

HYU_Capstone

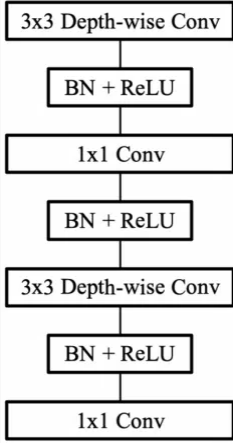
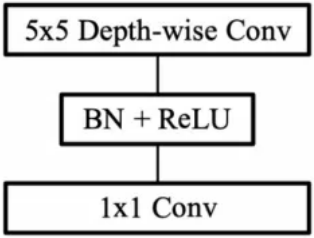
ZHANG HAOYU, student ID: 2018000337

Github Link: https://github.com/noahzhy/HYU_Capstone

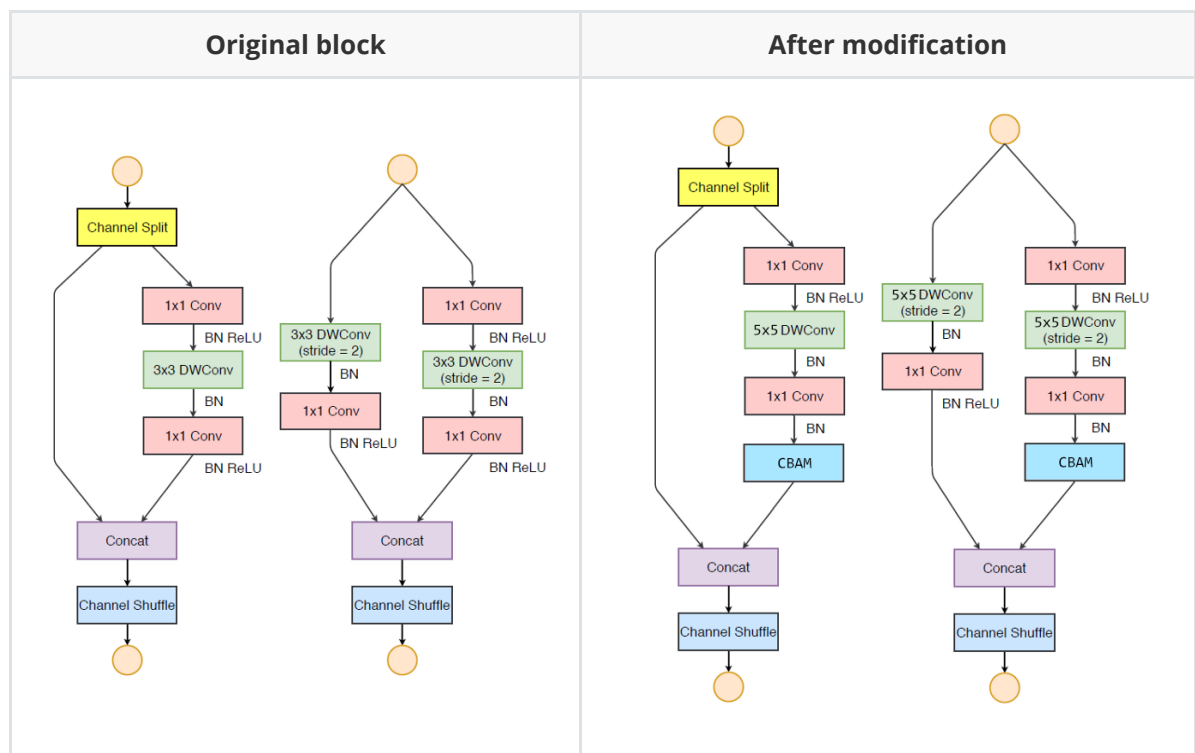
The goal of this project is to achieve the real-time (high fps) and better IDF1 score on MOT task, and try to deploy on the mobile devices.

In the actual project, it is required to deploy on the devices which the CPU-only environment. Therefore, I decided to design the lightweight real-time single stage multi-object tracking architecture.

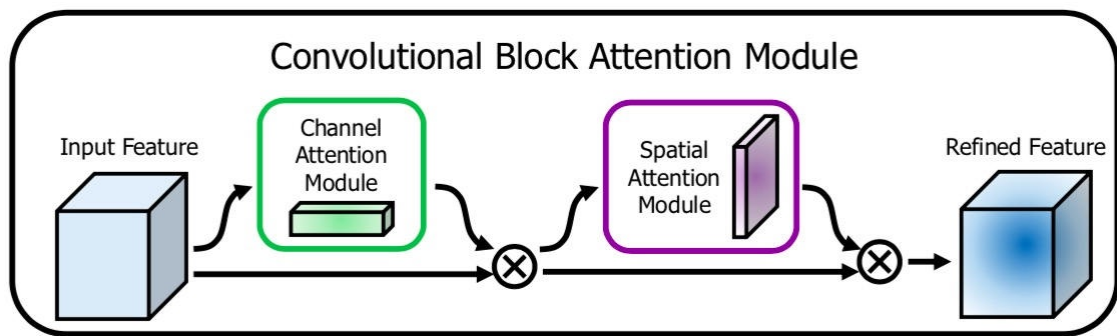
The first lightweight architecture that come to mind is shufflenet. It's every common lightweight architecture for mobile devices. Thus, I choose the shufflenet v2 1.5x as backbone because that it could achieve the high accuracy with less operations. In shufflenet, the 5x5 depth-wise convolution obtains twice the receptive field, with only a little increase in computation over the 3x3 depth-wise convolution. Therefore, I replace all the depth-wise convolutions in ShuffleNetV2 with 5x5 convolutions to get large receptive field.

	
Two 3x3 depth-wise conv with 32 channels	Single 5x5 depth-wise conv with 32 channels
$(3 \times 3 \times 32 + 32 \times 32) \times 2 = 2624$	$5 \times 5 \times 32 + 32 \times 32 = 1824$

In addition, I added CBAM module in the two basic unit of shufflenet blocks. The two basic unit of shufflenet after modification.



The detail of CBAM module is as following.

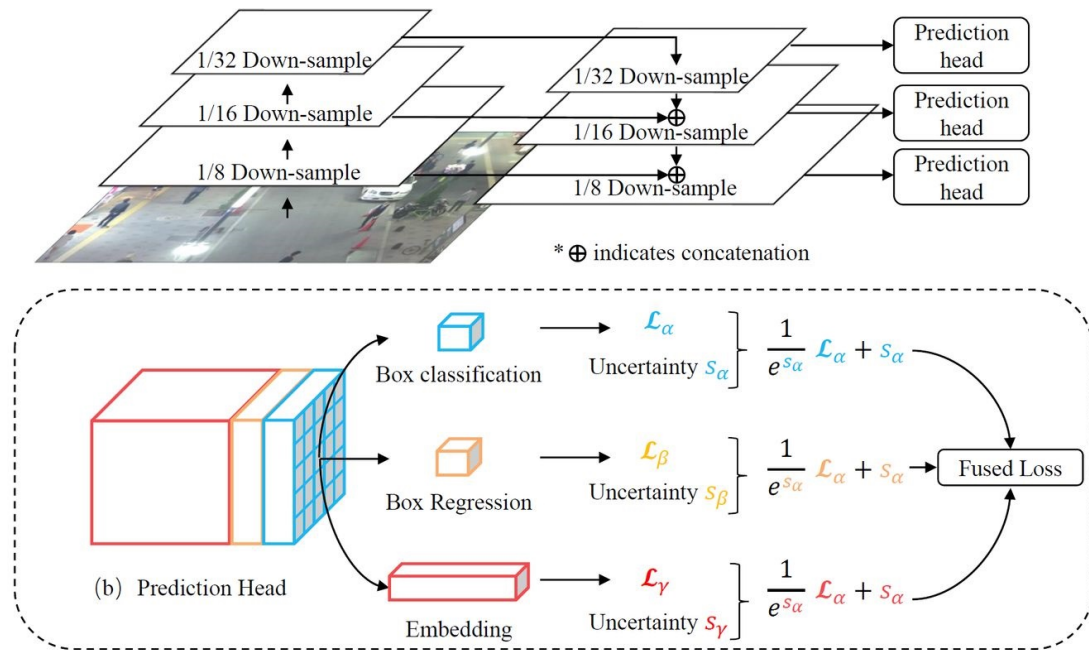


Comparing with original architecture with modified architecture.

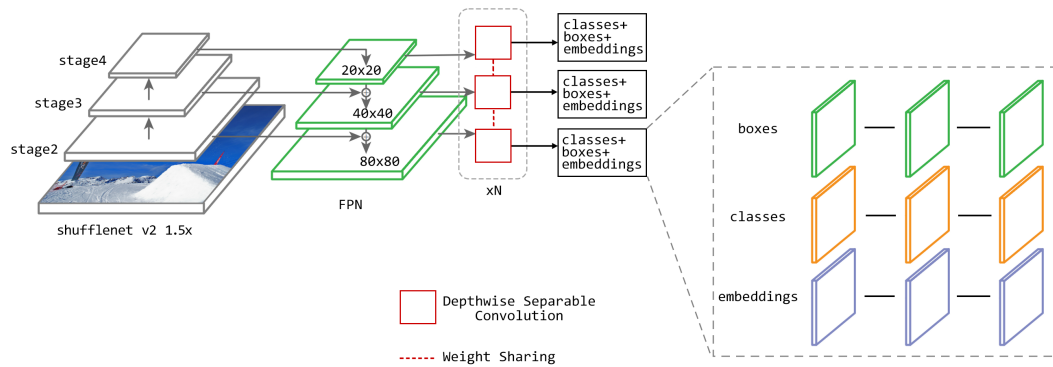
Layer	Output size	Output channels	After modification
Image input	640x640	3	3
Conv1, MaxPool	320x320, 160x160	24	24
Stage2	80x80	176	132
Stage3	40x40	352	264
Stage4	20x20	704	528

ShuffleTrack Architecture

The original JDE model is as following.



The shuffletrack architecture is as following.



Firstly, I constructed an object detection network using the modification of shufflenet v2 as backbone to extract feature, connect with the feature map of stage2, stage3, stage4 from backbone as FPN. In the JDE, embedding in prediction head with a 256 channels for ID tracking but 256 channel would causes overfitting. Therefore, I added a 128-dimensional branch of feature embeddings. In the JDE, each prediction head lacks of weight sharing. To deal with this problem, I add depthwise separable convolution blocks to share the parameters between different branch of FPN. Refer to the last layer of JDE model, the last full content layer is set to 547 to matching each track ID. The JDE extracts the embedding for each object, then using Kalman filter for tracking but shuffletrack

Part of loss function, I referred to SSD loss function which CE loss (Cross Entropy Loss) for the classification and embedding, smooth L1 loss for box regression.

Training

The model is training on the AWS GPU instance with one Tesla V100. I was tried to training on multiple GPUs, but it failed.

There are some changes in object detection parts, it need to be retrain the object detector. Thus I need to retrain the detector at first, then training the whole model. It needs more time. At the head of this document, I've included a link to my GitHub, and I'll update this document again when it's all trained.

Parameters and MAC

Architecture	Resolution	Parameters (M)	MAC (G)
ShuffleTrack(our)	640x640	2.55	13.21

Results (MOT17)

Model	MOTA	TP	FP	IDsw.	mAP	Inference time (ms/frame)
Tracktor	35.30	106006	15617	16652	36.17	45
Tracktor++	37.94	112801	15642	10370	36.17	2645
RetinaTrack	39.19	112025	11669	5712	38.24	70
ShuffleTrack(our)						20

Demo

Reference

- [ShuffleNetV2](#)
- [JDE](#)