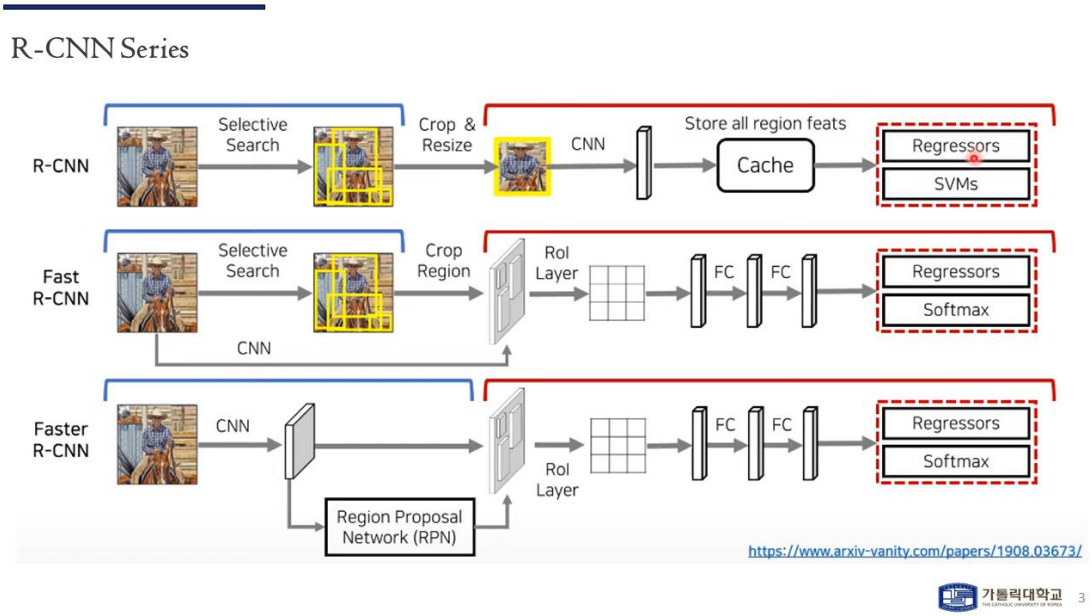
PVANet

PVANet은 Object Detection을 더 효율적으로 하기 위한 네트워크이다.
PVANet Detection을 할 때는 R-CNN 계열에서 제안한 알고리즘(ROI Pooling, RPN)을 사용하므로 R-CNN 계열에 대한 리뷰부터 시작하겠다.



R-CNN은 이미지에 대해 CPU상에서 Selective Search를 진행한다. 그러면 물체가 존재할 법한 위치(박스) 약 2000개를 찾게 되고 그 2000개의 물체를 개별적으로 CNN에 넣어서 Feature를 추출한다. 이후 SVM으로 Classification을 하고 Regressor을 학습해서 박스 위치를 조정한다.

Fast R-CNN도 마찬가지로 Selective Search를 진행한다. 그러나 Feature Map을 뽑기 위해서 CNN을 한 번만 거친다. 이후 ROI Pooling을 통해 각각의 Region들에 대해서 Feature에 대한 정보를 추출한다. CNN 구조를 생각해보면 Feature Map은 Input Image에 대해서 각각의 위치에 대한 정보를 어느정도 보존하고 있기 때문에 이러한 작업이 가능하다. 그리고 Fast R-CNN부터는 Softmax를 통해 각각의 class에 대한 Probability를 구한다.

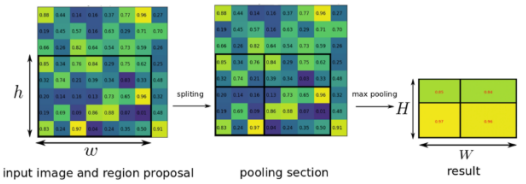
Faster R-CNN은 GPU상에서 Region Proposal을 할 수 있다. 이것은 RPN이라는 네트워크에서 진행한다. RPN은 Feature Map을 보고 어느 곳에 물체가 있을 법한지 예측하는 네트워크라고 생각하면 된다. 기존의 Selective Search보다 훨씬 시간적 장점이 있다. 이후에는 Fast R-CNN의 구조를 따른다.

R-CNN Series

- FAST R-CNN

1. Box를 따로 CNN에 넣는 것이 아니라 원본 영상을 넣는다.
2. AlexNet 대신 VGG-16을 기본 구조로 사용한다.
3. ROI Pooling Layer
4. 분류 시 Soft Max 방식을 사용한다.
5. Box Regressor을 따로 학습하지 않고 동시에 학습한다.

ROI Pooling



FC Layer로 넘김
(제각각인 크기의 ROI 영역이 동일한 크기로 맞춰져서 넘겨짐)

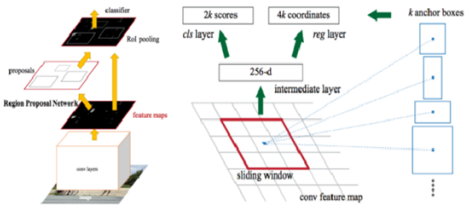
Feature Map의 ROI 영역에 대해 section을 지정 -> Max-pooling을 통해 가장 큰 값을 선택 ex) section을 2x2로 지정하고 pooling
(ROI들은 입력 영상에서 selective search를 통해 찾은 것을 feature map에 적용한 것)

FAST R-CNN의 특징을 나열하면 사진의 (1~5)와 같다. 여기서 주목할 점은 ROI Pooling이다. ROI Pooling은 ROI 영역에 대한 Feature를 찾기 전, Feature Map에 위치한 ROI 영역에 대해 size(H*W)를 통일하는 작업이다. size를 동일하게 만들어야 FC Layer의 입력으로 사용할 수 있다.
그러기 위해 Feature Map의 ROI 영역에 대해 Section을 지정한다. 예를 들어 2x2로 size를 통일하고 싶으면 ROI 영역을 2x2 Section으로 나눈다. 이후에는 Pooling을 해서 각 section마다 가장 큰 값을 선택한다. pooling이 끝나면 각각의 ROI Feature들이 2x2 size로 통일될 것이다.

이후에는 통일된 Feature Map들을 FC Layer로 넘긴다. FC Layer에서 ROI Feature정보를 더 많이 파악하고 classification 및 regression을 진행하는 것이다.

R-CNN Series

- FASTER R-CNN
  - Selective Search로 box를 제안하지 않고 RPN으로 box를 제안한다.



한 점을 중심으로 sliding window를 하는데 매 순간 k개의 box를 proposal한다.
이 중 가장 Object가 있을 것 같은 box만 ROI Pooling Layer로 넘긴다.

RPN에서는 cls loss와 regressor loss가 있다.
전자는 box에 Object가 있을 점수를 측정하고 후자는 박스의 위치 정보가 Object 위치 정보와 얼마나 차이가 나는지를 측정한다.

Faster R-CNN에서 주목할 점은 RPN이다. RPN은 GPU 상에서 Region Proposal을 하는 네트워크이다. 이로써 CPU 상에서 진행된 Region Proposal의 시간 문제를 해결할 수 있었다.

RPN의 아이디어는 이렇다. CNN을 통해 Feature Map이 나오면 한 점(픽셀)을 중심으로 sliding window를 진행한다. sliding window를 하면서 k개의 box를 제안한다. 각 픽셀에 대해 제안된 박스들은 cls layer를 통해 물체가 있을 확률, 없을 확률을 조사한다. (2k scores) 그리고 reg layer를 통해 박스 위치 정보(x,y,w,h)가 object 위치(x,y,w,h)와 얼마나 차이가 있는지 조사한다. (4k scores)

결국 두 레이어에서 cls loss랑 reg loss를 낮추는 학습을 하게 되는데 이러한 학습이 끝나면 물체가 있을 법한 박스들을 찾을 수 있는 것이다.

이렇게 박스들을 찾으면 ROI Pooling을 통해 size를 통일하고 FC Layer로 넘기게 된다.

Design Principles

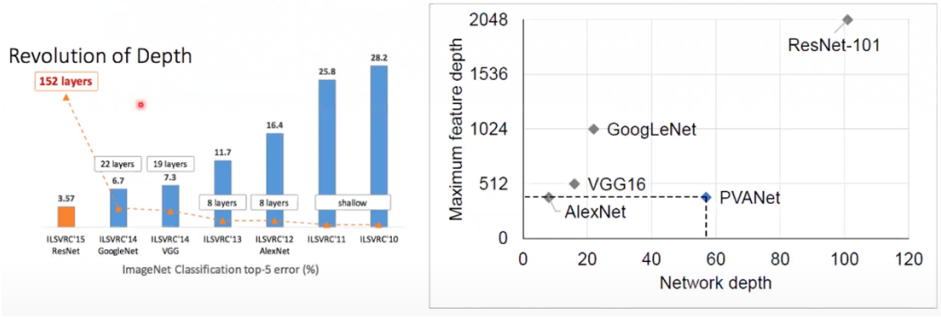
- Deep but Narrow (깊은 레이어 & 적은 필터 수)
- Modified concatenated ReLU (적은 필터 수를 가지고 학습을 잘하도록 함)
- Inception (다양한 스케일의 객체 검출에 효과적)
- Hyper-feature Concatenation (다양한 스케일의 객체 검출에 효과적)

이제부터 본격적인 PVANet 리뷰를 시작하겠다.

PVANet은 Deep하지만 Narrow한 네트워크이다.
Deep하다는 것은 네트워크가 깊다. 즉 레이어가 많다는(깊다는) 의미이고 Narrow하다는 것은 Filter의 수가 적어 Feature Map의 채널 수도 적다는 의미이다. 하지만 Narrow되면서 학습이 잘 안 될 수도 있다. 따라서 Modified concatenated ReLU를 적용하였다.
추가로 Inception과 Hyper-feature Concatenation 개념도 들어가는데 이것은 다양한 스케일의 객체 검출을 위해 사용된다.

Design Principles

- Deep but Narrow
  - 5개의 클래스 라벨을 가지고 훈련된 분류기들 -> Layer가 깊을수록 Error율이 떨어짐 (ResNet:3.57%)
  - Layer가 깊으면서 Feature Map 채널 수가 많은 것을 지적 -> 깊은 레이어, AlexNet과 채널 수는 비슷하게 -> 깊지만 연산량은 적게



ImageNet 대회에서 '5개의 클래스 라벨을 가지고 학습된 분류기들(네트워크들)'이 있었다.

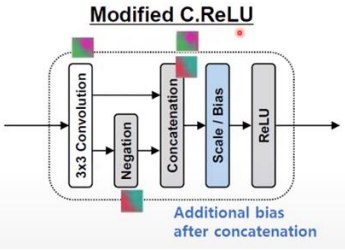
시대가 지나면서 분류기들의 오류율은 점점 작아졌고 레이어는 점점 깊어지는 것을 확인할 수 있다. 대표적으로 ResNet에서 레이어를 152개로 엄청 늘렸더니 오류율이 5% 이하로 떨어지게 되었다. 즉 딥러닝은 deep할수록 성능이 좋다는 것이 확인되었다.

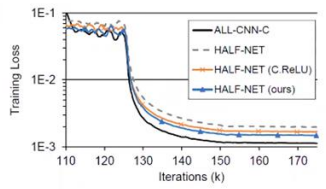
그러나 기존의 네트워크들을 보면 network depth와 feature depth가 비례 관계 형태를 보이게 된다. 즉 deep한 네트워크는 wide하고 (레이어가 많은 네트워크는 Feature Map 채널 수도 많고) deep하지 않은 네트워크는 narrow했다는 것이다. (레이어가 적은 네트워크는 Feature Map 채널 수도 적음)

이 논문에서는 바로 이 점을 지적했다. Deep하면서도 narrow하면 연산량이 적으면서 성능도 좋을 것이라는 의견이었다. 따라서 PVANet은 AlexNet과 채널 수는 비슷하게 하면서 약 60개의 깊은 레이어를 가지도록 하였다.

Design Principles

- Modified Concatenated ReLU
  - 필터 수가 적어지면서 학습하기 어려워진 점을 극복
  - Feature Map 정보(밝기,곡선 등)를 잘 살펴보면 쌍을 이루고 있음 ex) 왼쪽->오른쪽으로 밝아지는 정보, 오른쪽->왼쪽으로 밝아지는 정보
  - Convolution결과를 Negation하고 Concatenation을 하면 채널이 2배가 됨 (필터를 반으로 줄여도 결과는 같음)

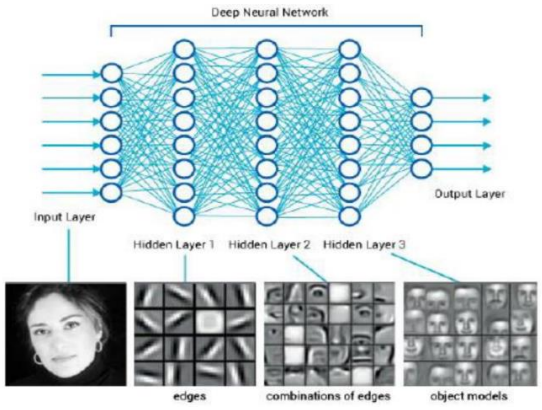




Model	Error (%)	Cost (MMACs)
ALL-CNN-C (our reproduction)	9.83	270
HALF-CNN	10.91	72
HALF-CNN-CReLU	9.99	140
HALF-CNN-mCReLU (ours)	9.84	140

하지만 필터 수가 적어지면(Feature Map의 채널 수가 적어지면) Feature를 적게 찾으므로 학습이 잘 안되는 상황도 일어날 수 있을 것이다.
따라서 Modified Concatenated ReLU라는 기법을 이용하였다. 이 기법은 ReLU를 하기 전에 Convolution한 결과를 Negation(반대로 만들기)하고 Concatenation(합치기)하는 것이다. 이러면 필터 수가 n개여도 Feature Map의 채널 수는 2*N개가 될 수 있다.

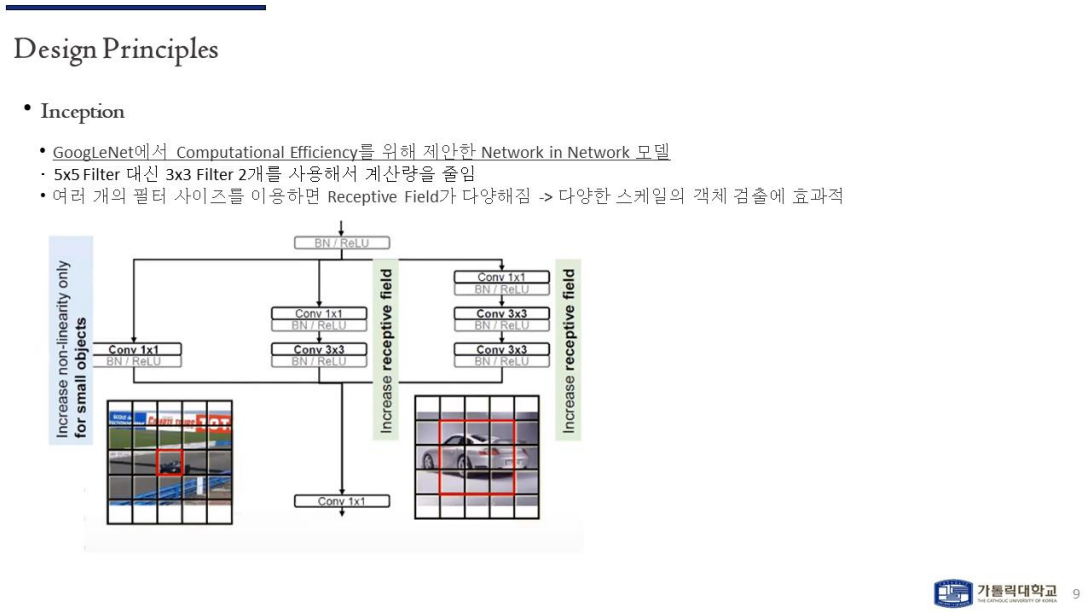
이것이 가능한 이유는 Feature Map의 Feature들이 쌍을 이루고 있기 때문이다.



딥러닝에서 Input과 가까운 층은 밝기, 곡선 등과 같은 low하면서 detatil한 정보를 갖고 Output과 가까운 층은 객체의 형태 등과 같은 high하면서 abstract한 정보를 갖는다.

그런데 각 layer가 가진 정보들, 즉 conv layer를 거쳐서 나온 Feature들의 정보는 쌍을 이루게 된다. 예를 들어 feature map의 첫 번째 채널에 있는 한 노드가 (왼쪽->오른쪽으로 밝아지는 밝기 정보)를 지니고 있다고 하자. 그럼 feature map의 마지막 채널에 있는 한 노드는 (오른쪽->왼쪽으로 밝아지는 밝기 정보)를 가지고 있다는 것이다.

이 의미는 filter를 절반만 사용한 뒤 negation해서 concatenation해도 결과는 filter를 전부 사용한 것과 크게 다르지 않다는 것이다.
따라서 이 논문은 filte 수가 적어도 Modified Concatenated ReLU라는 기법을 사용하면 학습이 잘 될 수 있다고 말한다.
ReLU를 하기 전 bias를 더하는 이유는 Feature들이 불완전한 쌍을 이룰 수도 있으므로 추가했다고 한다. 그리고 이 기법을 실제로 사용한 결과(HALF-NET), 필터 수가 많은 일반적인 CNN과 비슷한 성능을 보였다고 한다.



Inception은 GoogLeNet에서 제안한 Network in Network모델이다. 사진을 잘 보면 하나의 큰 네트워크 안에 여러 conv네트워크가 있는 것을 확인할 수 있다. 이렇게 하면 computational efficiency가 증가된다고 한다.

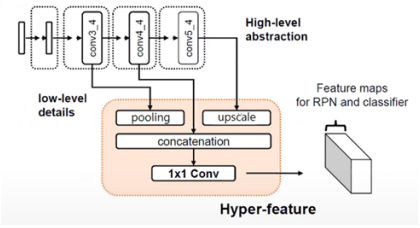
PVANet도 Inception을 사용하지만 차이점이 있다면 Conv 5x5를 Conv 3x3 2개로 바꿔서 receptive field는 유지한 채 parameter 수를 적게 만든 것이다.

또한 Inception에서 보이는 것처럼 여러 개의 필터 사이즈를 이용하면 Receptive Field가 다양해지고 다양한 스케일의 객체 검출에 효과적이라고 한다.

그런데 단순히 여러 개의 필터 사이즈만 이용하면 Computational Complexity가 너무 높아지므로 bottlenect layer(1x1 conv layer)를 추가하였다.

Design Principles

- Hyper-feature Concatenation
  - 세밀한 디테일 + 추상화된 정보를 지닌 Feature를 RPN, Classification에 사용하면 다양한 스케일의 객체 검출이 효과적
  - 세밀한 디테일을 가진 FeatureMap은 downscale, 추상한 정보를 가진 FeatureMap은 upscale -> Concatenation, 중복된 정보는 1x1 Conv로 제거
  - upscale을 할 때는 Bilinear Interpolation을 진행함



Hyper-feature concatenation Multi-scale representation and its combination are proven to be effective in many recent deep learning tasks [6, 12, 13]. Combining fine-grained details with highly abstracted information in the feature extraction layer helps the following region proposal network and classification network detect objects of different scales. However, since the direct concatenation of all abstraction layers may produce redundant information with much higher compute requirement, we need to design the number of different abstraction layers and the layer numbers of abstraction carefully.

fine-grained(세밀)한 디테일과 high한 추상화된 정보를 결합한 Feature Map을 RPN과 Classifier에 넘기면 다양한 스케일의 객체 검출에 도움을 준다고 한다. 따라서 세밀한 디테일을 가진 Feature Map은 pooling을 해서 downscaling하였고 추상화된 정보를 지닌 Feature Map은 upscaling을 한 뒤 중간 Feature Map과 Concatenation하였다. 이렇게 결합을 하면 중복된 정보도 많이 가지게 되는데 1x1 conv를 하면서 제거하였다고 한다. 참고로 upscaling을 할 때는 쌍 선형 보간법을 사용해도 문제가 없었다고 한다.

Overall Structure

- Detection Network
  - 54 Convolutional + 3 Fully connected Layers
  - RPN을 할 때 128개의 채널만 사용해서 계산량을 줄임, Faster R-CNN과 다르게 42 Anchor Box를 이용, classification 시 FC 레이어를 이용

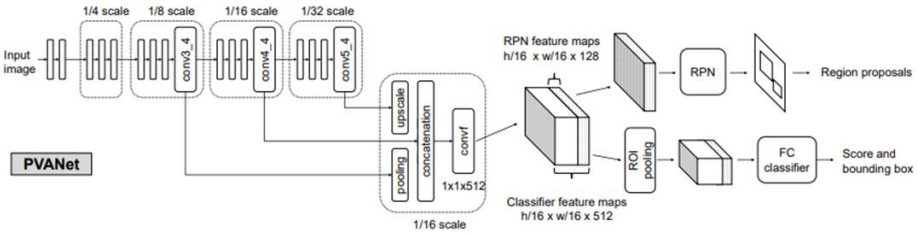


Figure 3: The structure of PVANET detection network.

전체적인 구조에 대한 사진이다. Hyper-feature concatenation한 Feature Map이 RPN과 FC Layer로 가게 된다. 다만 여기서 특이한 점은 RPN으로 갈 때는 128개의 채널만 사용해서 계산량을 줄였

다는 점이다. 또한 Faster R-CNN보다 훨씬 많은 Anchor Box를 이용했다는 것도 특징이다.

Overall Structure

- Details of Network
  - Modified concatenated ReLU는 앞 부분에서만 사용함 (FeatureMap이 쌍을 이루는 것은 Input에 가까운 Layer에서만 나타나는 특징임)

Name	Type	Stride	Output size	Residual	mCReLU #1x1-KxK-1x1	Inception #1x1	#3x3	#5x5	#pool	#out	# params	MAC
conv1_1	7x7 mCReLU	2	528x320x32		X-16-X						2.4K	397M
pool1_1	3x3 max-pool	2	264x160x32									
conv2_1	3x3 mCReLU		264x160x64	O	24-24-64						11K	468M
conv2_2	3x3 mCReLU		264x160x64	O	24-24-64						9.8K	414M
conv2_3	3x3 mCReLU		264x160x64	O	24-24-64						9.8K	414M
conv3_1	3x3 mCReLU	2	132x80x128	O	48-48-128						44K	468M
conv3_2	3x3 mCReLU		132x80x128	O	48-48-128						39K	414M
conv3_3	3x3 mCReLU		132x80x128	O	48-48-128						39K	414M
conv3_4	3x3 mCReLU		132x80x128	O	48-48-128						39K	414M
conv4_1	Inception	2	66x40x256	O		64	48-128	24-48-48	128	256	247K	653M
conv4_2	Inception		66x40x256	O		64	64-128	24-48-48		256	205K	542M
conv4_3	Inception		66x40x256	O		64	64-128	24-48-48		256	205K	542M
conv4_4	Inception		66x40x256	O		64	64-128	24-48-48		256	205K	542M
conv5_1	Inception	2	33x20x384	O		64	96-192	32-64-64	128	384	573K	378M
conv5_2	Inception		33x20x384	O		64	96-192	32-64-64		384	418K	276M
conv5_3	Inception		33x20x384	O		64	96-192	32-64-64		384	418K	276M
conv5_4	Inception		33x20x384	O		64	96-192	32-64-64		384	418K	276M
downscale	3x3 max-pool	2	66x40x128									
upscale	4x4 deconv	2	66x40x384								6.2K	16M
concat	concat		66x40x768									
convf	1x1 conv		66x40x512								393K	1038M
Total											3282K	7942M

전체적인 디테일을 살펴보면 Modified concatenated ReLU가 앞 부분에서만 사용됐는데 이것은 Feature들이 쌍을 이루는 것이 Input과 가까운 Layer에서만 나타나기 때문이라고 한다.

Results

- Classification (ILSVRC 2012)
  - 연산량은 AlexNet이랑 비슷, 결과는 GoogLeNet이랑 비슷

Model	top-1 err. (%)	top-5 err. (%)	Cost (GMAC)
AlexNet [14]	40.7	18.2	0.67
VGG-16 [15]	28.07	9.33	15.3
GoogLeNet [5]	-	9.15	1.5
ResNet-152 [8]	21.43	5.71	11.3
PVANET	27.66	8.84	0.6

- Detection (VOC 2007)
  - mAP > 80, FPS > 20, PVANet을 compress할수록 더 성능이 잘 나옴

Model	Proposals	Recall (%)	mAP (%)	Time (ms)	FPS
PVANET	300	99.2	84.4	48.5	20.6
	200	98.8	84.4	42.2	23.7
	100	97.7	84.0	40.0	25.0
	50	95.9	83.2	26.8	37.3
PVANET+	200	98.8	**84.9**	46.1	21.7
PVANET+ compressed	200	98.8	84.4	31.9	31.3

마지막으로 Classification과 Detection에 대한 결과를 살펴 보았다.

Classification 성능은 GooLeNet이랑 비슷하지만 연산량은 AlexNet이랑 비슷하므로 적은 연산량으로 높은 성능을 보인다는 것이 확인됐다.

Detection은 Region Proposal을 200개 정도 뽑았을 때 Map가 84.4가 되었다고 한다. 그리고

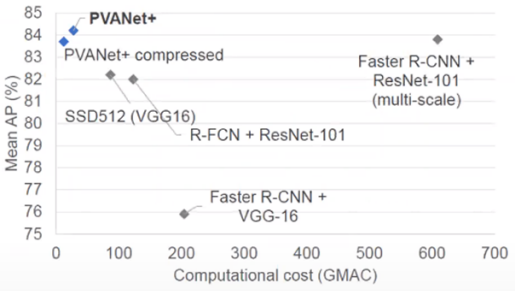
Recall도 약 99%에 해당하니 찾은 객체가 거의 다 맞았다고 받아들일 수 있다. 또한 PVANet을 compress할수록 중복된 정보가 사라지면서 성능이 좋아졌다고 한다.

Results

- Detection Computational cost
  - 80% 이상의 mAP를 가진 모델 중 Computational cost가 압도적으로 낮음
  - 기존에는 알고리즘적인 개선(ROI Pooling, RPN 등), 이 논문은 CNN 네트워크 자체를 개선함

Model name	Cost (GMAC)	Accuracy (mAP, %)
Faster R-CNN + ResNet-101	~600 (multi-scale testing)	83.8
PVANet	27.8	84.2
PVANet (compressed)	12.5	83.7

Better accuracy, 95% less computation



다른 모델들이랑 비교를 하면 Computational cost가 압도적으로 낮으면서 Accuracy는 더 높은 것을 확인할 수 있다.

이 논문은 알고리즘적인 개선보다는 CNN 네트워크 자체를 개선했다는 점에서 메리트가 있고 이 네트워크를 다른 Detection 모델에 적용해도 좋은 성능을 보일 것라 생각한다.

References

- PVANet: Lightweight Deep Neural Networks for Real-time Object Detection
  - https://arxiv.org/pdf/1611.08588.pdf
- PR-033: PVANet: Lightweight Deep Neural Networks for Real-time Object Detection
  - https://www.youtube.com/watch?v=TYDGTnxUGHQ
- 객체 검출(Object Detection) 딥러닝 기술: R-CNN, Fast R-CNN, Faster R-CNN 발전 과정 핵심 요약
  - https://www.youtube.com/watch?v=jqNCdjOB15s&t=1112s


PVANet: Lightweight Deep Neural Networks for Real-time Object Detection

Sanghoon Hong* Byungseok Roh, Kye-Hyeon Kim* Yeongjae Cheon, and Minje Park

Intel Imaging and Camera Technology, Seoul, Korea

Abstract

1 Introduction

Sanghoon Hong* , Byungseok Roh, Kye-Hyeon Kim , Yeongjae Cheon, and Minje Park
(Intel Imaging and Camera Technology, Seoul, Korea)