Assignment 2

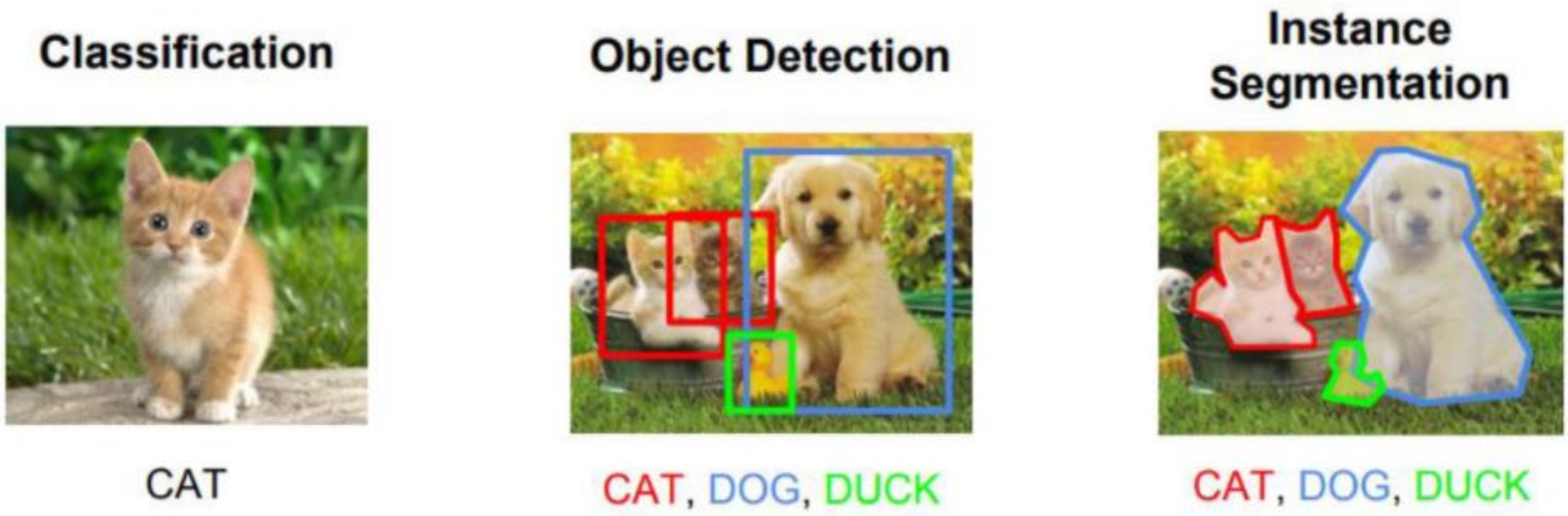
Hyeongrok Han

Department of Electrical and Computer Engineering

Seoul National University

http://data.snu.ac.kr

Computer vision tasks



(Assignment 2-1)

(Assignment 2-2)

- Besides, image reconstruction, image synthesis, style transfer, etc…

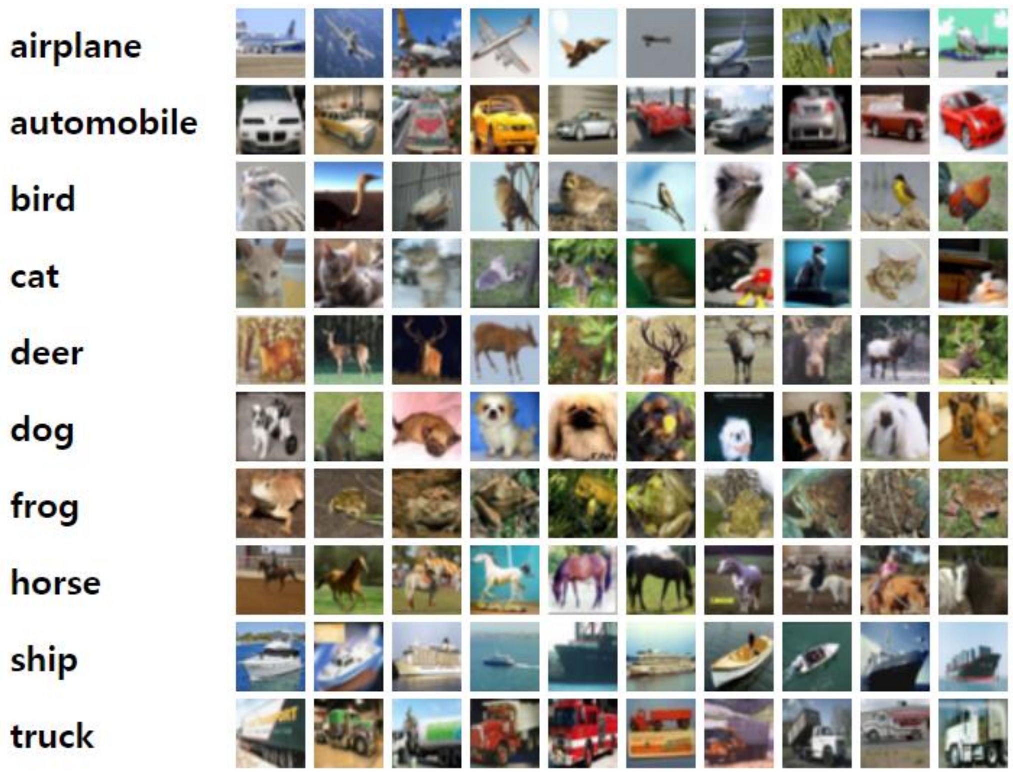
Assignment_2-1 Objective (Image classification)

- Problem 1: Simple CNN model training on CIFAR-10 dataset
  - 문제에서 제시한 모델 그대로 구현

- Problem 2: Inception module 구현
  - 문제에서 제시한 모델 그대로 구현
  - Hyper parameter 변경 가능 (e.g. filter 수)

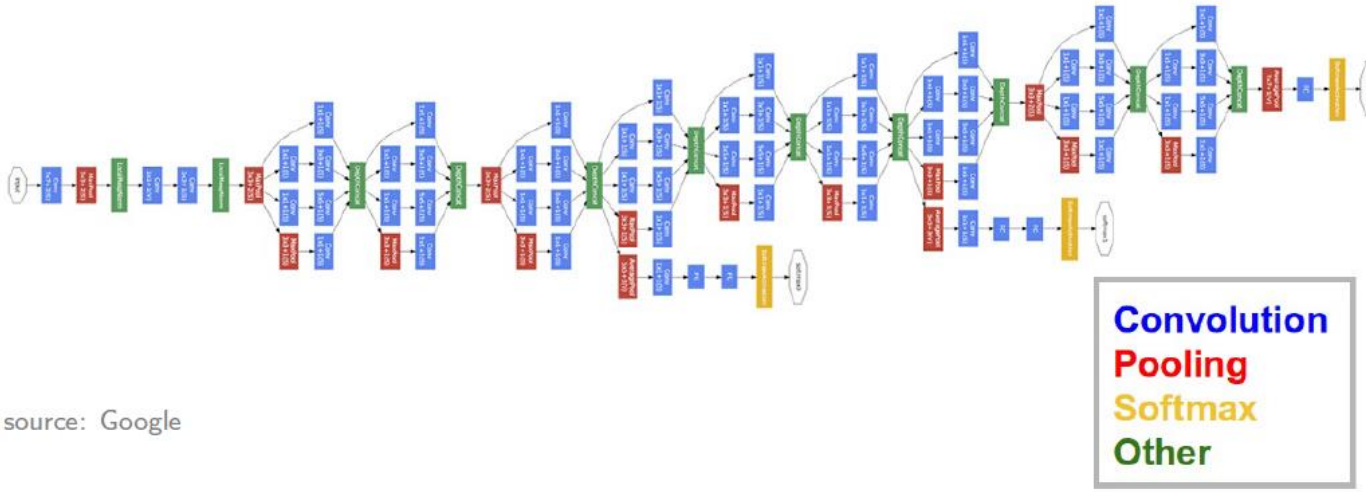
- Problem 3: Inception module을 활용한 CNN model training (CIFAR-10)
  - Test set accuracy >= 70%
  - Test set accuracy 상위 30% 가산점 부여
  - 구현한 모델 설명

CIFAR-10 dataset



- Collection of images used to train machine learning and computer vision algorithms
- Consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class
- There are 50,000 training images and 10,000 test images.
- The classes are completely mutually exclusive(no overlap between truck & automobile).

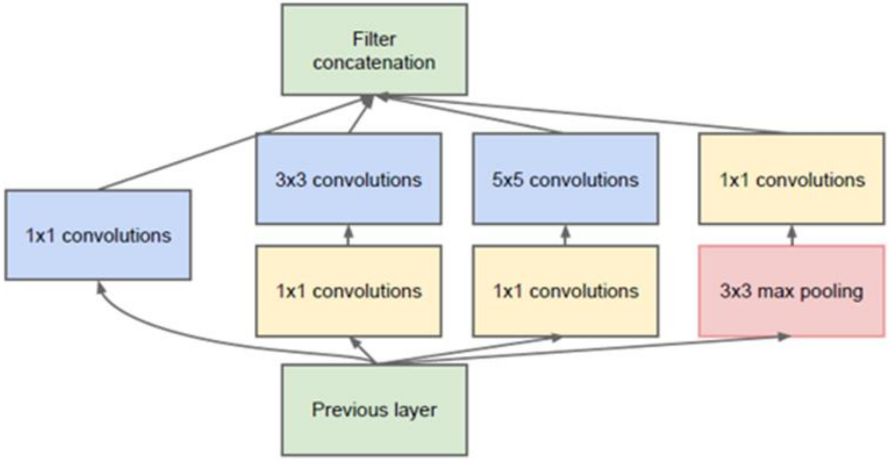
Inception model (a.k.a GoogLeNet)



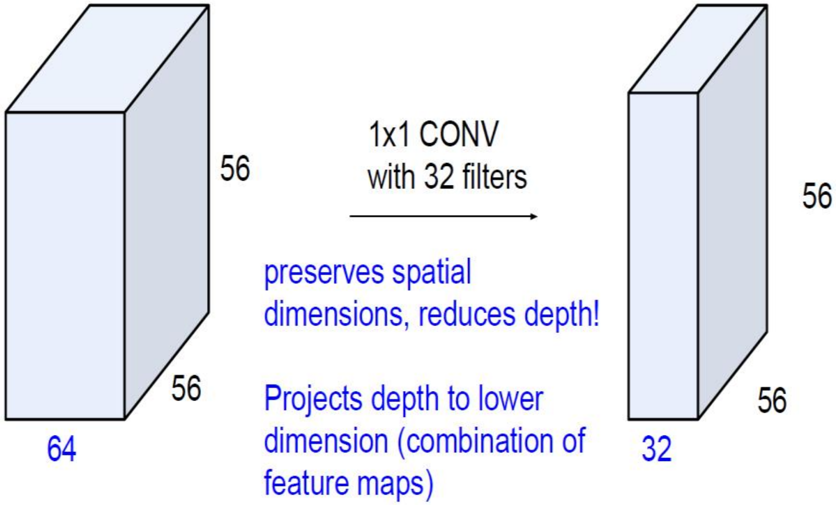
source: Google

- Deeper network with computational efficiency
  - ImageNet Large Scale Visual Recognition Competition (ILSVRC) 2014 winner (6.7% top 5 error)
  - 22 layers with 5 million parameters (12x less than AlexNet *ILSVRC 2012 winner)
  - Efficient “Inception” module

Inception module



Inception module



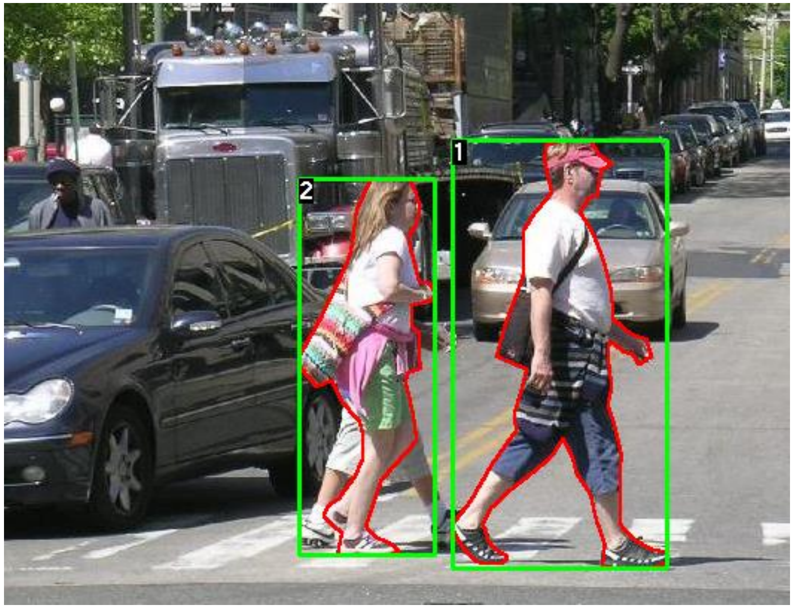
1x1 convolution

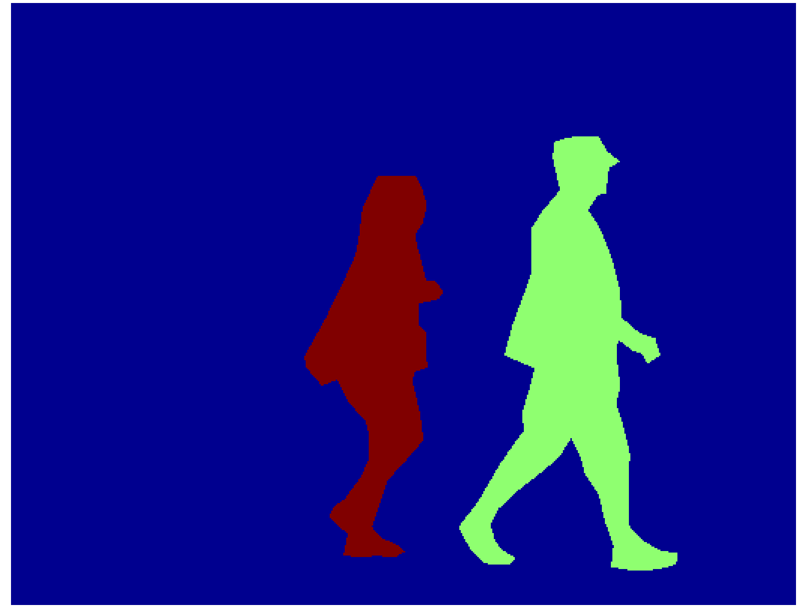
- Local network topology composing the Inception model
  - Apply parallel filter operations on the input from previous layer
  - Multiple filter sizes for convolution (1x1, 3x3, 5x5)
  - 1x1 convolution for dimensionality reduction

Assignment_2-2 Objective (Image segmentation)

- Problem 1: Finetuning model의 code 구현
  - 정의된 dataset과 pre-trained된 모델 활용하여 finetuning 코드 구현
  - 문제에서 제시한 4가지 구현
  - 예측된 segmentation mask(결과값) visualization
  - 구현한 코드 설명

Penn-Fudan dataset

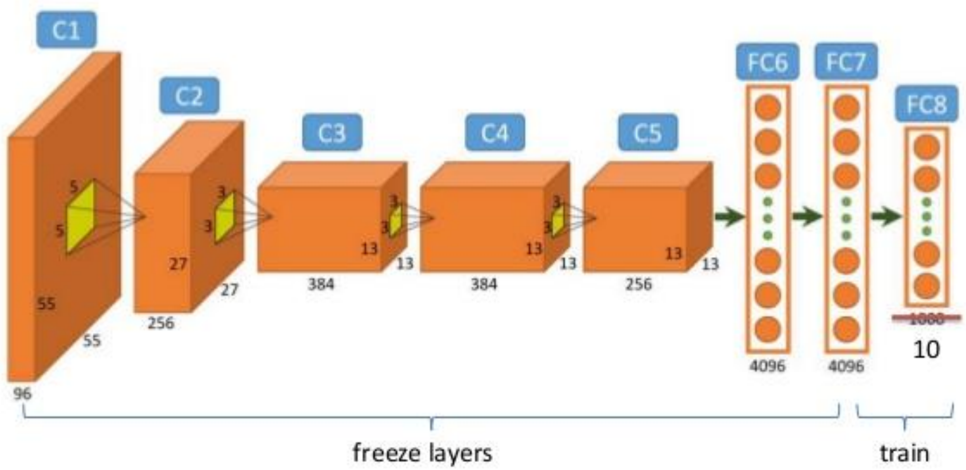




- Image dataset used for pedestrian detection and segmentation
- Consists of 170 images with 345 labeled pedestrians
- The heights of labeled pedestrians fall into [180, 390] pixels

Finetuning

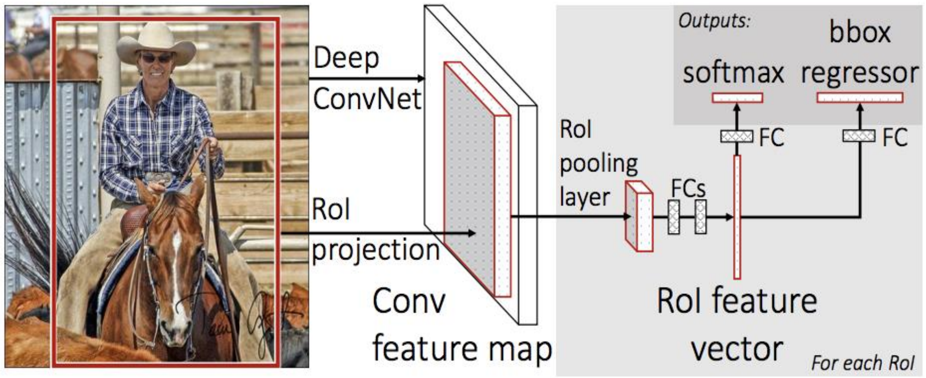
Fine-tuning Pretrained Network



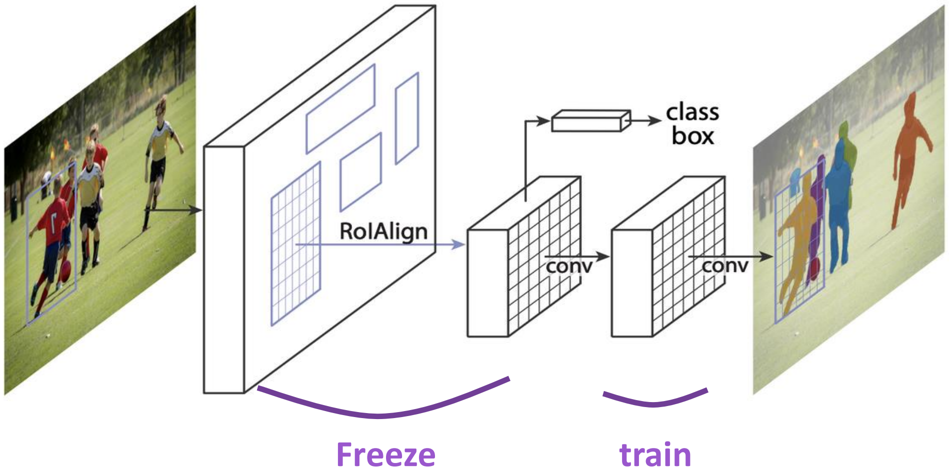
source: https://forums.fast.ai/t/training-layers-independently-and-backpropation/11862

- In practice, very few people train an entire CNN with random initialization.
- Instead, use pre-trained model trained on a very large dataset.
- Then replace and retrain the classifier on top of the ConvNet on the new dataset.

Mask R-CNN



Faster R-CNN



Mask R-CNN

- Faster R-CNN is a model that predicts both bounding box and class scores in image.
- Mask R-CNN adds an extra branch into Faster R-CNN, which also predicts segmentation masks for each instance.
- We will use pre-trained Mask R-CNN (by COCO dataset) and finetune extra branch for segmentation.

How to install assignment files

- 포함된 파일: 3개
  1. Assignment2-1_CNN.ipynb
  2. Assignment2-2_CNN.ipynb
  3. CollectSubmission.sh

- 다운 후 설치 방법
  1. $tar zxvf Assignment2.tar.gz (decompress tar.gz file)
  2. $cd Assignment2
  3. $chmod 755 CollectSubmission.sh (get permission of script file)
  4. $conda activate {가상환경명}
  5. $jupyter notebook

- IPython notebook상에서 과제 수행

Score criteria

- Assignment_2-1 : 60 points + α
  - Problem 1 : Simple CNN model training on CIFAR-10 dataset (15 points)
  - Problem 2 : Inception module 구현
  - Problem 3 : Inception module을 활용한 CNN model training (45 points +α)

- Assignment_2-2 : 40 points
  - Problem 1 : Finetuning model 코드 구현 및 output visualization (40 points)
    - ✓ Evaluation score 상관없이 output visualization에 성공하면 40 points

Output Examples

- Assignment_2-1
- Problem 1 : Simple CNN model training on CIFAR-10 dataset

```
print_accuracy(net, testloader)
```

```
Accuracy of the network on the 10000 test images: 55 %
```

- Problem 3 : Inception module을 활용한 CNN model training (CIFAR-10)

```
[1,  2000] loss: 1.942
[1,  4000] loss: 1.616
[1,  6000] loss: 1.406
[1,  8000] loss: 1.278
[1, 10000] loss: 1.169
[1, 12000] loss: 1.088
[2,  2000] loss: 0.977
[2,  4000] loss: 0.936
[2,  6000] loss: 0.893
[2,  8000] loss: 0.830
[2, 10000] loss: 0.799
[2, 12000] loss: 0.762
Finished Training
Saved Trained Model
Accuracy of the network on the 10000 test images: 73 %
```

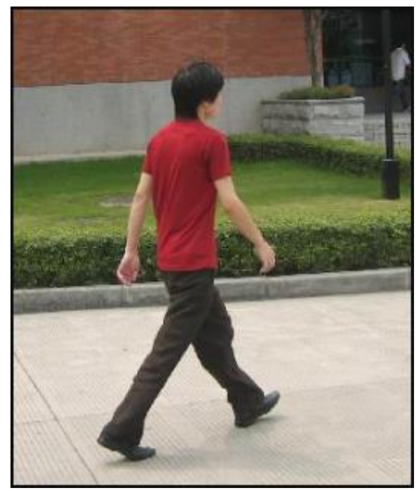
Output Examples

- Assignment_2-2
- Problem 1 : Finetuning model 코드 구현 및 output Visualization

```
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.825
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.990
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.957
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.569
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.835
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.383
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.871
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.871
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.800
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.877
IoU metric: segm
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.763
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.990
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.917
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.465
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.772
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.351
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.809
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.809
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.750
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.814
```

Evaluation result

(ignore -1.000 when area= small)



input image

output image

Important notes

- Due: 10/28(수) 23:59
- PLEASE read the notes on the notebooks carefully
- Google first before mailing TAs

- Submitting your work
  - DO NOT clear the final outputs
  - After you are done all two parts:
    1. $ ./CollectSubmission.sh. 2000-00000 (학번)
    2. Upload the 2000-00000.tar.gz on eTL

- TA email : deeplearning.snu@gmail.com

FAQ_1

- Q : Batch size를 수정해도 되나요?
- A : 네. 모든 hyperparameter는 수정 가능 합니다.

- Q : n3xn3_blue나 n5xn5_blue는 어떤 인자인가요?
- A : n3xn3_blue나 n5xn5_blue는 conv_layer의 output channel 수 입니다. 본인 원하시는 대로 output channel 개수를 조정하셔서 구현하시면 됩니다.

- Q : 주어진 inception module내에서 concat을 해주는데, 이때 tensor 크기를 고려해야 하나요?
- A : 주어진 모듈 내 forward의 리턴값을 보시면, torch.cat으로 y1,y2,y3,y4를 dimension 1에 대해 concat 해줍니다. PyTorch에서는 tensor를 batch x channel x height x width 순으로 정의하기 때문에 코드에서는 y1~4를 channel에 대해 concat 해주는 것입니다. 따라서 channel개수를 제외한 batch, height, width의 크기만 맞춰주시면 됩니다.

FAQ_2

- Q : pth 파일이 무엇인가요?
- A : 학습 후 저장되는 모델 파일입니다. 학습을 완료하시면 자동생성됩니다.

- Q : max iteration 수나 data loader의 batch size, num_worker 수 등도 변경가능한지요?
- A : 가능합니다. 다만 batch size와 num_worker는 실험의 시간적인 면에서만 영향이 있고, 성능에는 영향이 거의 없을 것입니다.
- + epoch도 마찬가지로 기존2에서 더 늘려도 됩니다.

- Q : 코드 구현을 설명하는 부분도 점수에 포함되나요?
- A : ipynb 파일 마지막에 better_net에 관해 설명하는 부분은 copy 확인용이고 점수에 포함되지는 않습니다.
- 다만 Assignment 2-2의 3문제에 대한 답은 점수에 포함됩니다.

- Q : code 처음에 !는 무엇인가요?
- A : ! 뒤에 code는 shell script입니다. 즉, terminal에서 해당 code 돌리기 위해 사용되어집니다.


Thank You!