Deep Learning

Final Project Report

Team Members:

Anjani Sai Mahesh Peddikuppa

Sree Sesha Sai Bharathi Kanteti

1.Introduction :

The project is based on Augmented Reality with a combination of Object Detection. In this project, the images of cats and dogs are identified by training the model with cats and dogs Dataset.The cats and dogs dataset consists of images of cats and dogs which are collected from kaggle and pexel websites and other internet sources. In our output video cats and dogs are detected through our deployed tensorflow lite android mobile application and tested live.

2. Project Description

In this project, we have used the ssd_mobilenet_v2_fpnlite model for training the dataset, evaluated it and tested it with a sample image from google. We converted the trained model into tensorflow-lite model.The tflite model is then imported into Android studio after making the necessary changes to the app, it is deployed into an android mobile. This model is used to identify the objects(cats and dogs) in the images with bounding boxes and its precision.

3.Purpose of the Project:

The main goal of this project is to put what we learned in class into practice. This project allowed us to explore various features of the tensorflow packages and its pre-trained models related to object detection. Also made us learn about the project deployment in Android mobile as an application using android studio.

4.Source of Data:

The source of data is from kaggle yolo-animal-detection-small and Pexels website and splitting the collected data into training data and validation folders.

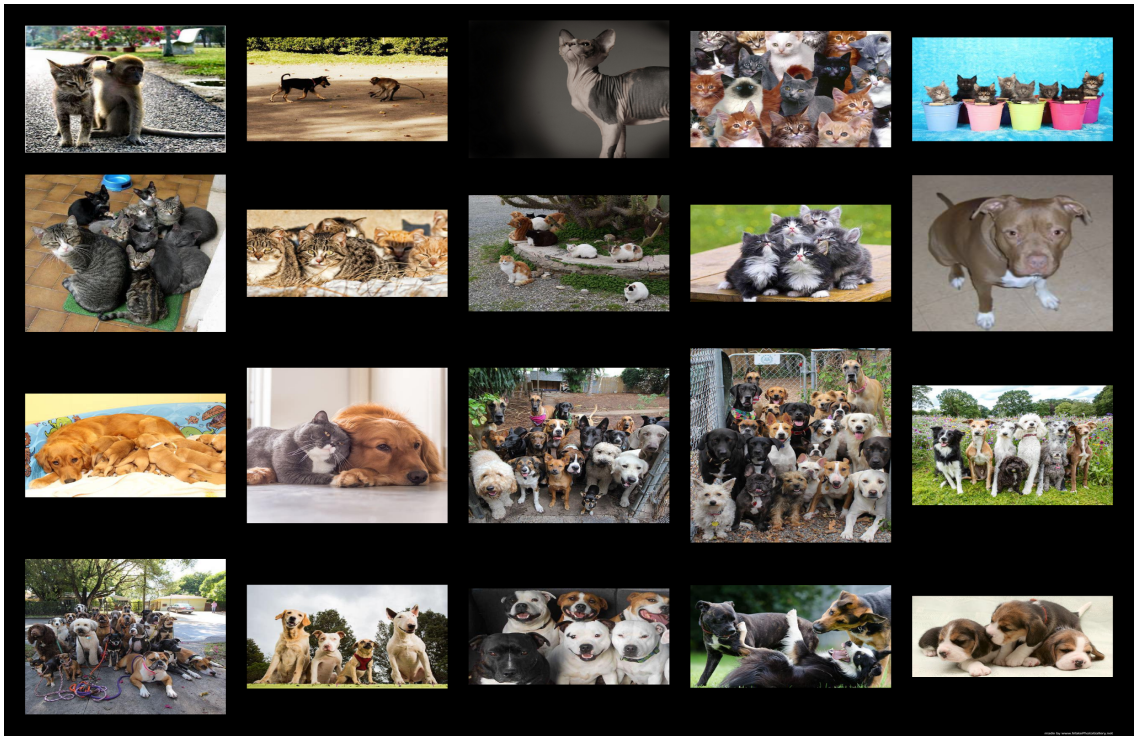


Figure illustrates the Complex Images

5. Proposed Method

The proposed method of our project is First, the images are splitted into training and validation folders and then converted to tf records. Used these tf records to train the model using the object detection API and the pre-trained model named SSD Mobilenet fpn lite. After training, we load the trained model from a checkpoint and then test the model with sample images. The test results will be the object names detected in the images with bounding boxes and their accuracy. This model is then converted into a tflite model which is used for the deployment of the application in android mobile phone.

6.Implementation Steps :

- **Acquiring data and Labeling images using LabelImg.**

  Collected images from kaggel yolo-animal-detection -small(as recommended), pexels website and from google images. Created folders Train and Validation and divided the images into the folders in 1:5 ratio respectively.

  Cloned the labelmg repository from a source in github which is mentioned in the references, downloaded and installed Anaconda (Python 3+).

  Opened the Anaconda Prompt and went to the labelImg directory and execute the below commands and the labelimg application gets automatically opened

  conda install pyqt=5

conda install -c anaconda lxml

pyrcc5 -o libs/resources.py resources.qrc
python labelImg.py

Opened the directory where the images are located and created bounding boxes on images, labeled them as cat/dog and saved it. This generates an xml of the image and is stored in the respective directory and is further used for creation of TFrecords.

- **Create a Virtual Environment**

  Create a virtual environment to avoid dependency conflicts so that there will be an isolated python virtual environment and activate it. Install dependencies and add a virtual environment to the python kernel.

- **Created a folder structure as follows,**

  Under tensorflow folder there are four sub folders :
  Models - cloned tensorflow model into this folder
  Workspace - Contains images folder, annotations folder, pretrained models, and models.
  A. Images : contains Test and Train images folders
  B. Annotations: Contain xmls folder, .tfrecord files, and labelmap.txt file
  C. Pretrained models: contain the cloned repositories of pre trained model (SSD MobileNet 320*320)
  D. Models: Contain another folder with pre trained model name which contains checkpoint files which are generated after training, and the pipeline.config file
  Protoc - contain the google b=protocol buffers script
  Scripts - contains generate_tfrecord script

  Inside the Tensorflow\workspace folder created another folder named images
  which contained train and test images folders with its xmls.
  Also created another folder named annotations under Tensorflow\workspace folder and placed an xml folder in it which only has xml files.
  Created a labelmap.pbtxt file which contain the class names and IDs

- Download pretrained model, Protobuf repositories

Cloned pretrained model to the pre-trained model folder which is under tensorflow\workspace folder. Modified the pre-trained model's configuration file based on our data set. Changes made are changing the num of classes to 2 from 90 as we have only 2 classes namely cat and dog. Given labelmap path,finetune checkpoint type and finetune checkpoint,train and validation tfrecord path, batch size. And moved the file to Tensorflow\workspace\models\my_ssd_mobnet folder

Cloned protobuf(protocol buffer) github repository in order to configure models and their training parameters. And set the protobuf path in environmental variables.

Move to cd F:\dlfinal\Tensorflow\models\research and the below commands

$ protoc object_detection/protos/*.proto --python_out=.

$ set PYTHONPATH=$PYTHONPATH:`pwd`:`pwd`/slim

pip install cython
pip install git+https://github.com/philferriere/cocoapi.git#*subdirectory=PythonAPI*

- **Tensorflow model**

  Copy the setup file present in object detection/packages/tf2/setup.py to the research folder and run

  python -m pip install .

  This will install the setup file of object detection api.((Install Tensorflow object detection)

- **Verify the installations**

  Go back to **F:\dlfinal\Tensorflow\models\research** path and run to test installations.
  python object_detection/builders/model_builder_tf2_test.py

  If all the installations are working fine. We proceed to further steps

  I faced the below errors while performing above installation steps.

Errors :

1. Failed Building wheels for apache-beam

Solution : set a proper pythonpath

protoc object_detection/protos/*.proto --python_out=.

Pip install wheel

Pip install apache-beam

2. Failed Building wheel for pycocotools, could not build wheels for pycocotools

Solution : Installing Visual Studio with c++ distributions solved this issue.

3. Failed building wheels for opencv-python which use PEP 517 and cannot be installed directly.

Solution : By performing

pip install --upgrade pip setuptools wheel
pip install opencv-python solved the issue

4. Module not found

Solution : imported the respective modules as mentioned in the error message.

5. For all the below errors:
   - Flash=tf.apps.flags AttributeError: module ‘tensorflow’ has no attribute app
   - Ssdmobilenet_v2 is not supported for tf version 2. See ‘model_builder.py’ for the features extractors compatible with different versions of tensorflow

- Import input_data ModuleNotFoundError: No module named input data.
- Optimizer =tf_opt.MovingAverageOptimizer(NameError: name ‘tf_opt’ is notdefined,model_main.py:AttributeError:‘WeightSharedConvolutionBoxPredition’
- object has no attribute ‘inputs’
- Object detection:AttributeError:module ‘tensorflow’ object has no attribute ‘init_scope’
- module ‘tensorflow’ object has no attribute ‘contrib’

Solution : The tensorflow object detection model was not compatible with the pretrained model that i used, when cloned the latest version of tensorflow model, all the below issues got rectified.

Import object detection library

- **Create TF records(.TFrecord files)**

Converted the data set into csv using the code xml_to_csv.py and then converted the csv file to .tfrecord files using generate_tfrecord.py script and labelmap.pbtxt file which has the classes information with respective id.

Errors:

6. Tensorflow.python.framework.errors_impl.NotFoundError: Failed to create a NewWritableFile: System cannot find the path specified. No such process.

Solution : The path of xml was incorrect, when I corrected the path, it worked. And tfrecords were generated successfully.

- **Train the model.**

Started training the model by running the following command:
python Tensorflow\models\research\object_detection\model_main_tf2.py
--model_dir=Tensorflow\workspace\models\my_ssd_mobnet
--pipeline_config_path=Tensorflow\workspace\models\my_ssd_mobnet\pipeline.config
--num_train_steps=20000

Which means execute model_main_tf2.py which is present in Tensorflow\models\research\object_detection\ path,
the model directory would be Tensorflow\workspace\models\my_ssd_mobnet and the configuration file can be taken from Tensorflow\workspace\models\my_ssd_mobnet\pipeline.config and train the model for 20000 steps.
After the successful training checkpoint files will be created in tensorflow/workspace/model/my_ssd_mobnet folder.A train folder with V2 file will be created under tensorflow/workspace/model/my_ssd_mobnet/train.

7. Could not load dynamic library cudart64_110.dll Dlerror: cudart64_110.dll not found

Solution : installed the version of cuda and cuda cnn which are compatible with the tensorflow version.https://www.tensorflow.org/install/source_windows

- **Evaluate the model.**

Evaluated the model using
Python Tensorflow\models\research\object_detection\model_main_tf2.py
--model_dir=Tensorflow\workspace\models\my_ssd_mobnet
--pipeline_config_path=Tensorflow\workspace\models\my_ssd_mobnet\pipeline.config
--checkpoint_dir=Tensorflow\workspace\models\my_ssd_mobnet

Which means execute model_main_tf2.py which is present in
Tensorflow\models\research\object_detection path, the model directory would be
Tensorflow\workspace\models\my_ssd_mobnet and the configuration file can be taken from
Tensorflow\workspace\models\my_ssd_mobnet\pipeline.config and the checkpoint directory will be in
Tensorflow\workspace\models\my_ssd_mobnet

After successful evaluation a folder named eval will be created in
tensorflow/workspace/model/my_ssd_mobnet.

- **Detect Objects in images**

After successful training and evaluation, we have taken the trained model form checkpoint directory which is ckpt-21 to detect the objects in images.

- **Freeze and exporting the model and convert the model into TFLite**

Using the following command we freezed the graph
#Freezing graph for tflite scriptpath
python tensorflow\models\research\object_detection\export_tflite_graph_tf2.py
--pipeline_config_path=tensorflow\workspace\models\my_ssd_mobnet_new\pipeline.config
--trained_checkpoint_dir=tensorflow\workspace\models\my_ssd_mobnet--output_directory=tensorflow\workspace\models\my_ssd_mobnet\tfliteexport

Aboved command executes export_tflite_graph_tf2.py taking trained checkpoint folder path and config file and saves the saved_model.pb file, variables with a folder named saved_model in TFLITE path

Convert the model into .tflite file using bellow command
tflite_convert
--saved_model_dir=Tensorflow\workspace\models\my_ssd_mobnet\tfliteexport\saved_model
--output_file=Tensorflow\workspace\models\my_ssd_mobnet\tfliteexport\saved_model\detect.tflite
--input_shapes=1,300,300,3 --input_arrays=normalized_input_image_tensor
--output_arrays='TFLite_Detection_PostProcess','TFLite_Detection_PostProcess:1','TFLite_Detection_PostProcess:2','TFLite_Detection_PostProcess:3' --inference_type=FLOAT --allow_custom_ops

- **Download and Install Android Studio**

Building App using Android studio:
Cloned the Android mobile object detection code from

https://github.com/tensorflow/examples/tree/master/lite/examples/object_detection/android repository
Copy out tflite model and labelmap.txt file to the https://github.com/tensorflow/examples/tree/master/lite/examples/object_detection/android/app/src/main/assets folder of the cloned repository

- **Import code folder to Android Studio**

Download Android Studio and import the android folder of the tensorflow repository to android studio
Comment the apply from: “Download_model.gradle” and perform gradle sync.

- **Update Detector Activity File**

Change the .tflite file name and labelmap.txt in DetectorActivity.java file, and TF_OD_API_IS_QUANTIZED to fals

- **Sync the gardle and Run the application**

Pair and android device using pair device using wifi, scan the qr code and run the application in android studio
After successful execution, the app will be installed in the paired mobile phone and we can detect images via app tflite.

Errors Android studio

8. Runtime exception while loading android app:

   Solution: there was no metadata found in my detect.tflite file. After creating the meta data using https://www.tensorflow.org/lite/convert/metadata_writer_tutorial , I was able to fix the issue and run the app successfully.

9. Manifest merger failed : android:exported needs to be explicitly specified for element <activity#org.tensorflow.lite.examples.detection.detectoractivity>. apps targeting android 12 and higher are required to specify an explicit value for android:exported` when the corresponding component has an intent filter defined. see https://developer.android.com/guide/topics/manifest/activity-element#exported for details.

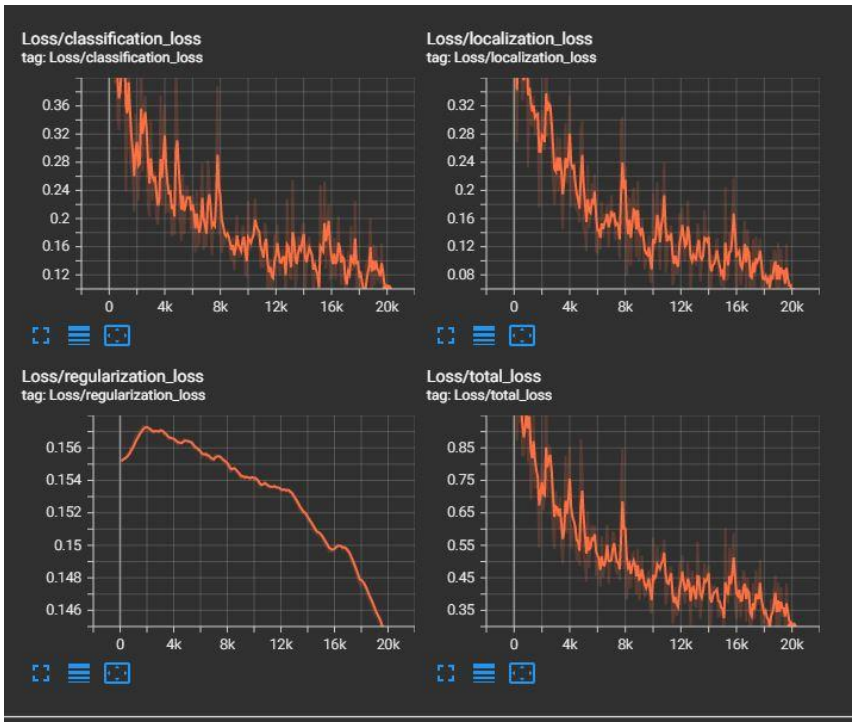
Solution : this error occurred in android studio when updating the SDK version to 32 but when I went back to SDK version 30 it worked.

7. Programs with Documentation

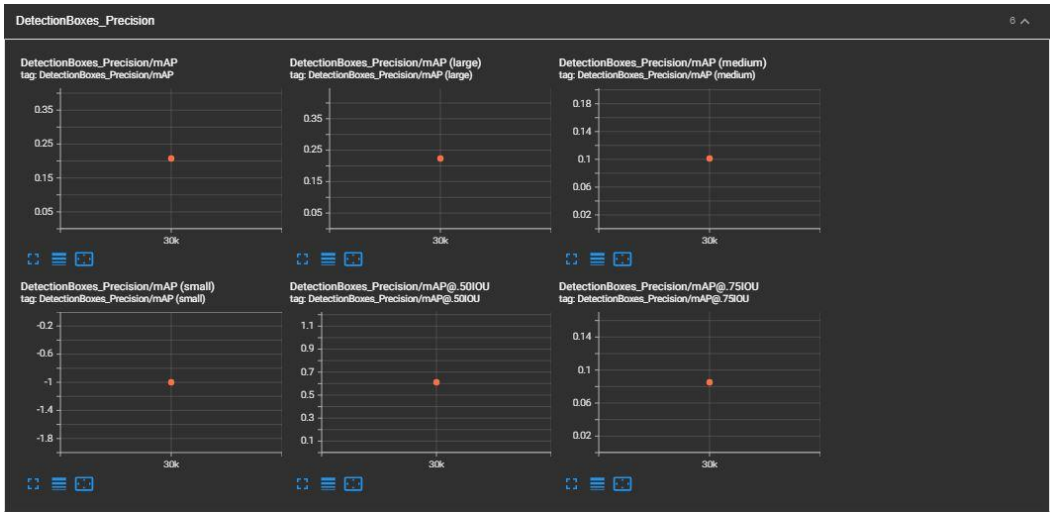
Attached in isidore.

8. Results:

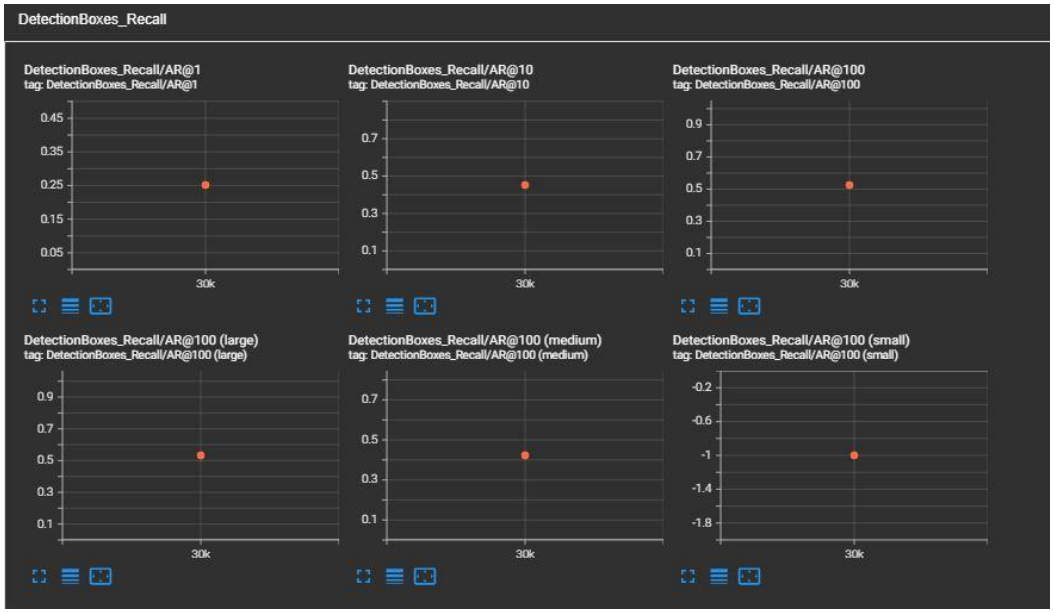
Loss Graph



Precision:



Recall:



9. Conclusion

In conclusion, this project detects the cats and dogs objects in images through bounding boxes with its accuracy. This is achieved using Tensorflow Object Detection API and a pretrained model Ssd MobileNet v2 fpnlite 320x320 . Which is then converted into a tflite model, and deployed as an android mobile application.

Reference Links:

https://towardsdatascience.com/custom-object-detection-using-tensorflow-from-scratch-e61da2e10087

https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/training.html

https://github.com/nicknochnack/TFODCourse