Team: DeepZS

Towards Real Time Drone Detection on Embedded Platforms

2019 DAC System Design Contest

Jianing Deng, Tianhao Shen, Xingang Yan, Yufei Chen, Cheng Zhuo

Huifan Zhang, Ruoyu Wang*, Pingqiang Zhou

浙江大学
ZHEJIANG UNIVERSITY

上海科技大学
ShanghaiTech University

Challenges for Drone Detection

- TINY objects

Small object (truck2-000001)

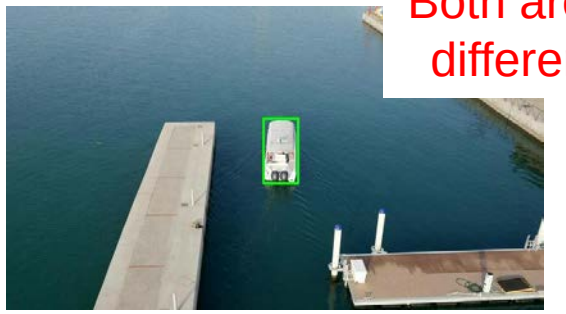
Small object (truck2-000006)

Small object (truck2-000032)

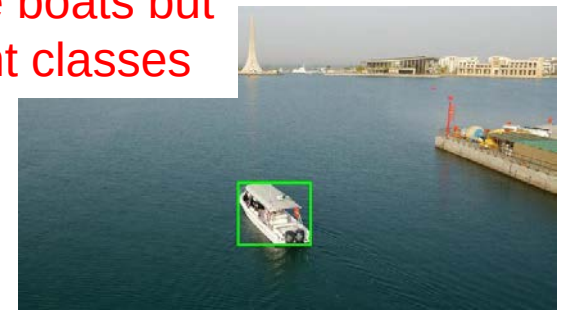
Challenges for Drone Detection

- Distraction: A or B?

Both are boats but different classes



Distractions (boat1-000000)



Distractions (boat2-000000)

One target object with something similar

Distractions (whale1-000000)

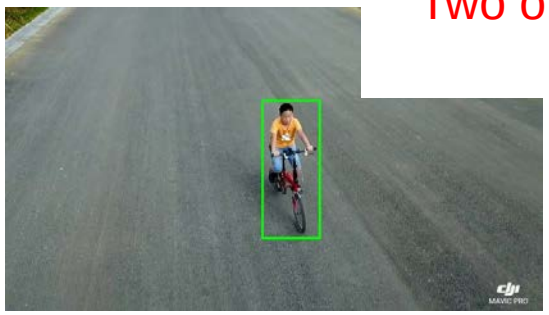
Distractions (whale1-000009)

One target object with many similar objects

Distractions (building2-000001)

Distractions (building2-000008)

Two objects in one image



Distractions (riding2-000000)

Distractions (riding3-000024)

What Makes It Harder?

- Platform: Nvidia Jetson TX2
- Problem size: 95 classes
  - Detection just based on the image itself
- Speed constraint: >20fps for real time effect
  - The faster, the higher score
- Energy constraint
  - Implicit, but the smaller, the better

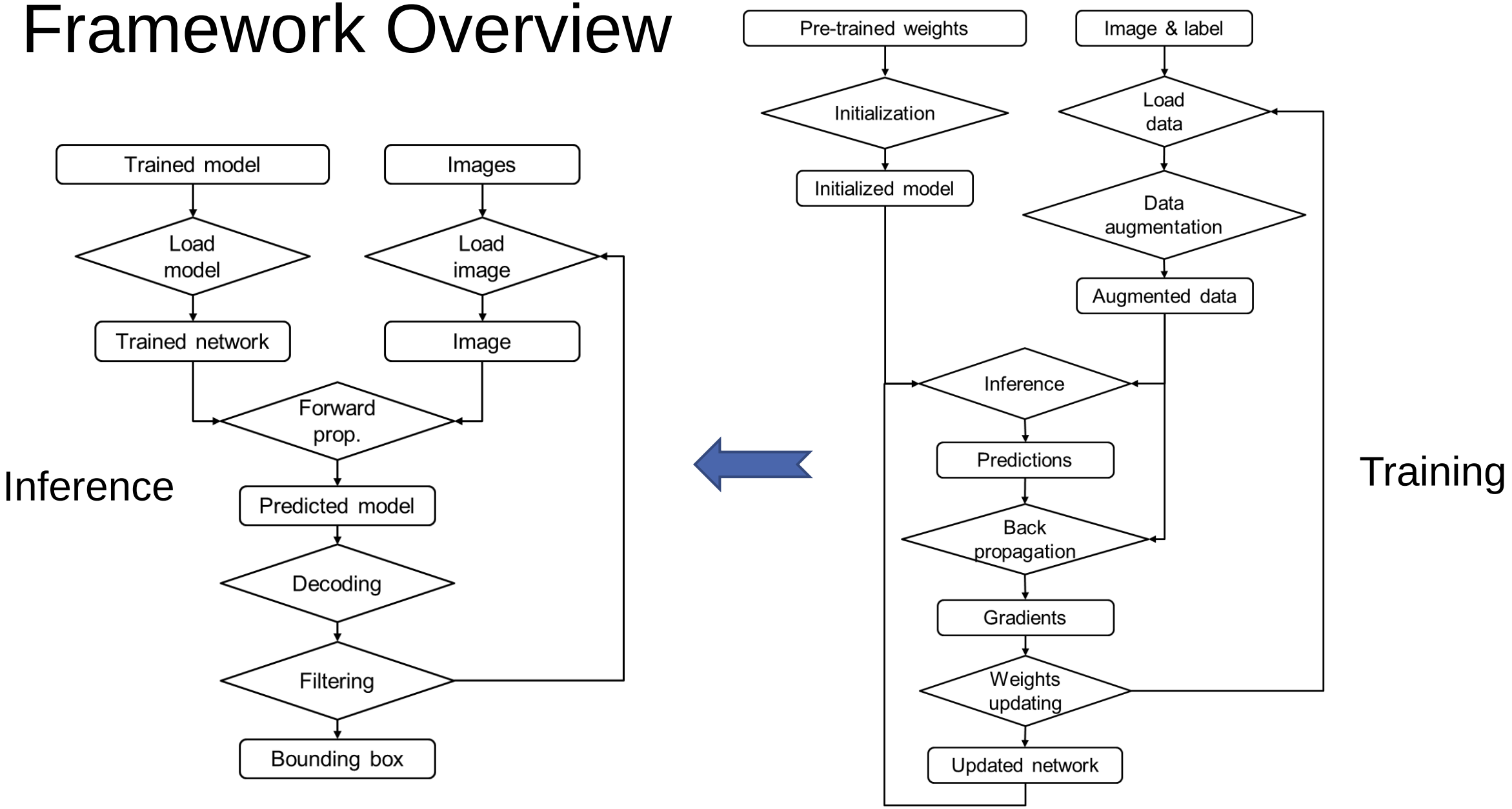
What Makes Life a Little Easier

- ONE target object/class to be detected per picture
- No classification, only detection
  - Do not need to worry about A or B question for boat classes
- Color images with fixed sizes: 640x360 pixels

Design Metrics

- Accuracy
  - IoU
- Speed
  - >20fps
- Energy
  - The smaller, the better

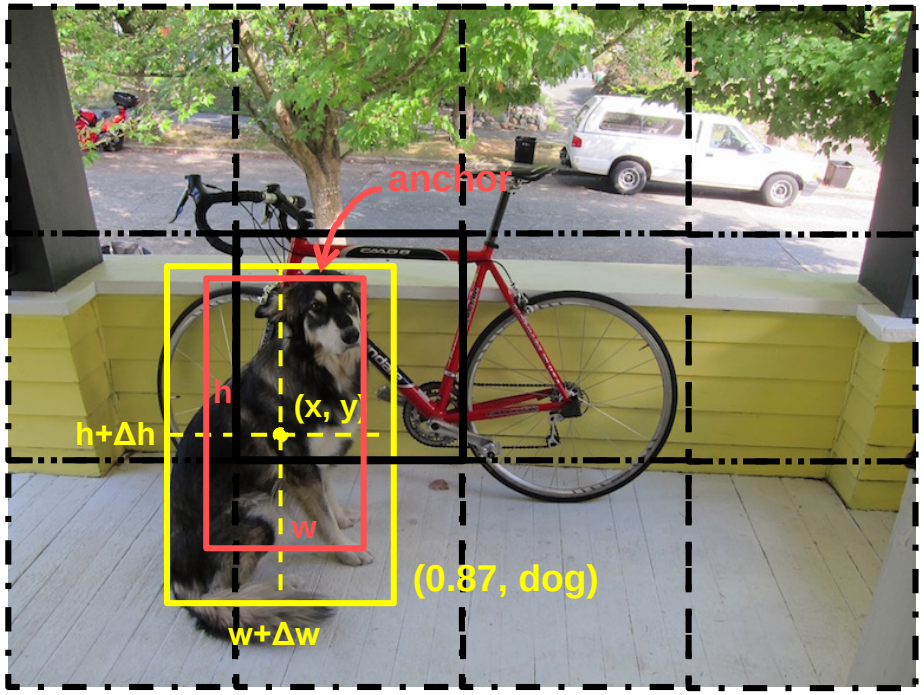
Framework Overview



Candidate Frameworks

- SSD (Single Shot MultiBox Detector)
  - ZFNet/VGGNet as backbone
  - Good accuracy even with small image size
  - Difficult to reach 20 fps
- Faster RCNN
  - VGGNet/ResNet as backbone
  - Highest accuracy but slowest
- YOLO
  - Darknet-19 as backbone
  - Simple structure with fast execution
  - Maintain a proper accuracy ranges
- Tiny YOLO
  - A smaller model (darknet reference network) as backbone
  - Fastest and easier to customize

YOLO: You Only Look Once



- Predict bounding boxes and class probabilities directly from full images
- Divide the input image into MxN cells
- Predict 6 bounding boxes in each cell using the feature maps
- Each bounding box is represented by
  - x, y (central coordinates relative to the cell)
  - Δw, Δh (shape offset relative to anchor shape)

Source for YOLO: J. Redmon and A. Farhadi, YOLO9000: Better, Faster, Stronger, CVPR 2017.

Contest Journey

- Last year

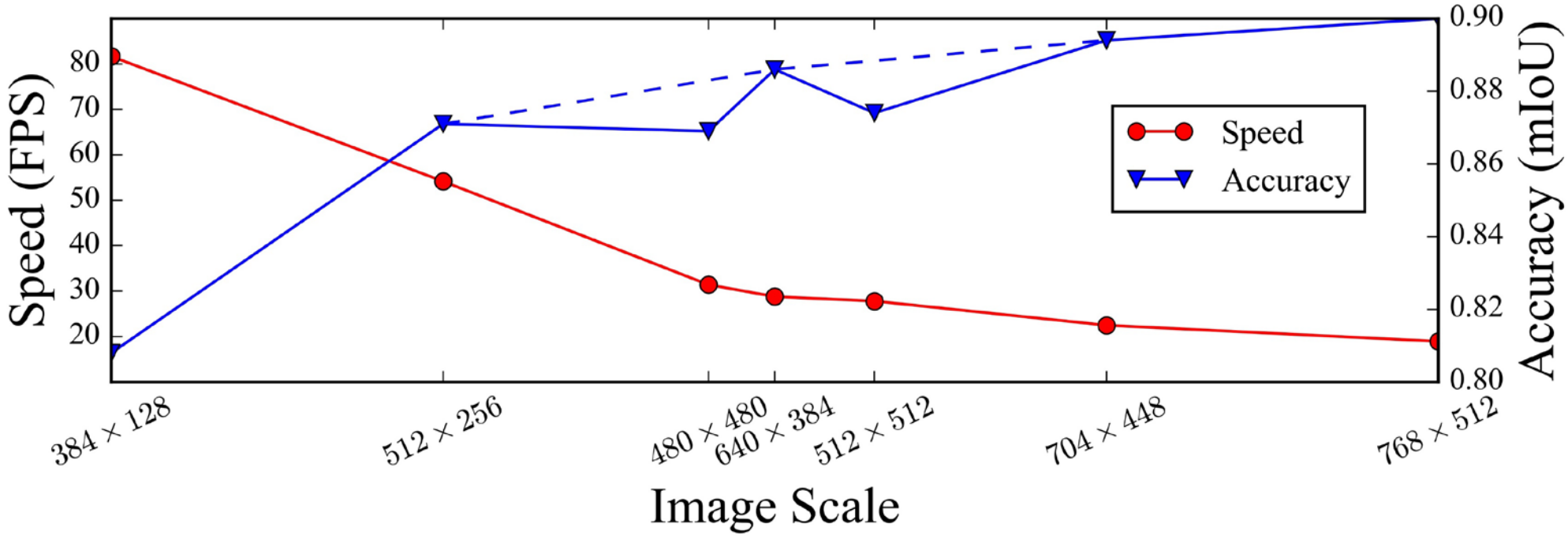
Local Accuracy	Speed (mode2)	Power (mode2)	Energy Efficiency
0.879	22	10.088	2.18

- This year

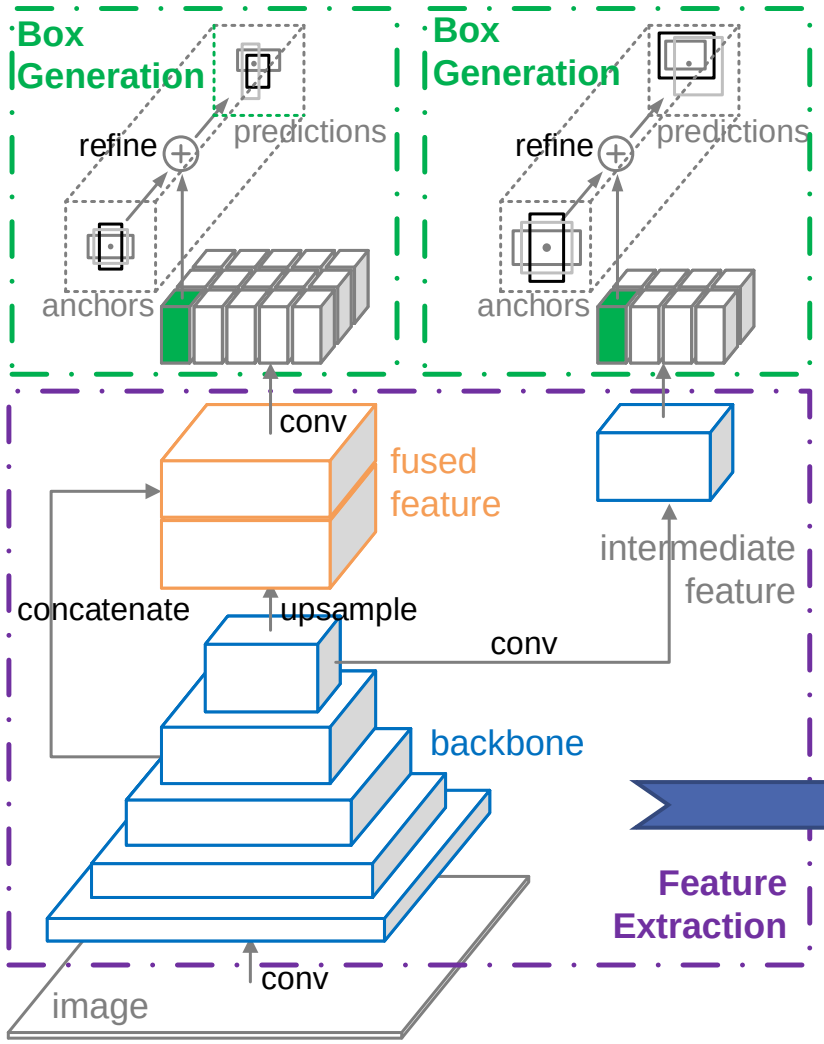
Local Accuracy	Speed (mode2)	Power (mode2)	Energy Efficiency
0.894	25	10.260	2.44

#1: Choice of Image Scales

- Input image size: 640x360
- Trade-off between Image Scale & Running Speed
  - We used 704x448 in our final submission



#2: Better Network Structure

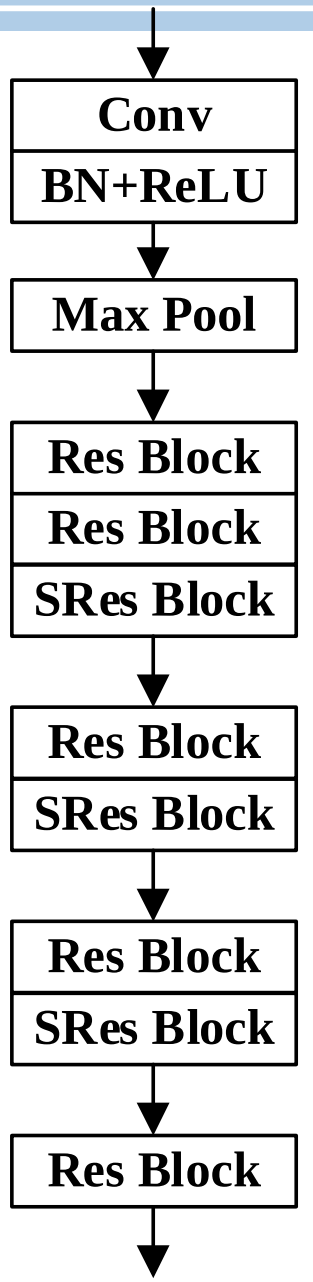


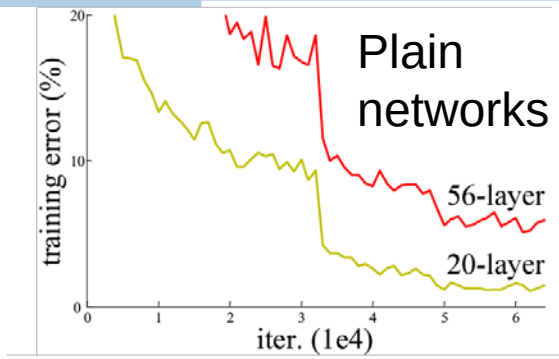
Coarser Resolutions

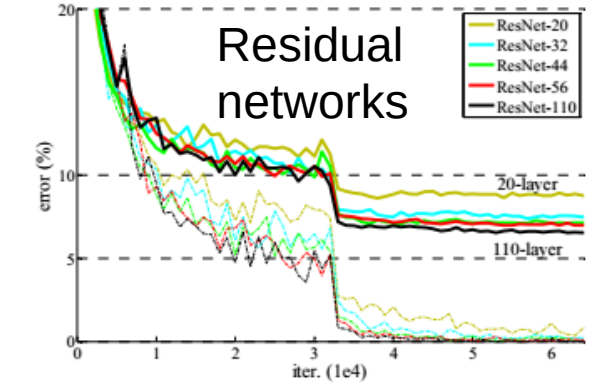
Stronger Semantics

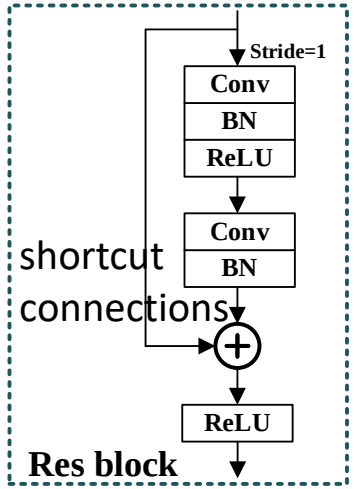
Down sample ~32x

Replace DarkNet7 w/ ResNet18





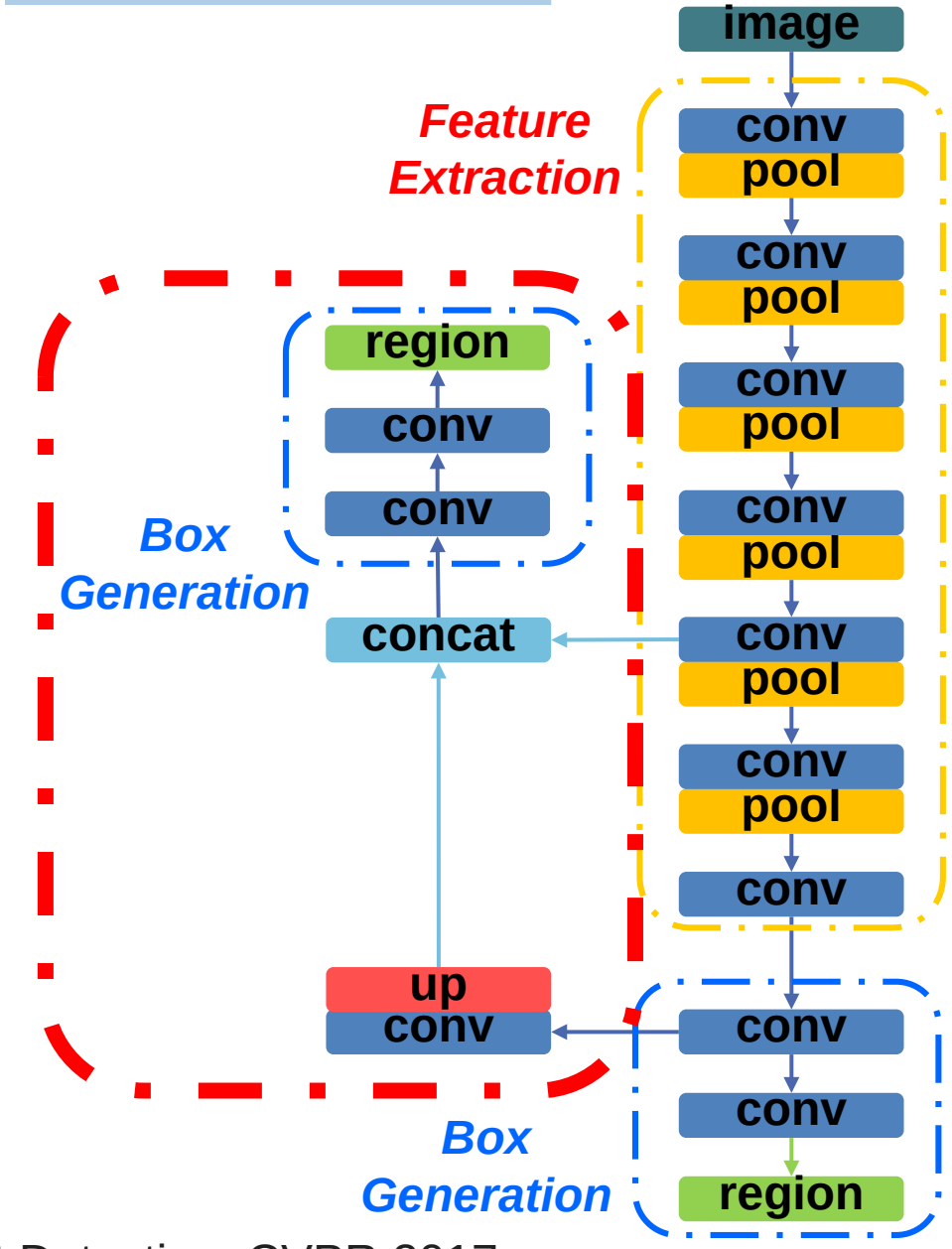




Source for ResNet18: K. He, J. Sun et. al, Deep Residual Learning for Image Recognition, CVPR 2016.

Feature Pyramid Network

- To improve location accuracy, combine the shallow features (semantically weak but w/ high resolution) with the deep features (w/ low resolution but semantically strong)
- Use *nearest neighbor up-sampling* to align two feature maps with different resolutions
  - Concatenating them by channels instead of element-wise addition



Source for FPN: T.-Y. Lin et. al, Feature Pyramid Networks for Object Detection, CVPR 2017.

#3: Focal Loss

- Focal loss is a powerful loss function to address class imbalance

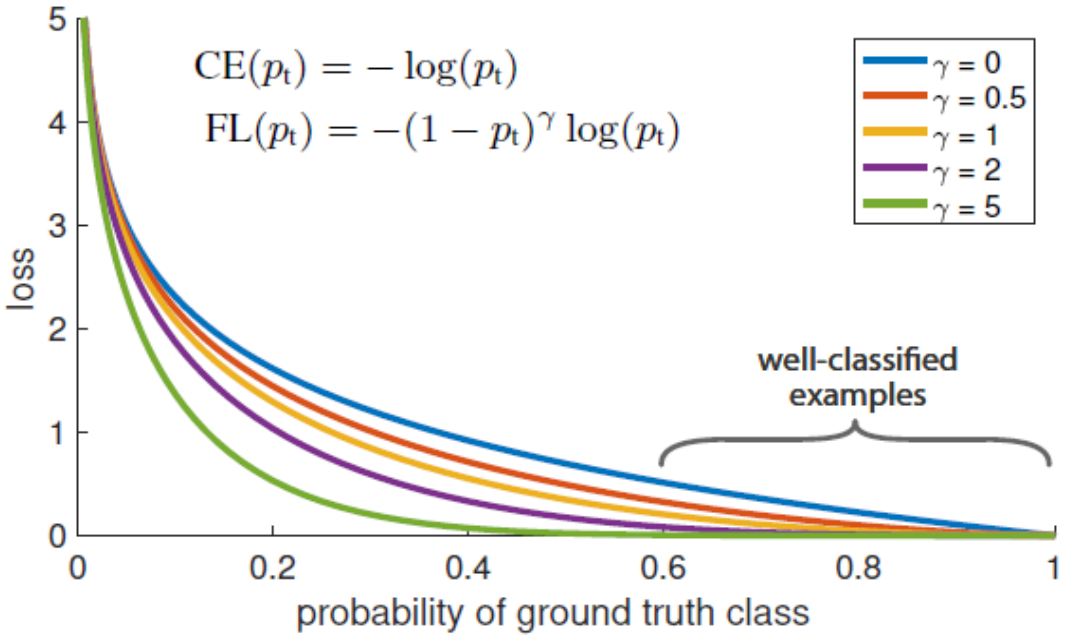
- Modify the original Loss function

$$\mathcal{L} = L2Loss_{confidence} + L2Loss_{location}$$

to

$$\mathcal{L} = FocalLoss_{confidence} + L2Loss_{location}$$

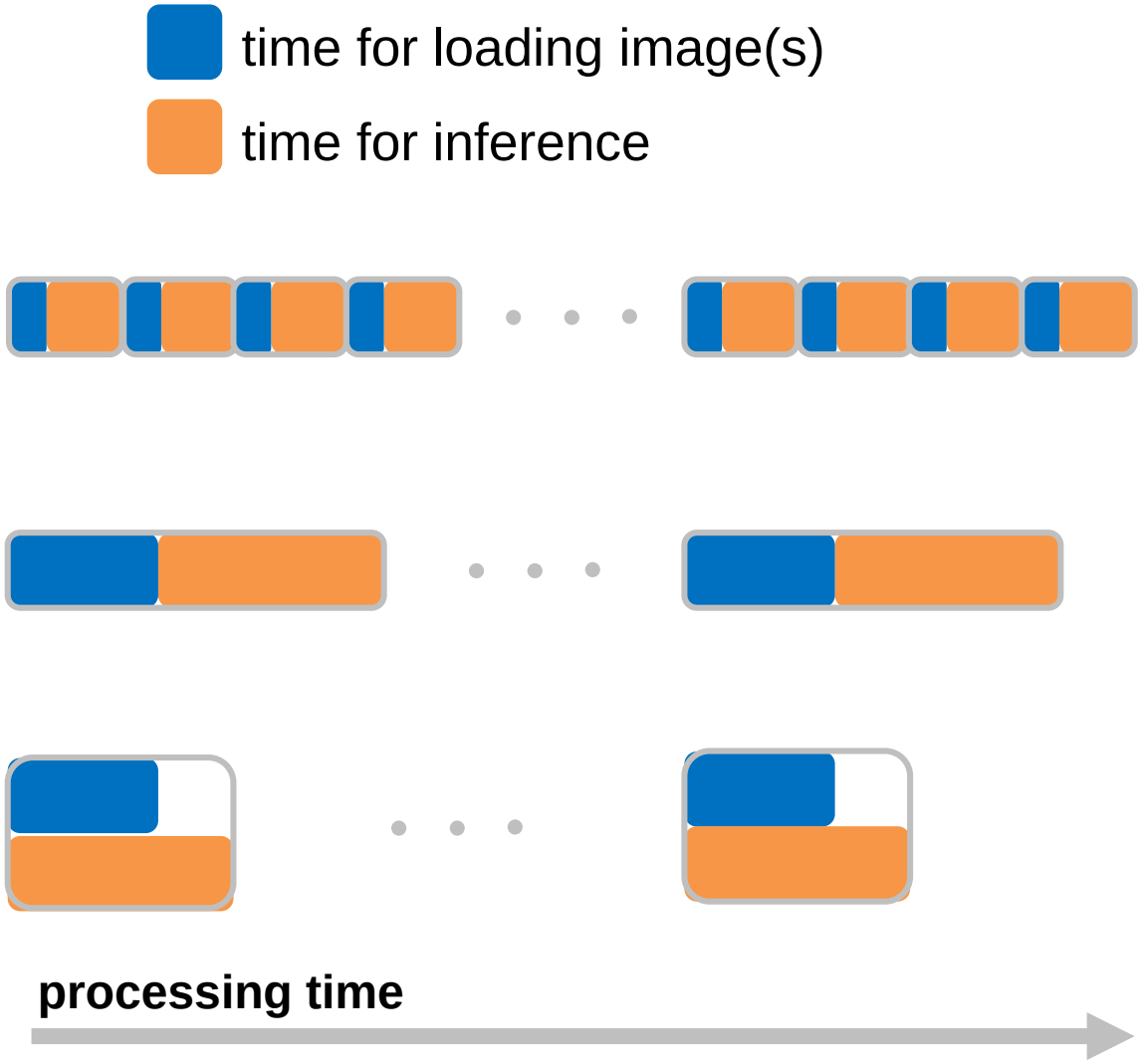
  - Resolve the imbalance between the single ground truth box and the candidate boxes



Source: T. Lin, et. al., Focal Loss for Dense Object Detection, ICCV 2017.

#4: Tuning for Speedup

- Serial load and process, one image per batch
  - **FPS: 14.9 (TX2 mode 2)**
  - **Energy Efficiency: 2.232**
- Make full use of GPU, several images per batch
  - **FPS: 16.7**
  - **Energy Efficiency: 2.331**
- Use multithreading to load images and inference in parallel
  - **FPS: 28.5**
  - **Energy Efficiency: 2.774**



Conclusions

- Addressed multiple challenges for drone object detection
- The ResNet18 backbone improves both the accuracy and running speed compared with DarkNet7 (backbone we used last year)
- The use of FPN and focal loss helps tiny object detection and distraction problems