CS492 Systems for Machine Learning
Project 1 Report

Duman Kuandyk 20160857
Mirali Ahmadli 20170847

Introduction

In the first project, we had to implement the inference framework on Tensorflow based on the YOLO-v2-Tiny model. Using the pre-trained model weights, we had to implement an object detection model based on the input video so that the output video would have labels on the corresponding objects.

Implementation

We have started the implementation by loading the pickle file and performing initial data exploration to better understand the model. The model was missing a graph, and to build the graph, we visualized the model. For this purpose, we have downloaded a model from ONNX Model Zoo and visualized it through Netron. The model visualization helped us to build the graph and learn more about its hyperparameters.

The graph building part was implemented with the help of TensorFlow, mainly with `tf.nn.conv2d()`, `tf.nn.batch_normalization()`, `tf.nn.leaky_relu()` and `tf.nn.max_pool2d()` functions.

The function to read and write video files in the `__init__.py` file was implemented with the help of OpenCV documentation. The input video was processed through `cv2.VideoCapture()` function, while the output video was processed by `cv2.VideoWriter()` function. We have also resized and restored the video input to ensure the proper processing along the inference process. The main action was defined in the `video_object_detection()` function, which reads and resizes the input, does the inference, recovers the initial video input size, and then puts the coordinates of the bounding boxes, and then saves the video file. The function also measures total elapsed time to run inference. The model performance, measured in the FPS processed per second, was implemented outside of the function in the `__init__.py` file.

The initial implementation was overhauled several times to ensure better performance and bug fixes, which included black screen output, random bounding box assignments, and others.

Results

Type	FPS processed per second	Total elapsed time, sec
GPU	3.804288357580815	8.306427478790283
CPU	1.1394462496996516	26.32857847213745

Comparison between GPU and CPU

However, the speed of GPU and CPU is highly dependent on their models, and after running our code using CPU and GPU for several times, we observed that GPU performs ~3 times faster than CPU. We measure the end-to-end time and time spent only for inferencing which runs Tensorflow session with a given graph (model) and input frame. The graph consists of tensors (matrices) and it is known that GPUs are much more efficient for matrix calculations and convolutions. This is mainly because of the GPU being bandwidth-optimized in comparison to CPU, which optimized for latency. GPU is able to process a larger amount of memory at a time in comparison to CPU, and while CPU does the processing faster, the amount factor makes GPU process faster in the long-term, if we consider the entire inference process.