Q9kneXaz/view?usp=sharing) for coordinate conversion, this [jupter notebook](https://drive.google.com/file/d/1-eAM7_hECnvASzoDOAmnBztF6WJ9Wkpk/view?usp=sharing) helps to automatically select the right points.

**Network Architecture**

The network consists of a contracting path (left side) and an expansive path (right side). The contracting path follows the typical architecture of a convolutional neural network. It consists of a repeated application of two 3x3 padded convolutions (specifically a pad of one), each followed by a rectified linear unit (ReLU) and a 2x2 max pooling operation with stride of 2 for down-sampling. At each down-sampling step, we double the number of feature channels. Every step in the expansive path consists of an up-sampling of the feature map followed by a 2x2 convolution (up-convolution) that halves the number of feature channels, a concatenation with the correspondingly same-sized feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU function. **Before the feature map gets to the each ReLU function for the channel output, batch normalization is used to improve the network’s learning rate. A padding of one prevents the input feature from being cropped which makes the network work good irrespective of the image size used for training**. At the final layer a 1x1 convolution is used to map each 64- component feature vector to the desired number of classes. In total the network has 23 convolutional layers. In general, the whole network has a UNET architecture.

**Feature Engineering, Feature Selection & Training**

As the training image sizes vary, the model is trained with 512x512 sized images. The UNET model has been trained at a learning rate of 0.004 with 2 workers to simultaneously feed data to the RAM. As a result of noise in the data, only 49 carefully selected images with no or less are selected for training as **explained in the dataset section**. All 49 are used for data augmentation. Out of 196 augmented data, 24 are used for testing. In each batch, fifteen images are loaded. The model runs for 200 epochs with an Adam optimizer.

**Model Validation**

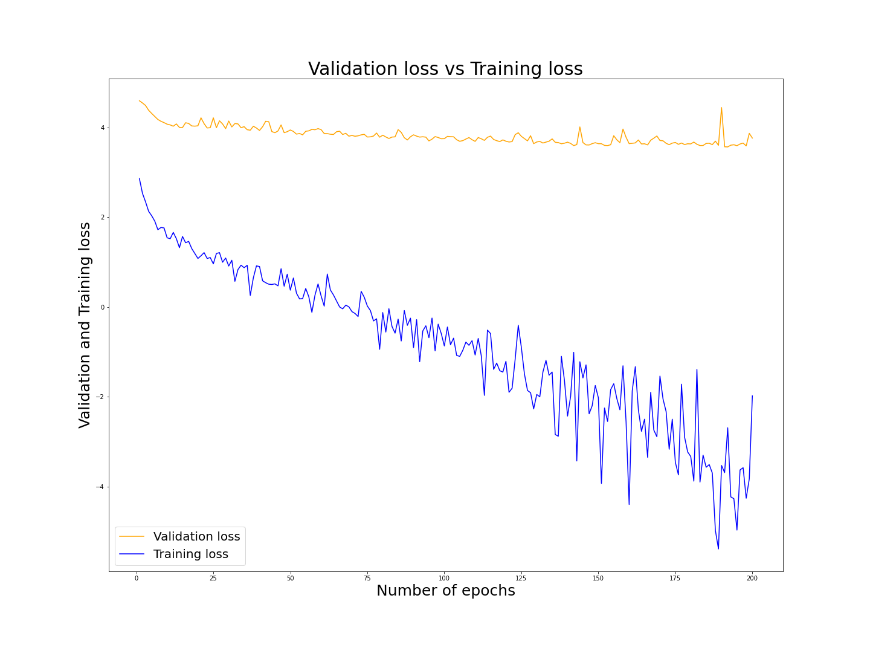
As mentioned earlier, 74 images are used in the model from training to validation. 49 images are used for data augmentation for training. These 49 images are carefully selected from all fire events. These images have less or no noise. 24 images are used for the testing set. This formed 13% of the whole set of images used for training.

25 of the remaining datasets that the model did not see are used validation of the model validation after training (some of these 25 images are very noisy and are there on purpose to see how the model performs on noisy images). Dice scores of testing sets is 81% and the dice score of validation sets ranged from 51% to 96% taking individual images into consideration. Noisy data gives less dice scores (55% to 68%), less or no noise data gives better dice scores (71% to 96%). This were the best results after different outcomes.

The 23 images (second challenge) are also used for validation. These datasets were less noisy. The predictions from the model are used for the next frame prediction. The results from the next frame prediction strongly suggests that the mask prediction from our image segmentation model was very good. (Dice scores cannot be given since the original polygons are not available for extraction).

**Performing features**

The image pixel is an important feature which really affects the model. Since some images had mask falling outside the real land boundary in the image, we tried manually drawing some of the polygons. After using these polygons for training, the pixels exceeded the maximum and the model was adversely affected. The graph below depicts the results.



WrongPixels

Graphical user interface, chart, histogram

Description automatically generated

Right Pixels

**Evolution of our approach**

Datasets were the main headache. We used all the datasets (including noisy ones) for augmentation at first, which produced very bad predictions. We reduced it further to these no or less noisy data and realized the model gave better predictions.

Again, the training loss to evaluation loss in the first scenario (where all data sets were used) was worse than the latter one (where carefully selected data sets were used). The model might look outfitted from the graph above but those were the best results we had from the data, and the predictions compared to the original masks are good.

**Unique idea about approach**

With this architecture, a usual contracting network with pooling operators are replaced with up-sampling operators. These layers increase the resolution of the output. As a result, very few training images yield more precise segmentations.

Also, with ArcGIS Pro, you get to see polygons graphically, which helps to avoid polygons that lie in black areas of the linescans. It helped the model really distinguish black areas meddling with fire.

**Limitations**

In some cases, prediction is not accurate as needed. Only the boundaries of the fire area are predicted and therefore we had we fill the inner areas with white denoting fire. This was the case for some of the test images.

Also, smoke around fire is being segmented as fire by the model some times.

At times, some unusual surfaces are being detected as fire and therefore were avoided.

**Again, if images fed to the model are not of the same pixel as the satellite images given to us for the competition, then the output prediction will not be correct. So, if the same satellite images are taken outside Australia with the same pixels, then yes, the model can work outside Australia. The pixel of the image is the main factor here, not the location.**

**Optional Requirements**

One element we would like to highlight is if polygons within black areas of linescans are drawn as not fire areas, then the model can learn well to produce better results. With more time, we think we could have done that. We failed on our first attempt, but we believe we can get it with time. Also, creating more augmented images for training created a bad predictions, so we used few augmented images which produced better results. Also, there were some predictions that had noise in addition to the right prediction so we got rid of them by redrawing the prediction. Also, for the Macalister91 event, we realized the boundary from the prediction, and redrew the prediction again with the right demarcations. We compared this model’s prediction with the other checkpoint’s predictions. This helped to arrive at the best prediction for images where the fully mask was not predicted.

One factor that caused a breakthrough was using the carefully selected images for augmentation instead of all the data.

Manual extraction of polygons using ArcGIS Pro really took time (roughly 22 days). Also, discrepancies of ids of polygons caused a lot of confusion during extraction. We had to start the process again, since we missed some images.

For prediction time, a single 512x512 image takes less than 20s. For a maximum prediction of 50 images at a go, it takes less than 8mins.