Fire Detection Using Tensorflow 2 Object Detection API

1. Downloads and Installation
    1. Install Tensorflow & Clone Object Detection API
    2. Download fire dataset ,Clone scripts from my github ,efficientdet_d0 weights from TensorFlow Repo
2. Training
    1. Generate tfrecord to train
    2. Training
    3. Exporting the trained model
3. Inference
    1. loading detection function
    2. Inference

References

Dataset : https://github.com/OlafenwaMoses/FireNET/releases/download/v1.0/fire-dataset.zip
Code Docs : https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/

Training and inference for this report was done in Google Colab.

Description

First we download and install Tensorflow 2.2 and clone the Object detection API from github. Then download the FireNET dataset containing the annotated images and XML files.

Tensorflow provides many model for detection, and i have chosen efficientdet_d0 to train as the final model size is small and easy to share the final trained weights (easy downloads,uploads).So we download the weights of efficientdet_d0. Along with this we need few files from my github repo, these are

- generate_tfrecord.py – Convert XML to TF Record.(From Docs)
- labelmap.pbtxt – Contains label FIRE
- pipeline.config – Contains structure for training such as No.of classes,learning rate,batch size...,etc

Now we generate the Tfrecords from the XML file and start training. Documentation suggests to reach a loss of 1 before termination but as the dataset is quite small, loss starts at 0.6 and plateaus at 0.2 after 6300 steps. Now after terminating the training we export the model which is used for inference.

We define a detection function that loads the model into memory .This trained model weights are also in my Github repo along with the test images . To this function we provide a list of paths to the images we would like to infer and then visualize the output.

Output

The inference part of the code loads the weights, and if the detection confidence is greater than 0.5 draws a bounding box. A statement will be printed saying ‘Fire’ or ‘No Fire’ which can be used as a trigger for the alerting system.

Model pruning and quantization can further increse the inference speed.


```
Step 100 per-step time 1.135s loss=0.559
Step 200 per-step time 1.135s loss=0.476
Step 300 per-step time 1.152s loss=0.520
Step 400 per-step time 1.162s loss=0.403
Step 500 per-step time 1.134s loss=0.373
Step 600 per-step time 1.199s loss=0.556
Step 700 per-step time 1.182s loss=0.420
Step 800 per-step time 1.149s loss=0.388
Step 900 per-step time 1.123s loss=0.540
Step 1000 per-step time 1.122s loss=0.473
Step 1100 per-step time 1.139s loss=0.342
Step 1200 per-step time 1.128s loss=0.397
Step 1300 per-step time 1.164s loss=0.352
Step 1400 per-step time 1.125s loss=0.308
Step 1500 per-step time 1.132s loss=0.426
Step 1600 per-step time 1.068s loss=0.273
```