Team Responsibility

Charlie Xu
Group leader, Algorithm Design

Terry Xie
Software Framework, Hardware Interface Design

Jenny Zeng
Deep Learning Algorithm Design

Chenghao Jiang
Hardware and Software Interface Design
Purpose

We aim to build a Deep Neural Network Image Recognition and Detection algorithm on a embedded GPU device. Attending 2018 DAC Contest. Sponsored and collaborated with Prof. Xie and SEAL LAB. Finally implemented on drones as well.
2018 DAC Contest

- Features embedded system implementation of neural network based object detection for drones
- Provided with hardware (Nvidia Jetson Tx2) and training dataset
- Evaluation
  - 20 FPS
  - Accuracy
  - Power consumption
Hardware - Nvidia Jetson TX2

- **Ubuntu 16.04 LTS**
- **Jetpack 3.0 SDK**
  - Deep Learning: TensorRT, cuDNN, NVIDIA DIGITS™ Workflow
  - Computer Vision: NVIDIA VisionWorks, OpenCV
  - GPU Compute: NVIDIA CUDA, CUDA Libraries
  - Multimedia: ISP Support, Camera imaging, Video CODEC
GPU

Streaming Multiprocessors (SM): 2

128 CUDA Cores/MP: 256 CUDA Cores

CUDA Capability Major/Minor version number: 6.2

Maximum number of threads per multiprocessor: 2048

Total amount of global memory: 7854 MBytes
Software - Algorithm

- Detect and tracks people and objects in video captured by drones.
- Problem with conventional tracking algorithm without deep learning...
- Deep learning - use training dataset to train computer to recognize people and be able to track them, even in difficult scenarios.
- We are implementing our design based on the state-of-the-art YOLO algorithm
Deep Learning: example - training

Training:
Trained from a large dataset consisting many pictures, specifying the object position and group

Object:
People

Coordinates:
Xmin = 100
Xmax = 130
Ymin = 70
Ymax = 120
Deep Learning: Example - Inferencing

Take a frame of video (picture) as input, inferenced by YOLO through layers to get a XML file output that draws the bounding box around the object. The pictures is not the same as in the training dataset, as long as objects are in same group (car), can detect.

Red is contest specified bounding box, green is what we detected.
Demo

*wrong
Algorithm: YOLO \textit{(You Only Look Once)}

- Object detection with a single Convolutional Neural Network (CNN)
Algorithm: YOLOv2

- Fast and Accurate
Training with YOLO

- Train for classification: Pretrain with ImageNet 1000-class dataset
- Train for detection: Convert the model to perform detection
Inference

- Speed: FPS (Frames per Second)
- Accuracy: IOU (Intersection over Union)

**Appendix | Intersection over Union (IoU)**

\[ \text{IoU}(\text{pred, truth}) = [0, 1] \]

\[ \text{IoU}(A,B) = \frac{|A \cap B|}{|A \cup B|} \]

Inferencing through one iteration
## Inference Results:

<table>
<thead>
<tr>
<th></th>
<th>Tiny YOLO</th>
<th>YOLO</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Accuracy</strong></td>
<td>31.5%</td>
<td>46.8%</td>
</tr>
<tr>
<td>(Average IOU)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>FPS</strong></td>
<td>21</td>
<td>9</td>
</tr>
<tr>
<td>(On Jetson TX2)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Bar chart showing accuracy distribution](image)
PARROT BEBOP 2

Camera: 14 mega-pixels with fish-eye lens

Video resolution: 1920 x 1080p (30 fps)

Battery life: 25 minutes flying time (with 2700 mAh battery)

GPS: Yes

Processor: Dual core processor with quad-core GPU

Storage: 8 GB flash storage system

Connectivity: Wi-Fi 802.11a/b/g/n/ac

Signal range: 300 m
ARDroneSDK3

Platform: IOS, Android, Linux

- Discover the drones on the network
- Connect the drones
- Send piloting and camera commands
- Configure the drones
- Get informations (depends on drones capabilities)
- Get H264 video stream on bebop
- Get MJpeg video stream on Jumping Sumo
- Transfer photos / videos
- Update the drones
- Handle Drone Academy / Mavlink files
Block Diagram

PARROT BEBOP 2 DRONE → Drone Controller

ARDroneSDK3
Nvidia Jetson TX2 Module With Trained Neural Networks

Wi-Fi Module

HMP Dual Denver + Quad ARM AS7 CPU

NVIDIA Pascal GPU
Trained Neural Networks

Monitor

Learning Algorithm (e.g. stochastic gradient descent)

Optional Image Processing (e.g. grayscale, edge detection)

Training Dataset (Images and locations and sizes of tacking boxes)

Neural Networks

Input Layer
The neurons at input layer should store the information of the input images. E.g. each neuron cab represent a pixel of the input image.

Hidden Layers
The number of hidden layers and the number of neurons in each layer vary, depending on specific design and requirement.

Output Layer
Neurons at the output layer should indicate the predicted location of tracked objects. E.g. the four neurons that are 1 can represent the tracking box's four corners, x_min, x_max, y_min and y_max.
Summary

- Received hardware Jetson TX2
- I/O specification set
- Working algorithm with both Pytorch and C
- Focusing on accuracy
- *Already submitted for alpha version, getting ready for 2nd submission*
Spring Quarter

Optimize YOLO - Improve fps and accuracy, limit power consumption

Update each month

See how we do on the contest

Final Goal: After we have our version of the algorithm, implement it on drones.
Collaborators / Mentors / Sponsor

Prof. Yuan Xie (UCSB SEAL LAB)
Prof. Yogananda Isukapalli
Dr. Lei Deng (UCSB SEAL LAB)
Yiming Gan (Master Student)
NVIDIA (SPONSOR)
DJI (SPONSOR)
Questions?