Early Fire Detection using Image Segmentation

Business Objective

Fire is one of the deadliest risks any person can encounter in his life. In a concise time frame, fire can wreck an area, say in a forest, hut, house, building, etc. Fire can lead to the loss of expensive property, and in the worst possible scenario, it can also lead to the loss of human life. Detecting fire thus becomes a significant concern.

Our project will carry out fire detection by adopting an RGB (Red, Green, Blue) model based on chromatic and disorder measurement for extracting fire pixels and smoke pixels. Our main aim is to locate the position where the fire is present, which will help the authorities take proper measures to avoid any loss. To prevent this, we will build an early fire detection using image segmentation with the help of the mrcnn model.

Image Localization helps to detect the location of a single object in any given image. In our case, image localization can locate the fire in a given image.
We will be using image segmentation, in which we group a similar set of pixels, i.e., divide the image into segments and thus make use of the essential segments. Hence image segmentation is used in this project as it gives us the desired location of our object in the image.

Further, we will use the Mask RCNN model to train and build predictions over our input images. Mask RCNN is a deep neural network algorithm that is used for solving segmentation problems.

This project aims to build a deep neural network model that will give the best accuracy in detecting fire in an image.

Data Description

- In this case, we use 20 fire images for the training set and ten fire images for the validation set, and one image for test data. These images can be JPG, PNG, TIF formats.
- Via_project.json is the annotation file containing the region of interest marked.

Aim

This project aims to make predictions over images and detect mask and bounding boxes around the area of interest, i.e., mask and bounding around the fire in a given image.

Tech stack

- Language - Python
- Libraries – numpy, keras, tensorflow, mrcnn, scikit-image, etc (as per mentioned in the requirements.txt file)

Approach

1. Install the requirements.txt file.
2. Import the required libraries.
3. Using the VGG Annotator, create annotations and save them in the JSON files. Then creating a class to add these annotations and use them. (Link for the VGG Annotator - https://www.robots.ox.ac.uk/~vgg/software/via/via_demo.html)
4. Define LoadDatatset class for loading of image data and annotations for train and inference data.
5. The setting of the root directory for the project and the path of the weights file.
6. Define config class for setting configuration for training on the dataset.
7. Define the LoadBackBone for train and inference data class, for loading the backbone pre-trained weights to extract features.
8. Model Building (Using Mask RCNN)
    - Overtraining data images
    - Over validation data images.
9. Predict the mask and then plot it over the test data images.

Modular code overview

```
input
  |_train(images + via_project.json)
  |_test
  |_val(images + via_project.json)
  |_mask_rcnn_coco.h5


src
  |_Engine.py
  |_ML_Pipeline
            |_InferenceConfig.py
            |_InferenceModel.py
            |_LoadBackbone.py
            |_LoadDataset.py
            |_References.py
            |_TrainConfig.py
            |_TrainModel.py
            |_VisualizeMask.py


lib
  |_FireMask.ipynb


output
  |_logs
```

Once you unzip the modular_code.zip file, you can find the following folders within it.

1. input
2. src
3. output
4. lib

1. input folder - It contains all the data that we have for analysis. These are three subfolders present in the dataset folder. These folders contain images along with the via_project.json files. Also, a rcnn coco.h5 file is present.

   Train (20 images)

   Test (1 image)

   Val (10 images)

2. src folder - This is the most important folder of the project. This folder contains all the modularized code for all the above steps in a modularized manner. This folder consists of:
   - Engine.py

- ML_pipeline
  The ML_pipeline is a folder that contains all the functions put into different python files, which are appropriately named. These python functions are then called inside the Engine.py file.

3. output folder **–** The output folder contains the logs generated after training the mrcnn model. A training model is saved per epoch in h5 format. The model is trained on 20 images and a h5 file is been generated.
   This model can be quickly loaded and used for future use and the user need by passing different set of images.
   Note: This model is built over a chunk of data. One can obtain the model for the entire data by running Engine.py by taking the whole data to train the models.

4. lib folder **-** This is a reference folder. It contains the original ipython notebook that we saw in the videos.

Project Takeaways

1. Understand the business problem.
2. Understand the concepts of image detection, image localization, and image segmentation.
3. Understand the concepts behind Region Proposal Network (RPN), ROI Classifier, and bounding box Regressor.
4. Understand the idea behind transfer learning and how it is different from machine learning.
5. Understand the concept of Backbone and the role of the backbone (restnet101) in the Mask RCNN model.
6. Understand what MS COCO is and how to load the pre-trained COCO as weights.
7. Performing image annotation using VGG Annotator.
8. Import the image data and required libraries.
9. Define and understanding a class for configuration for training on the dataset.
10. Learn how to add and use annotated images.
11. Fitting Mask RCNN modelling over train and validation images along with annotation files.
12. Understand how to create and store the log files per epoch.
13. Defining of Inference configuration and model
14. Making predictions over the test data (Inference)
15. Using matplotlib.image to visualize our results, i.e., the bounding and masking over the inference image.