ICIP 2020

ISSD Submission template

NOTE: This is a confidential document, ISSD will not share your information, nor commercialize it

Team name	BUET Endgame
Team members (firstname, surname)	• Sheikh Asif Imran • Subrata Biswas • Shakhrul Iman Siam • Omar Tawhid Imam • Dr. Celia Shahnaz
General	
Used framework (keras, pytorch, tensroflow ...etc.)	Keras,Tensorflow
If a ready model was used, please state the modifications and tweaks done? if Any?	We have used Efficientnet B2 as the backbone. And have used a novel motion and multithresholding based post-prossesing approach propossed by us for final detection part. We have used elemination of grid sensitivity while trainning the network.
Training	
Training configuration (epochs, batchsize, optimizer, scale, augment, loss function, training time on x device ...etc.)	**Training time on Kaggle(1xP100):** It took around 01:05 hour per epoch which means 7-8 epochs per 9 hours of kaggle session. **Epochs:** We trained the network for 120 epochs. And used the .h5 file saved of 117th epoch as it measured best loss and validation loss result. We trianed discontinuously through 9 hours of kaggle session. To know furthermore about trainning process please check out the “Train the network ” section of readme.md file submitted by us.

	Batchsize: 08 (On Kaggle) **Optimizer:** SGD **Scale:** 640×640×3 **Augment:** • Resizing & cropping • Changing Brightness randomly • Changing contrast randomly • Changing chroma randomly • Vertical flipping • Horizontal flipping • Sharpness • grayscaling **Loss function:** • Confidence loss: Binary Crossentropy • Location loss: Distance IoU loss • Class loss: Binary Crossentropy
Algorithm	
Describe in (200 words) the algorithm (you can add diagrams)	We used K-means on the training set to prepare 9 anchors for yolov4-based algorithm. We used custom data generator to load the training (80%) and validation (20%) data into the training model (shuffling for cross-validation). It resizes ground truth images and reshapes the boxes based on random augmentation per batch for each images. For model backbone we used Efficientnet B2 after testing various backbones, which was originally trained on 240x240 image size with 337 layers. Our algorithm performs padding-based correction and uses it for 640x640 input image size. We used SGD optimizer with Piecewise Constant Decay for warmup phase. We prepared it to work for discontinuous training too, automatically increasing learning rate once epoch 40 has been reached. For callbacks, we used early stopping (10 epoch), logging, checkpoint (save best model based on validation loss metric) and data shuffle (for cross-validation). For post-processing, we have proposed a novel method that takes into account grid-based thresholds, motion sensitivity, radial distance and object size or proportion. We have implemented an IoU based

	tracking algorithm for a reasonable fps e.g 10 fps. An average car is 4.5 meter long (3.2 meter for mini car). Even after crossing in-town speed limit 50 kmh, a car can move at most 1.4 meter per frame (at 10 fps). Hence, an IoU-based method would work well for tracking even at corner cases such as two small cars following each other closely at high speed.
If you have more than a submission, describe fail and success scenarios (how best submission was reached)	We used higher theshold at the initial satges of the submission for all grids with or without motion. Then we lowered the thresholds for all the grids regardless of motion which resulted in some improvement of result but added more false positive cases. After that we lowered the threshold for the grids where motion was detected as it's highly probable that a grid with motion will have the presence of a moving vehicle. We latter used distance from center and bounding box size based method keeping the distiortion caused by fish-eye camera in mind for further improvement and generilzation as it's highly unlikly to detect a vehicle of large size at the edeges of the image.
[Optional] **References (journal, book ...etc.)**	1.Alexey Bochkovskiy, H.Y. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection*arXiv*. 2.Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement*arXiv*. 3.Zhaohui Zheng, D. (2020). Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression*arXiv*. 4. david8862, keras-yolov3-model-set https://github.com/david8862/keras-YOLOv3-model-set (2020) 5. Tan, M., & Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946.