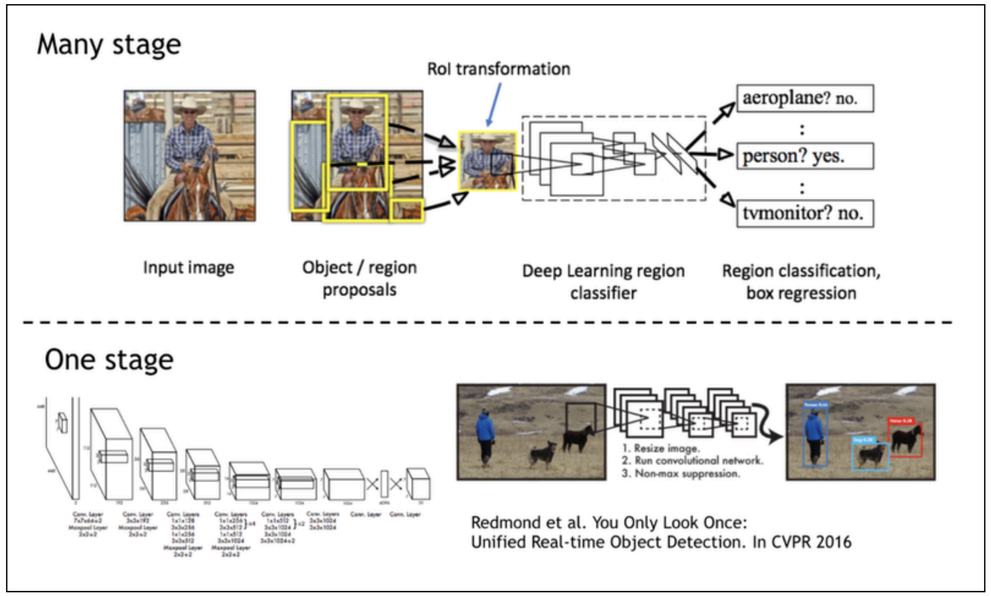
Yolo v3

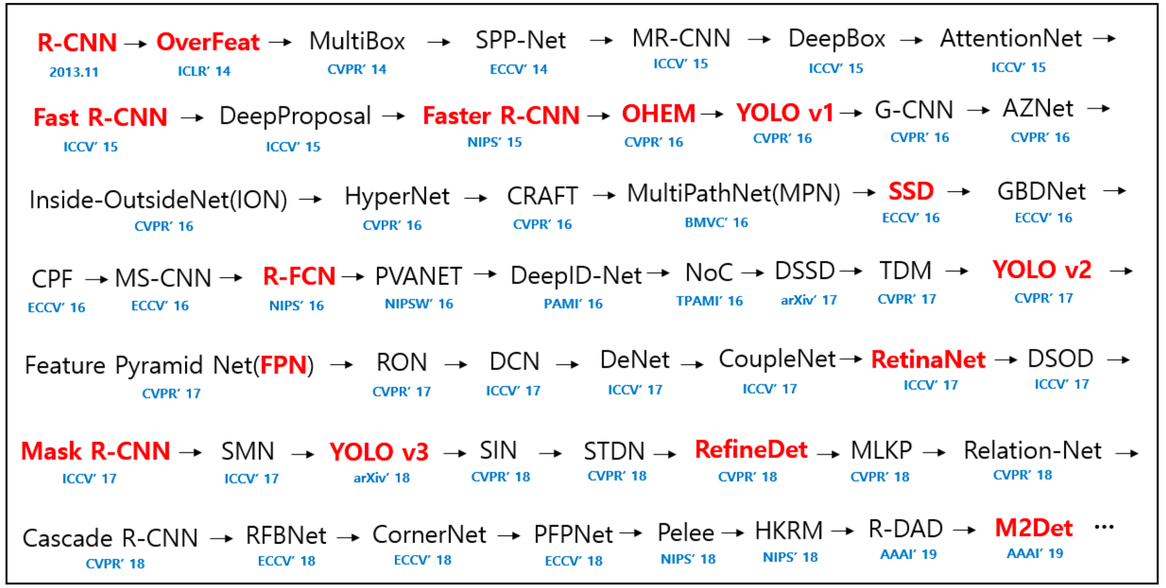
object detection

- **one-stage detector (vs 2-stage)**



- **Yolo 목적: real-time application (speed > accuracy)**
- **Yolo v3: v2보다 속도 다소 ⇩, 정확도 ⇧**

- **object detection 연대기**

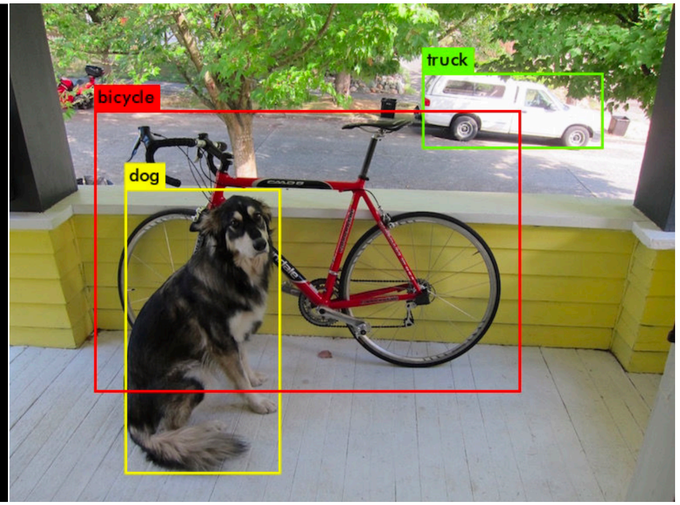


- **Yolo 단점: 그리드 영역 단위로 bbox prediction 진행 → 그리드보다 작은 크기의 물체의 detection 정확도 떨어짐**
- **Yolo v3에서 작은 object에 대한 성능 개선**

Yolo v3 - 서버컴에서 실행

```
./darknet detect cfg/yolov3.cfg yolov3.weights data/dog.jpg
```

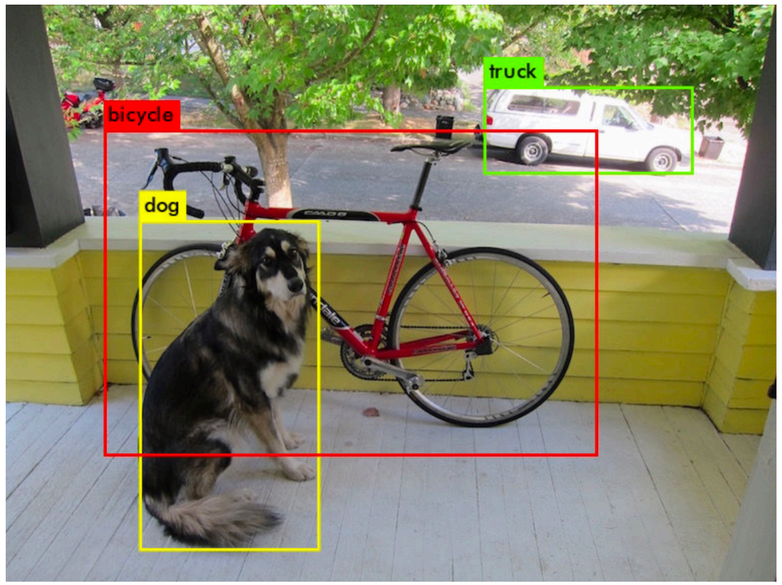
```
 user  ~/Desktop/yolov3  master  ./darknet detect cfg/yolov3.cfg yolov3.weights data/dog.jpg
layer     filters    size              input                output
    0 conv     32  3 x 3 / 1   608 x 608 x   3   ->   608 x 608 x  32  0.639 BFLOPs
    1 conv     64  3 x 3 / 2   608 x 608 x  32   ->   304 x 304 x  64  3.407 BFLOPs
    2 conv     32  1 x 1 / 1   304 x 304 x  64   ->   304 x 304 x  32  0.379 BFLOPs
    3 conv     64  3 x 3 / 1   304 x 304 x  32   ->   304 x 304 x  64  3.407 BFLOPs
    4 res    1                 304 x 304 x  64   ->   304 x 304 x  64
    5 conv    128  3 x 3 / 2   304 x 304 x  64   ->   152 x 152 x 128  3.407 BFLOPs
    6 conv     64  1 x 1 / 1   152 x 152 x 128   ->   152 x 152 x  64  0.379 BFLOPs
    7 conv    128  3 x 3 / 1   152 x 152 x  64   ->   152 x 152 x 128  3.407 BFLOPs
    8 res    5                 152 x 152 x 128   ->   152 x 152 x 128
    9 conv     64  1 x 1 / 1   152 x 152 x 128   ->   152 x 152 x  64  0.379 BFLOPs
   10 conv    128  3 x 3 / 1   152 x 152 x  64   ->   152 x 152 x 128  3.407 BFLOPs
   11 res    8                 152 x 152 x 128   ->   152 x 152 x 128
   12 conv    256  3 x 3 / 2   152 x 152 x 128   ->    76 x  76 x 256  3.407 BFLOPs
   13 conv    128  1 x 1 / 1    76 x  76 x 256   ->    76 x  76 x 128  0.379 BFLOPs
   14 conv    256  3 x 3 / 1    76 x  76 x 128   ->    76 x  76 x 256  3.407 BFLOPs
   15 res   12                  76 x  76 x 256   ->    76 x  76 x 256
   16 conv    128  1 x 1 / 1    76 x  76 x 256   ->    76 x  76 x 128  0.379 BFLOPs
   17 conv    256  3 x 3 / 1    76 x  76 x 128   ->    76 x  76 x 256  3.407 BFLOPs
   18 res   15                  76 x  76 x 256   ->    76 x  76 x 256
   19 conv    128  1 x 1 / 1    76 x  76 x 256   ->    76 x  76 x 128  0.379 BFLOPs
   20 conv    256  3 x 3 / 1    76 x  76 x 128   ->    76 x  76 x 256  3.407 BFLOPs
   21 res   18                  76 x  76 x 256   ->    76 x  76 x 256
   22 conv    128  1 x 1 / 1    76 x  76 x 256   ->    76 x  76 x 128  0.379 BFLOPs
   23 conv    256  3 x 3 / 1    76 x  76 x 128   ->    76 x  76 x 256  3.407 BFLOPs
   24 res   21                  76 x  76 x 256   ->    76 x  76 x 256
   25 conv    128  1 x 1 / 1    76 x  76 x 256   ->    76 x  76 x 128  0.379 BFLOPs
   26 conv    256  3 x 3 / 1    76 x  76 x 128   ->    76 x  76 x 256  3.407 BFLOPs
   27 res   24                  76 x  76 x 256   ->    76 x  76 x 256
   28 conv    128  1 x 1 / 1    76 x  76 x 256   ->    76 x  76 x 128  0.379 BFLOPs
   29 conv    256  3 x 3 / 1    76 x  76 x 128   ->    76 x  76 x 256  3.407 BFLOPs
   30 res   27                  76 x  76 x 256   ->    76 x  76 x 256
   31 conv    128  1 x 1 / 1    76 x  76 x 256   ->    76 x  76 x 128  0.379 BFLOPs
   32 conv    256  3 x 3 / 1    76 x  76 x 128   ->    76 x  76 x 256  3.407 BFLOPs
   33 res   30                  76 x  76 x 256   ->    76 x  76 x 256
   34 conv    128  1 x 1 / 1    76 x  76 x 256   ->    76 x  76 x 128  0.379 BFLOPs
   35 conv    256  3 x 3 / 1    76 x  76 x 128   ->    76 x  76 x 256  3.407 BFLOPs
   36 res   33                  76 x  76 x 256   ->    76 x  76 x 256
   37 conv    512  3 x 3 / 2    76 x  76 x 256   ->    38 x  38 x 512  3.407 BFLOPs
   38 conv    256  1 x 1 / 1    38 x  38 x 512   ->    38 x  38 x 256  0.379 BFLOPs
   39 conv    512  3 x 3 / 1    38 x  38 x 256   ->    38 x  38 x 512  3.407 BFLOPs
   40 res   37                  38 x  38 x 512   ->    38 x  38 x 512
   41 conv    256  1 x 1 / 1    38 x  38 x 512   ->    38 x  38 x 256  0.379 BFLOPs
   42 conv    512  3 x 3 / 1    38 x  38 x 256   ->    38 x  38 x 512  3.407 BFLOPs
   43 res   40                  38 x  38 x 512   ->    38 x  38 x 512
   44 conv    256  1 x 1 / 1    38 x  38 x 512   ->    38 x  38 x 256  0.379 BFLOPs
   45 conv    512  3 x 3 / 1    38 x  38 x 256   ->    38 x  38 x 512  3.407 BFLOPs
   46 res   43                  38 x  38 x 512   ->    38 x  38 x 512
   47 conv    256  1 x 1 / 1    38 x  38 x 512   ->    38 x  38 x 256  0.379 BFLOPs
   48 conv    512  3 x 3 / 1    38 x  38 x 256   ->    38 x  38 x 512  3.407 BFLOPs
   49 res   46                  38 x  38 x 512   ->    38 x  38 x 512
   50 conv    256  1 x 1 / 1    38 x  38 x 512   ->    38 x  38 x 256  0.379 BFLOPs
   51 conv    512  3 x 3 / 1    38 x  38 x 256   ->    38 x  38 x 512  3.407 BFLOPs
   52 res   49                  38 x  38 x 512   ->    38 x  38 x 512
   53 conv    256  1 x 1 / 1    38 x  38 x 512   ->    38 x  38 x 256  0.379 BFLOPs
   54 conv    512  3 x 3 / 1    38 x  38 x 256   ->    38 x  38 x 512  3.407 BFLOPs
   55 res   52                  38 x  38 x 512   ->    38 x  38 x 512
   56 conv    256  1 x 1 / 1    38 x  38 x 512   ->    38 x  38 x 256  0.379 BFLOPs
   57 conv    512  3 x 3 / 1    38 x  38 x 256   ->    38 x  38 x 512  3.407 BFLOPs
   58 res   55                  38 x  38 x 512   ->    38 x  38 x 512
   59 conv    256  1 x 1 / 1    38 x  38 x 512   ->    38 x  38 x 256  0.379 BFLOPs
   60 conv    512  3 x 3 / 1    38 x  38 x 256   ->    38 x  38 x 512  3.407 BFLOPs
   61 res   58                  38 x  38 x 512   ->    38 x  38 x 512
   62 conv   1024  3 x 3 / 2    38 x  38 x 512   ->    19 x  19 x1024  3.407 BFLOPs
   63 conv    512  1 x 1 / 1    19 x  19 x1024   ->    19 x  19 x 512  0.379 BFLOPs
   64 conv   1024  3 x 3 / 1    19 x  19 x 512   ->    19 x  19 x1024  3.407 BFLOPs
   65 res   62                  19 x  19 x1024   ->    19 x  19 x1024
   66 conv    512  1 x 1 / 1    19 x  19 x1024   ->    19 x  19 x 512  0.379 BFLOPs
   67 conv   1024  3 x 3 / 1    19 x  19 x 512   ->    19 x  19 x1024  3.407 BFLOPs
   68 res   65                  19 x  19 x1024   ->    19 x  19 x1024
   69 conv    512  1 x 1 / 1    19 x  19 x1024   ->    19 x  19 x 512  0.379 BFLOPs
   70 conv   1024  3 x 3 / 1    19 x  19 x 512   ->    19 x  19 x1024  3.407 BFLOPs
   71 res   68                  19 x  19 x1024   ->    19 x  19 x1024
   72 conv    512  1 x 1 / 1    19 x  19 x1024   ->    19 x  19 x 512  0.379 BFLOPs
   73 conv   1024  3 x 3 / 1    19 x  19 x 512   ->    19 x  19 x1024  3.407 BFLOPs
   74 res   71                  19 x  19 x1024   ->    19 x  19 x1024
   75 conv    512  1 x 1 / 1    19 x  19 x1024   ->    19 x  19 x 512  0.379 BFLOPs
   76 conv   1024  3 x 3 / 1    19 x  19 x 512   ->    19 x  19 x1024  3.407 BFLOPs
   77 conv    512  1 x 1 / 1    19 x  19 x1024   ->    19 x  19 x 512  0.379 BFLOPs
   78 conv   1024  3 x 3 / 1    19 x  19 x 512   ->    19 x  19 x1024  3.407 BFLOPs
   79 conv    512  1 x 1 / 1    19 x  19 x1024   ->    19 x  19 x 512  0.379 BFLOPs
   80 conv   1024  3 x 3 / 1    19 x  19 x 512   ->    19 x  19 x1024  3.407 BFLOPs
   81 conv    255  1 x 1 / 1    19 x  19 x1024   ->    19 x  19 x 255  0.189 BFLOPs
   82 yolo
   83 route  79
   84 conv    256  1 x 1 / 1    19 x  19 x 512   ->    19 x  19 x 256  0.095 BFLOPs
   85 upsample           2x     19 x  19 x 256   ->    38 x  38 x 256
   86 route  85 61
   87 conv    256  1 x 1 / 1    38 x  38 x 768   ->    38 x  38 x 256  0.568 BFLOPs
   88 conv    512  3 x 3 / 1    38 x  38 x 256   ->    38 x  38 x 512  3.407 BFLOPs
   89 conv    256  1 x 1 / 1    38 x  38 x 512   ->    38 x  38 x 256  0.379 BFLOPs
   90 conv    512  3 x 3 / 1    38 x  38 x 256   ->    38 x  38 x 512  3.407 BFLOPs
   91 conv    256  1 x 1 / 1    38 x  38 x 512   ->    38 x  38 x 256  0.379 BFLOPs
   92 conv    512  3 x 3 / 1    38 x  38 x 256   ->    38 x  38 x 512  3.407 BFLOPs
   93 conv    255  1 x 1 / 1    38 x  38 x 512   ->    38 x  38 x 255  0.377 BFLOPs
   94 yolo
   95 route  91
   96 conv    128  1 x 1 / 1    38 x  38 x 256   ->    38 x  38 x 128  0.095 BFLOPs
   97 upsample           2x     38 x  38 x 128   ->    76 x  76 x 128
   98 route  97 36
   99 conv    128  1 x 1 / 1    76 x  76 x 384   ->    76 x  76 x 128  0.568 BFLOPs
  100 conv    256  3 x 3 / 1    76 x  76 x 128   ->    76 x  76 x 256  3.407 BFLOPs
  101 conv    128  1 x 1 / 1    76 x  76 x 256   ->    76 x  76 x 128  0.379 BFLOPs
  102 conv    256  3 x 3 / 1    76 x  76 x 128   ->    76 x  76 x 256  3.407 BFLOPs
  103 conv    128  1 x 1 / 1    76 x  76 x 256   ->    76 x  76 x 128  0.379 BFLOPs
  104 conv    256  3 x 3 / 1    76 x  76 x 128   ->    76 x  76 x 256  3.407 BFLOPs
  105 conv    255  1 x 1 / 1    76 x  76 x 256   ->    76 x  76 x 255  0.754 BFLOPs
  106 yolo
Loading weights from yolov3.weights...Done!
data/dog.jpg: Predicted in 22.869935 seconds.
dog: 100%
truck: 92%
bicycle: 99%
 user  ~/Desktop/yolov3  master
```



Yolo v3 - test

```
./darknet detector test cfg/coco.data cfg/yolov3.cfg yolov3.weights data/dog.jpg
```

- The detect command is shorthand for a more general version of the command
- 두 명령어 동일: detect = detector test




```
  70 conv   1024  3 x 3 / 1    19 x  19 x 512   ->    19 x  19 x1024  3.407 BFLOPs
  71 res   68                   19 x  19 x1024   ->    19 x  19 x1024
  72 conv    512  1 x 1 / 1    19 x  19 x1024   ->    19 x  19 x 512  0.379 BFLOPs
  73 conv   1024  3 x 3 / 1    19 x  19 x 512   ->    19 x  19 x1024  3.407 BFLOPs
  74 res   71                   19 x  19 x1024   ->    19 x  19 x1024
  75 conv    512  1 x 1 / 1    19 x  19 x1024   ->    19 x  19 x 512  0.379 BFLOPs
  76 conv   1024  3 x 3 / 1    19 x  19 x 512   ->    19 x  19 x1024  3.407 BFLOPs
  77 conv    512  1 x 1 / 1    19 x  19 x1024   ->    19 x  19 x 512  0.379 BFLOPs
  78 conv   1024  3 x 3 / 1    19 x  19 x 512   ->    19 x  19 x1024  3.407 BFLOPs
  79 conv    512  1 x 1 / 1    19 x  19 x1024   ->    19 x  19 x 512  0.379 BFLOPs
  80 conv   1024  3 x 3 / 1    19 x  19 x 512   ->    19 x  19 x1024  3.407 BFLOPs
  81 conv    255  1 x 1 / 1    19 x  19 x1024   ->    19 x  19 x 255  0.189 BFLOPs
  82 yolo
  83 route  79
  84 conv    256  1 x 1 / 1    19 x  19 x 512   ->    19 x  19 x 256  0.095 BFLOPs
  85 upsample            2x    19 x  19 x 256   ->    38 x  38 x 256
  86 route  85 61
  87 conv    256  1 x 1 / 1    38 x  38 x 768   ->    38 x  38 x 256  0.568 BFLOPs
  88 conv    512  3 x 3 / 1    38 x  38 x 256   ->    38 x  38 x 512  3.407 BFLOPs
  89 conv    256  1 x 1 / 1    38 x  38 x 512   ->    38 x  38 x 256  0.379 BFLOPs
  90 conv    512  3 x 3 / 1    38 x  38 x 256   ->    38 x  38 x 512  3.407 BFLOPs
  91 conv    256  1 x 1 / 1    38 x  38 x 512   ->    38 x  38 x 256  0.379 BFLOPs
  92 conv    512  3 x 3 / 1    38 x  38 x 256   ->    38 x  38 x 512  3.407 BFLOPs
  93 conv    255  1 x 1 / 1    38 x  38 x 512   ->    38 x  38 x 255  0.377 BFLOPs
  94 yolo
  95 route  91
  96 conv    128  1 x 1 / 1    38 x  38 x 256   ->    38 x  38 x 128  0.095 BFLOPs
  97 upsample            2x    38 x  38 x 128   ->    76 x  76 x 128
  98 route  97 36
  99 conv    128  1 x 1 / 1    76 x  76 x 384   ->    76 x  76 x 128  0.568 BFLOPs
 100 conv    256  3 x 3 / 1    76 x  76 x 128   ->    76 x  76 x 256  3.407 BFLOPs
 101 conv    128  1 x 1 / 1    76 x  76 x 256   ->    76 x  76 x 128  0.379 BFLOPs
 102 conv    256  3 x 3 / 1    76 x  76 x 128   ->    76 x  76 x 256  3.407 BFLOPs
 103 conv    128  1 x 1 / 1    76 x  76 x 256   ->    76 x  76 x 128  0.379 BFLOPs
 104 conv    256  3 x 3 / 1    76 x  76 x 128   ->    76 x  76 x 256  3.407 BFLOPs
 105 conv    255  1 x 1 / 1    76 x  76 x 256   ->    76 x  76 x 255  0.754 BFLOPs
 106 yolo
Loading weights from yolov3.weights...Done!
data/dog.jpg: Predicted in 24.620853 seconds.
dog: 100%
truck: 92%
bicycle: 99%
```

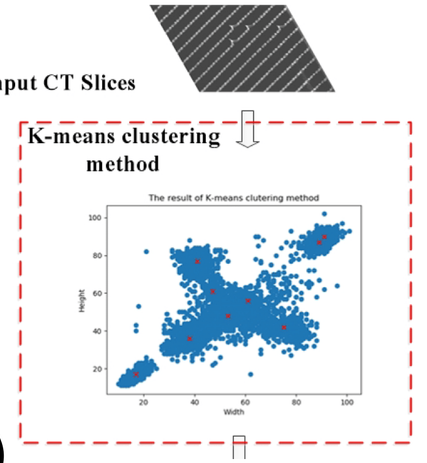
Yolo v3 - train

```
./darknet detector train cfg/voc.data cfg/yolov3-voc.cfg darknet53.conv.74
```

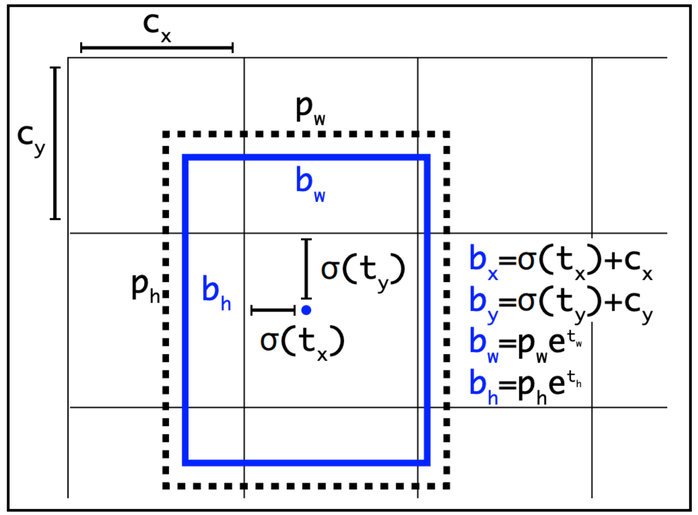
```
833: 3.052898, 7.765930 avg, 0.000481 rate, 80.197244 seconds, 833 images
Loaded: 0.000088 seconds
Region 82 Avg IOU: 0.631598, Class: 0.057110, Obj: 0.001416, No Obj: 0.001518, .5R: 1.000000, .75R: 0.000000,  count: 1
Region 94 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000085, .5R: -nan, .75R: -nan,  count: 0
Region 106 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000026, .5R: -nan, .75R: -nan,  count: 0
834: 0.798131, 7.069150 avg, 0.000484 rate, 80.049766 seconds, 834 images
Loaded: 0.000070 seconds
Region 82 Avg IOU: 0.846931, Class: 0.064752, Obj: 0.000796, No Obj: 0.001429, .5R: 1.000000, .75R: 1.000000,  count: 1
Region 94 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000093, .5R: -nan, .75R: -nan,  count: 0
Region 106 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000025, .5R: -nan, .75R: -nan,  count: 0
835: 1.097970, 6.472033 avg, 0.000486 rate, 80.260722 seconds, 835 images
Loaded: 0.000077 seconds
Region 82 Avg IOU: 0.533565, Class: 0.123666, Obj: 0.001857, No Obj: 0.001450, .5R: 1.000000, .75R: 0.000000,  count: 2
Region 94 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000127, .5R: -nan, .75R: -nan,  count: 0
Region 106 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000013, .5R: -nan, .75R: -nan,  count: 0
836: 2.047835, 6.029613 avg, 0.000488 rate, 79.784090 seconds, 836 images
Loaded: 0.000071 seconds
Region 82 Avg IOU: 0.248816, Class: 0.034317, Obj: 0.001530, No Obj: 0.000663, .5R: 0.000000, .75R: 0.000000,  count: 1
Region 94 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000182, .5R: -nan, .75R: -nan,  count: 0
Region 106 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000109, .5R: -nan, .75R: -nan,  count: 0
837: 2.806063, 5.707258 avg, 0.000491 rate, 80.220300 seconds, 837 images
Loaded: 0.000092 seconds
Region 82 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.004298, .5R: -nan, .75R: -nan,  count: 0
Region 94 Avg IOU: 0.548553, Class: 0.484042, Obj: 0.000622, No Obj: 0.000170, .5R: 0.500000, .75R: 0.000000,  count: 2
Region 106 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000025, .5R: -nan, .75R: -nan,  count: 0
838: 2.337715, 5.370304 avg, 0.000493 rate, 79.821647 seconds, 838 images
Loaded: 0.000088 seconds
Region 82 Avg IOU: 0.365163, Class: 0.027143, Obj: 0.001777, No Obj: 0.000900, .5R: 0.000000, .75R: 0.000000,  count: 1
Region 94 Avg IOU: 0.455324, Class: 0.114283, Obj: 0.000251, No Obj: 0.000089, .5R: 0.666667, .75R: 0.000000,  count: 3
Region 106 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000011, .5R: -nan, .75R: -nan,  count: 0
839: 15.034252, 6.336699 avg, 0.000496 rate, 79.508525 seconds, 839 images
Loaded: 0.000099 seconds
```

Bounding box 예측

변경사항 1



- k-means clustering을 이용해 Anchor Box(p_* 정해짐)를 생성 (예측 전 이미 정해져 있음)
  - 총 9개 (3 bbox x 3 feature map)의 anchor box 결정
  - Anchor Box 생성 후 예측 시작 (효율적으로 예측하기 위함)
- t_*: 예측하는 대상
- b_*: 예측 후, bbox의 중심점 위치 & 크기 (예측한 t_*에서 적절한 계산을 통해 구함)
  - 기존에는 그리드 중심점을 기준으로 예측했지만, 좌상단 기준에서의 offset (c_*) & Anchor box (p_*) 의 비율을 조절하여 예측
  - Loss는 b_*로 변형한 후 구함
- b_x, b_y: 예측한 bbox의 중심점 위치
  - ∂ (t_x, t_y) (0~1), c_x, c_y (좌상단 기준에서의 offset)
  - center 좌표가 cell 중심을 너무 벗어나지 않도록 시그모이드 값(0~1)으로 조절
- b_w, b_h: 예측한 bbox의 비율 크기
  - p_x, p_y: 사전에 정의된 Anchor box 크기 비율에 지수승을 곱함



Bounding box 예측 - CODE

t_x 와 직접 L1 loss를 통해 학습시키는 방식

```
box get_yolo_box(float *x, float *biases, int n, int index, int i, int j, int lw, int lh, int w, int h, int stride)
{
    box b;
    b.x = (i + x[index + 0*stride]) / lw;
    b.y = (j + x[index + 1*stride]) / lh;
    b.w = exp(x[index + 2*stride]) * biases[2*n]   / w;
    b.h = exp(x[index + 3*stride]) * biases[2*n+1] / h;
    return b;
}

float delta_yolo_box(box truth, float *x, float *biases, int n, int index, int i, int j, int lw, int lh, int w, int h, float *delt
a, float scale, int stride)
{
    box pred = get_yolo_box(x, biases, n, index, i, j, lw, lh, w, h, stride);
    float iou = box_iou(pred, truth);

    float tx = (truth.x*lw - i);
    float ty = (truth.y*lh - j);
    float tw = log(truth.w*w / biases[2*n]);
    float th = log(truth.h*h / biases[2*n + 1]);

    delta[index + 0*stride] = scale * (tx - x[index + 0*stride]);
    delta[index + 1*stride] = scale * (ty - x[index + 1*stride]);
    delta[index + 2*stride] = scale * (tw - x[index + 2*stride]);
    delta[index + 3*stride] = scale * (th - x[index + 3*stride]);
    return iou;
}
```

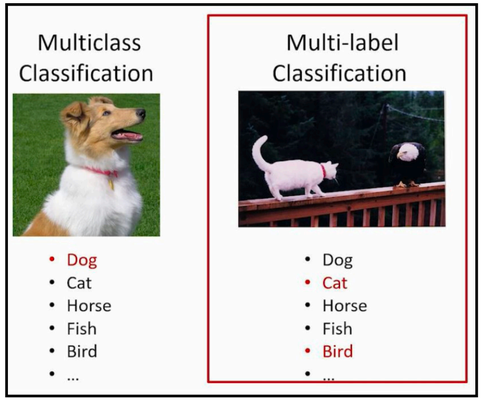
$$b_* = \sigma(t_*) + c_x$$
$$\sigma(t_*) = b_* - c_*$$
$$t_* = log(b_* - c_*)$$

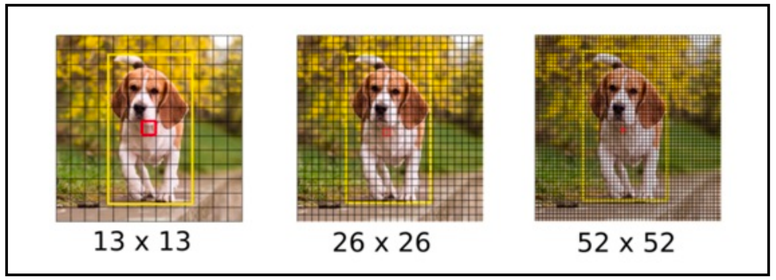
class 예측

변경사항 2

https://herbwood.tistory.com/21

- classification
  - softmax 분류기 -> logistic 분류기
  - softmax: n개의 클래스 중 하나를 결정
  - logistic: 각 class 마다 binary cross-entropy 수행
    - n개의 클래스에 대한 각각 확률값 (0~1)
- BackBone: Darknet 53
  - Google Net -> Darknet 19 -> Darknet 53 (53 conv)
  - ResNet 추가
- Network (추가)
  - output feature map 3개: 각각 size가 다름
  - 네트워크 모양 추가 공부 필요





Yolo Loss

Bounding box 예측

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2 \qquad (3)$$

- Yolo v1 - Loss function
- **total Loss: 3개의 Loss를 합산**

- **B-box regression Loss**
- **Object Confidence Loss**
- **Classification Loss**

```
void correct_yolo_boxes(detection *dets, int n, int w, int h, int netw, int neth, int relative)
{
    int i;
    int new_w=0;
    int new_h=0;
    if (((float)netw/w) < ((float)neth/h)) {
        new_w = netw;
        new_h = (h * netw)/w;
    } else {
        new_h = neth;
        new_w = (w * neth)/h;
    }
    for (i = 0; i < n; ++i){
        box b = dets[i].bbox;
        b.x =  (b.x - (netw - new_w)/2./netw) / ((float)new_w/netw);
        b.y =  (b.y - (neth - new_h)/2./neth) / ((float)new_h/neth);
        b.w *= (float)netw/new_w;
        b.h *= (float)neth/new_h;
        if(!relative){
            b.x *= w;
            b.w *= w;
            b.y *= h;
            b.h *= h;
        }
        dets[i].bbox = b;
    }
}
```

Herbarium 2022 - FGVC9

식물 Dataset

- **The categories in the training_metadata.json contain three levels of hierarchical structure, family - genus - species**
- **Plant Dataset - object detection 가능한 것인지 확인 못 함**

The Herbarium 2022: Flora of North America is a part of a project of the New York Botanical Garden funded by the National Science Foundation to build tools to identify novel plant species around the world. The dataset strives to represent all known vascular plant taxa in North America, using images gathered from 60 different botanical institutions around the world.