So, the first model that I created was a retina net and was based on a resnet-50 CNN model. On my testing data of monuments, the final trained model achieved about 95 % accuracy and I had trained it through google Collaboratory after 450 epochs. While the task only required classification my model also performs object detection with a bounding box along with the assigned task of object recognition. I added this feature because you are an organisation which works in displaying maps and I figured that a bounding box would be a lot more useful than a simple probability score. I used a retina net because It has achieved higher accuracies then even two-staged architecture models and a higher efficiency than single-staged architecture models. The model is a single, unified network composed of a backbone network and two task-specific subnetworks. The backbone is responsible for computing a conv feature map over an entire input image. The first subnet performs classification on the backbones output; the second subnet performs convolution bounding box regression. The backbone is also masked with a feature pyramid network which extracts the final useful features from the underlying convolutional feature network.

While attempting that task, I faced problems trying to find a suitable dataset to train my model and ended up creating my own dataset and my own bonding boxes in about 100 images of which 20 were used for testing. Since, my dataset was small, I had to train it on a pretrained model that I found elsewhere.

I think that in the end the results were satisfactory and up to expectations. When I tried the model, the lower layers correctly extracted basic features while the upper layers were responsible for more complex features. However, I also believe that had I trained the model on an even higher number of epochs then I could have achieved higher accuracy as when I reviewed the per epoch progress report, accuracy was slowly still increasing. However, as google Collaboratory has a 12 hour limit, I couldn't do so.

My second model was to classify level of damage in forests due to fires. This was a complicated task as I didn't have to recognise an object in the image, rather I had to classify the type of colour scores that the image had. To attempt this with a CNN was tough as CNNs usually look for patterns and lines; however, I undertook this challenge.

The final model was trained on 500 epochs using google Collaboratory on the dataset provided. Since RGB values were to be the main filter, the model is small and simple. It starts off by flattening the three RGB values into one as to get the colour of pixels.

After this it is passed through the following three convolutional layers:

1. Filter size – 3*3; number of filter-32
2. Filter size – 3*3; number of filter-32
3. Filter size – 3*3; number of filter-64

Finally, the resulting layer was flattened again and passed through a fully connected layer and a sigmoid layer to reduce the final values to between 0 and 1 which resulted in three scores for the three categories

So, this was the model I designed and it achieved about 50% accuracy on my testing dataset. This was rather disappointing and I attribute the following reasons for this

1. Classifying a forest as burnt, half burnt and green is a rather subjective topic. For example- even my human eye could often fail at recognising a forest that is burnt and one that is not.

2. The training dataset was rather small. Perhaps, a larger dataset might have resulted in higher accuracy.

In hindsight, I think that instead of a CNN had I converted the image to an array of RGB values and then run a simple classifier on it; I might have achieved better results. Even still, when I manually tested the model, its inaccuracies looked like valid mistakes that any human might also make.

Overall, I am somewhat satisfied with both of these models and am glad that google Collaboratory allowed me to swiftly train these models in spite of me having a CPU-only PC