# Korea Advanced Institute of Science and Technology
# CS492 Spring 2021
# Project 1

Jinwoo Hwang
jwhwang@casys.kaist.ac.kr

Due: Apr. 02 11:59 PM KST

## Rules

- Do not copy from others. You will get huge penalty according to the University policy. Sharing of code between students is considered as cheating.
- Every code was written and will be tested in the provided docker environment. You need to implement your code in the given environment.
- Read carefully not only this document but the comment in the skeleton code.
- This is the version 1.0 document, which means not perfect. Any specifications or errors can be changed during the project period with announcement. Stay tuned to Piazza.

## Introduction

This project is made to study the basic understanding of ML inference using high-level framework, Tensorflow, with graph construction. Your job is to implement an inference code dectecting objects with YOLO v2 tiny algorithm using the framework. A sample video is given, you can check your implementation using the video as input. Or you can take your own video to test your code.

## Project Sturcture

This project contains the following files:

- `__init__.py`: Main source file. Following command will produce a labeled video:

  ```
  $ python3 __init__.py [path_to_input] [path_to_output]
  ```

- `yolov2tiny.py`: Source file of `YOLO_V2_TINY` class.
- `activation.py`: Supplementary file implementing activation functions.
- `sample.mp4`: Sample input video for the inference job.

## Run Docker

Docker is a widely deployed virtualization technique that is mainly used to distribute the same environment to users. For instance, a program may not be dependent on a specific version of libraries and output of the program may vary between the different versions of Linux or libraries. However, enusuring every single libraries with a specific version is a troublesome and difficult job, prone to making mistakes. Using docker, we can sync the environment easily.
Install `docker`, refer to this link.
Considering the network policy of the KCloud machines you will be using, we will be providing you a pre-built image uploaded on `DockerHub`. The following code will pull the docker image.

```
$ docker pull jinwooh/cs492:tensorflow-gpu
```

Once you have completed downloading the image, launch a container and name it `cs492-gpu` with following command.

```
$ docker run -it -v $HOME:/home/st/ \
    --name cs492-gpu jinwooh/cs492:tensorflow-gpu bash
```

Press `Ctrl+p, Ctrl+q` to detach from your docker container and go back to host machine. If you want to re-attach to the docker, use this command.

```
$ docker attach cs492-gpu
```

If you have accidently exited the container, you have to start the stopped container again. Please refer to the official document for more informations.

## Implementation

### Read and Write Videos (5pt)

The function `open_video_with_opencv` in `__init__.py` takes two pathes to videos from command-line arguments and opens them. Your first job is to write a code to create objects for each videos with an image library, `opencv`. You will be able to read and write image frames through these objects. Open the given input and output video file using `cv2.VideoCapture` and `cv2.VideoWriter` respectively and return them. You may need the size of the input video or something else; return whatever you want for future computation.

### Tensorflow Graph Construction (20pt)

Look carefully at the `YOLO_V2_TINY` class in `yolov2tiny.py`. It has a constuctor, a function that builds a tensor graph, and a function that initiates the inference. Your job is to implement `build_graph` with tensorflow library which builds a **tensor graph**. The graph will be consist of nodes, called **tensor**, which represent internal computations in the inference job. You need to add the tensors according to the YOLO v2 tiny algorithm model. Specifically, return the starting tensor of graph and the list of all tensors to compute intermediate values. By the way, they will be evaluated not on this construction but on the running a session at the `inference`. Check how the return values are used.
Tensorflow API provides basic operations for machine learning computation such as convolution or adding a bias. Refer to this API website. `tf.maximum, tf.nn` will come in handy. Furthermore, use $10^{-5}$ for $\epsilon$ in batch normalization layers.

### Inference (10pt)

Now you are ready to process the video in `video_object_detection`. Before starting the inferecing, make a `YOLO_V2_TINY` object with proper argument to construct a graph for inference. Take frames from opened video and do inference job with the graph you've implemented. Your code must include following jobs for each frame.

- Do the `inference`,
- Run `postprocessing` in `yolov2tiny.py` using the inference result, accumulate them through the video writer object. Note that coordinates from `postprocessing` are calculated according to resized input; you must adjust them to fit into the original video.
- Measure the time only for inference. Also, measure the inference time including preprocessing and postprocessing. `time` library will be helpful.

Note that the raw input must be adjusted to fit into the algorithm, including resizing the frame and changing the dimension.

Plus, save the intermediate values of the *first frame* in the `intermediate` directory. You may assume the directory already exists. `np.save` will help you. The full path of the files must be *intermediate/layer_i.npy* where $i$ is the layer number. It will be graded by comparing them to the output from refernce code.

### Report (5pt)

Write a report of one or two pages long. Your report must include 1. how you implemented, 2. execution time and how many FPS processed (end-to-end, only for inference), 3. comparison the execution time from CPU and GPU and analyze it. The purpose of the report is to show your understanding. Please write the answer short and clear.

## Handin

Your final outcome should be a single tar file that contains two python files, a report in pdf format and the directory with intermediate tensors. The name of the file should be `[student_number].tar`. Upload your tar file to KLMS.

## Environment Detail

- You must use the KCloud VPN to connect to the given server. Note that it is a different program from KVPN.
  - Windows Manual [link]
  - MacOS Manual [link]

  If you encounter a technical issue while accessing KCloud VPN, please contact kcloud@kaist.ac.kr.
- The servers handed out to you does not have any GPU installed, hence you will not be able to test the GPU version of your code. Also, when running the Tensorflow, you will encounter errors as following,

  ```
  Could not load dynamic library 'libcuda.so.1'; dlerror: libcuda.so.1:
      cannot open shared object file: No such file or directory;
  failed call to cuInit: UNKNOWN ERROR (303)
  ```

  This is due to the absence of the GPU, which you can safely ignore.

  In order to conduct a comparison with GPUs for your report, you will be time sharing 4 GPU servers by reserving a time slot on a Google Spreadsheet. There will be an additional announcement regarding the link to the spreadsheet.

- If you have any other question, please post them on class Piazza.
- In case you encounter a technical issue with the server that needs a reset, please contact your TA by e-mail.