Fast R-CNN

Ross Girshick(Microsoft Research)

INDEX

Abstract

- R-CNN을 더 발전 시킨 연구.

- R-CNN과 비교해서 PASCAL VOC 2012 데이터셋에서 훈련시간은 9배, 테스트 시간은 213배 더 빠르고, 더 높은 mAP를 달성.

- SPP net과 비교해서는 3배 빠른 훈련시간, 10배 빠른 테스트 시간 더 높은 정확도를 달성.

- 코드 (https://github.com/rbgirshick/fast-rcnn)

Introduction

Classification에 비해 Detection이 작업 파이프 라인이 여러 단계로 구성되는 이유.

- 많은 양의 Object Location candidates(Proposals)를 처리해야함.

- 처리 후의 결과로서 Location에 대한 추정치를 도출해내는데 이 추정치를 정확하게 하기 위해서 후속 처리 필요.

이 연구에서의 목적: 객체의 클래스를 분류하고 위치 파악하는 작업을 한꺼번에 할 수 있는 알고리즘 제안.

Introduction

- R-CNN and SPP net

R-CNN의 단점:

- 훈련 파이프 라인이 여러 단계이다. – CNN, SVMs, BB regressors.
- 훈련 간 시간, 공간적 시간 복잡도가 크다. – Proposal에 대한 특징 벡터를 디스크에 저장.
- 탐지 시간이 느리다. – 각 Proposal에 대해 CNN부터 작업을 개별적으로 수행.

SPP net의 단점:

- 여전히 훈련 파이프 라인이 여러 단계이다. – CNN 연산만 한 번으로 줄임.
- . 여전히 특징 벡터들을 디스크에 저장한다.
- Fine-tuning 시에 컨볼루션 계층들의 가중치를 업데이트 하지 않아 정확도 떨어짐.

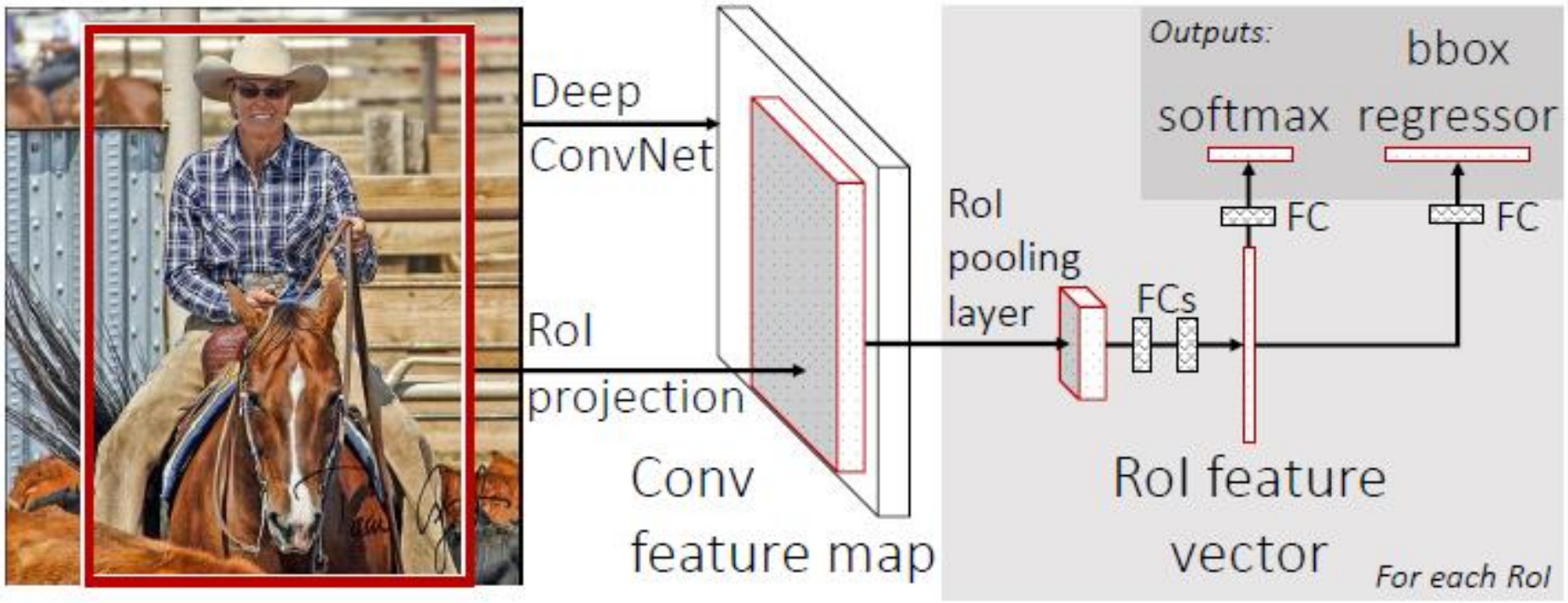
Introduction

- Contribution

이 연구가 기여하는 바

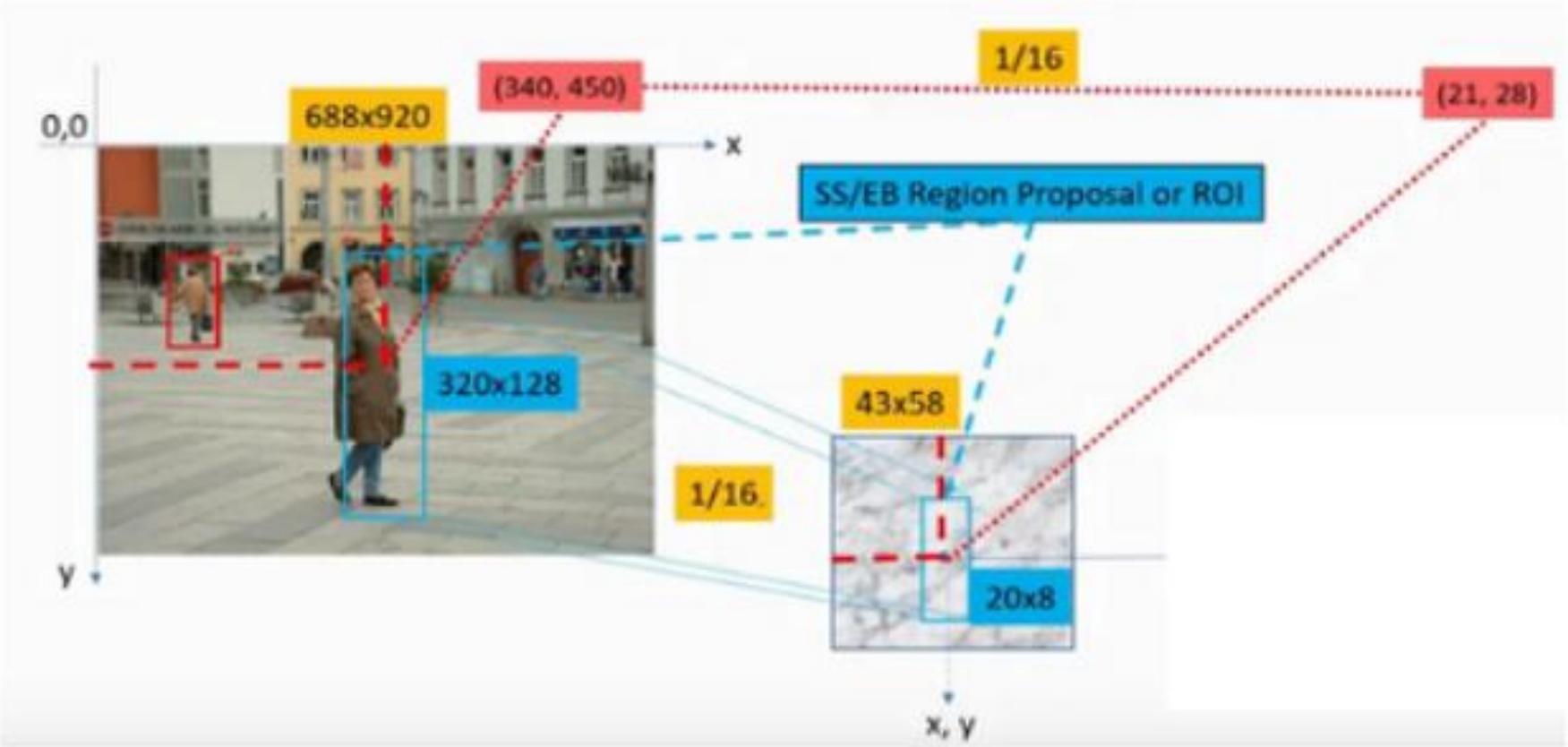
- R-CNN, SPP net보다 향상된 탐지 성능.
- Proposal 생성 과정을 제외하고 Multi-task Loss를 적용한 Single-stage Training.
- 훈련 간 모든 계층의 가중치 갱신 가능.
- 특징 벡터를 디스크에 저장할 필요 없어짐.

Fast R-CNN Architecture and Training



Q: CNN 연산 적용 후에 만들어지는 Feature map과 원본 이미지의 크기가 다르면 원본 이미지에서의 ROI 좌표 정보가 달라지지 않는데 어떻게 할까?

Fast R-CNN Architecture and Training



Arun Mohan - A Review On Fast RCNN
(https://medium.com/datadriveninvestor/review-on-fast-rcnn-202c9eadd23b)

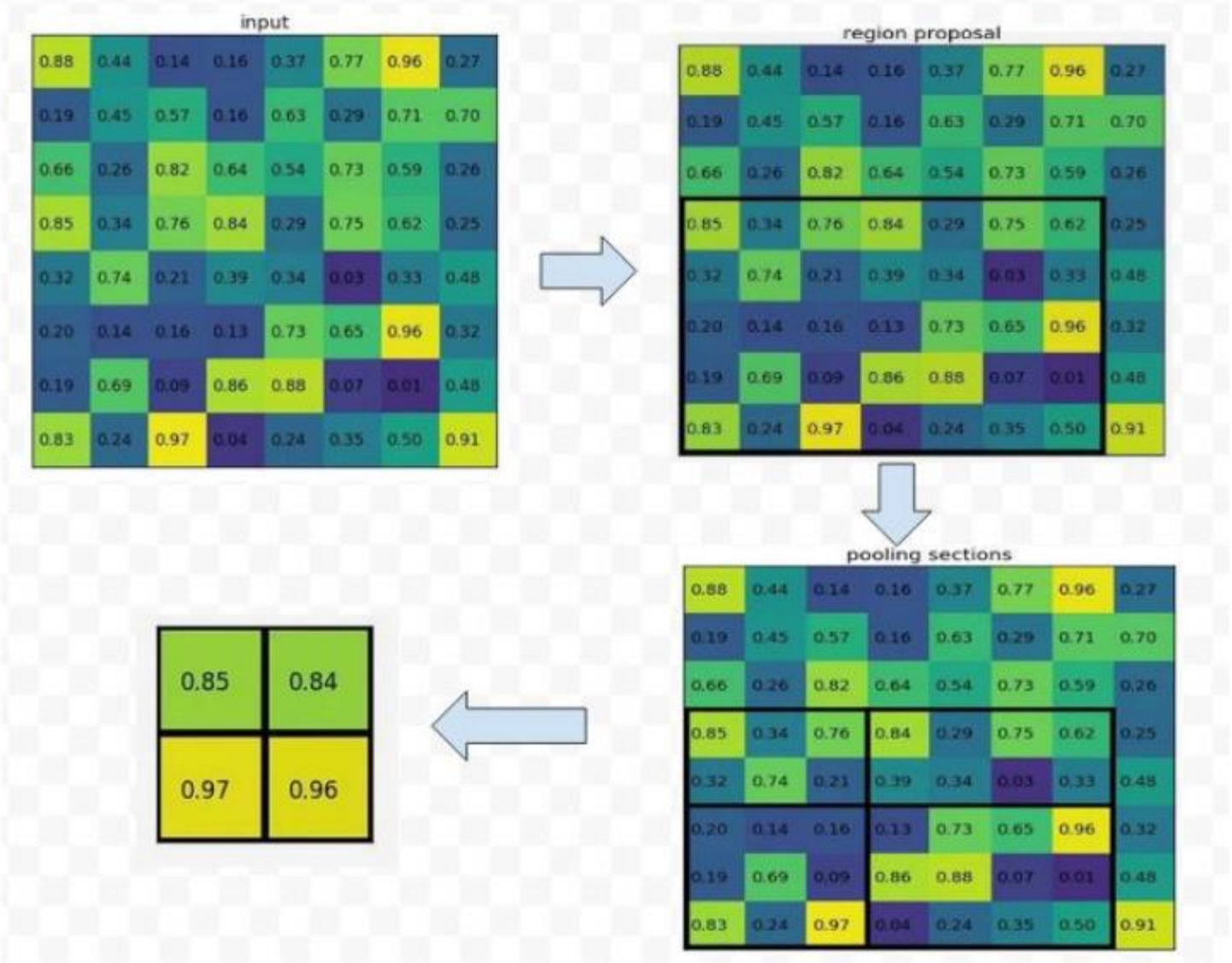
원본 이미지와 특징 맵의 스케일만큼을 지역 후보의 좌표 정보에 곱한다.

Fast R-CNN Architecture and Training

- The ROI Pooling Layer

ROI Pooling Layer에서는 Proposal의 특징을 어떤 고정된 길이의 벡터로 바꾸는 작업을 한다. 이때 임의의 h x w 크기의 ROI를 H x W 그리드로 바꾸는 방법은 각 Cell의 크기를 h/H x w/W로 하는 것이다.

예를 들어서 RoI가 7x5이고 그리드의 크기가 2x2이면 7/2 = 3.5, 5/2 = 2.5이므로 각 Cell의 크기는 2x3, 2x4, 3x3, 3x4가 된다.



Arun Mohan - A Review On Fast RCNN
(https://medium.com/datadriveninvestor/review-on-fast-rcnn-202c9eadd23b)

Fast R-CNN Architecture and Training

- Initializing from Pre-trained Networks

Pre-trained on ImageNet의 Network를 Fast R-CNN에 맞게 초기화 하는 방법

1. 마지막 Max Pooling Layer를 ROI Pooling Layer로 교체
2. 네트워크의 마지막 Fully Connected Layer(with Softmax)를 앞에서 언급한 두 가지 브랜치로 교체 한다.
3. 네트워크가 이미지 리스트와 각 이미지의 ROI 리스트를 입력으로 받을 수 있도록 네트워크를 변경한다.

Fast R-CNN Architecture and Training

- Fine-tuning for Detection – Multi-Task Loss

- Fast R-CNN은 최종적으로 두 가지 종류의 값을 출력한다.

ROI의 K+1 카테고리에 대한 Softmax 확률 분포	ROI의 k클래스 바운딩 박스 OFFSET
$p = (p_0, \ldots, p_K)$	$t^k = (t^k_x, t^k_y, t^k_w, t^k_h)$

- 각 ROI에는 GT class u와 GT BB Regression Target v가 매칭되어 있다.

- 이 연구에서는 Class Loss와 Localization Loss를 다음과 같이 동시에 계산한다.

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(t^u, v), \quad (1)$$

Fast R-CNN Architecture and Training

- Fine-tuning for Detection – Multi-Task Loss

$$L(p, u, t^u, v) = L_{\text{cls}}(p, u) + \lambda[u \geq 1] L_{\text{loc}}(t^u, v), \qquad (1)$$

- 첫번째 항은 Softmax에 의해 계산된 p에 대한 log Loss를 계산한다.

$$L_{\text{cls}}(p, u) = -\log p_u$$

- 두 번째 항은 예측 박스와 GT 박스의 Smooth L1 Loss를 계산한다.

$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{x,y,w,h\}} \text{smooth}_{L_1}(t_i^u - v_i), \qquad (2)$$

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases} \qquad (3)$$

- L1 대신에 L2 손실을 적용할 경우 Gradients Exploding을 신경 써서 Learning Rate를 튜닝해야 한다.
- u는 적어도 하나 클래스가 맞는 박스가 있으면 1, 없으면 0이므로 Background Class에 대한 박스는 두 번째 항의 Loss를 고려하지 않는다.
- λ는 두 종류의 Loss의 균형을 담당하는데 여기서는 1로 설정한다.

Fast R-CNN Architecture and Training

- Fine-tuning for Detection – Mini-Batch Sampling

- Foreground Object Class – GT 박스와 IOU가 적어도 0.5되는 ROI. 전체의 25%정도 된다. (u >= 1)
- Background Object Class – GT 박스와 IOU가 [0.1, 0.5)되는 ROI.
- Hard example mining에서 GT 박스와 IOU가 적어도 0.1정도 되도록 Background Object Class를 추출.
- 각 이미지는 훈련 간에 0.5의 확률로 Horizontally Flipping 될 수 있음.
- 다른 Data augmentation은 수행하지 않음.

Fast R-CNN Architecture and Training

- Fine-tuning for Detection – Back-Propagation through ROI Pooling Layers

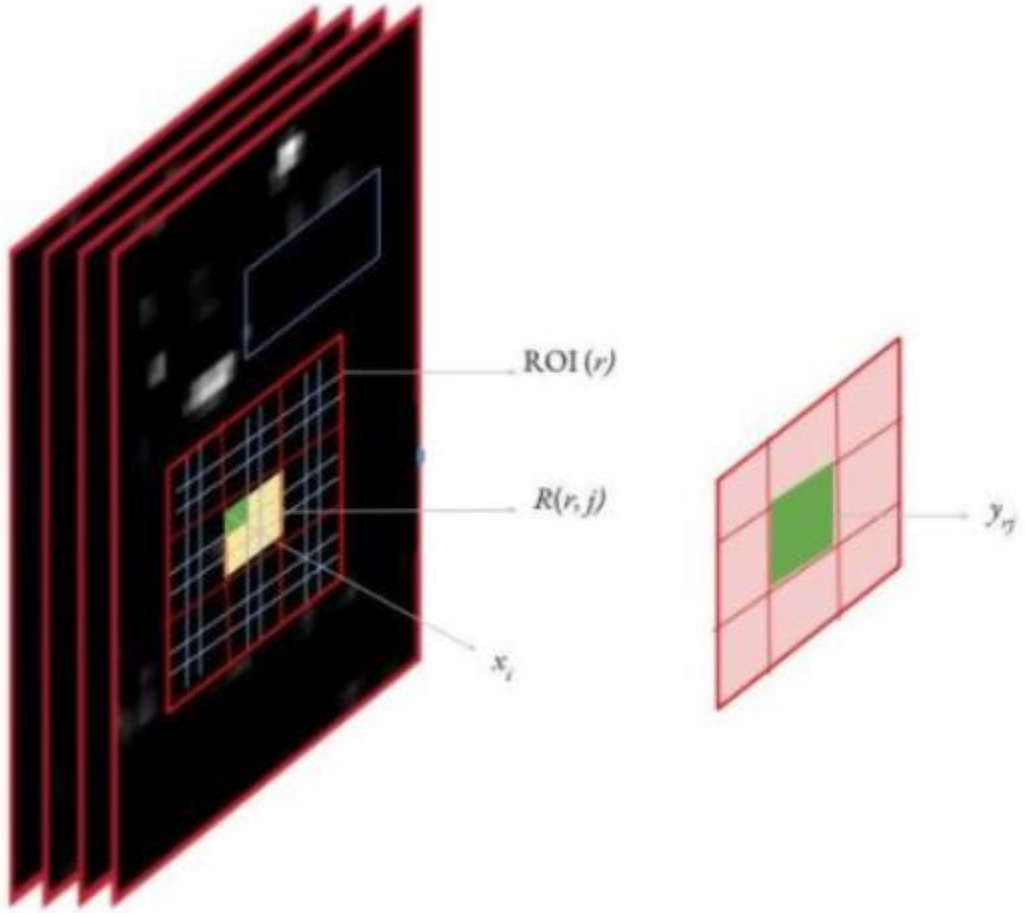
- 역전파시에 모든 계층의 가중치가 갱신 될 수 있도록 ROI Pooling Layer에서 편 미분 값을 계산하는 방법을 구현.

$$\frac{\partial L}{\partial x_i} = \sum_r \sum_j [i = i^*(r,j)] \frac{\partial L}{\partial y_{rj}}. \qquad (4)$$

x_i : ROI에서 하나의 실수 값.
y_ri : r번째 ROI의 j번째 출력 값. 다음과 같이 계산.

$$y_{rj} = x_{i^*(r,j)} \quad i^*(r,j) = \text{argmax}_{i' \in \mathcal{R}(r,j)} x_{i'}$$

여기서 i*(r, j)는 ROI와 Sub Window index j가 주어져 있을 때 Max pooling을 통해서 선택된 최대 값을 가진 특징 맵의 요소의 인덱스를 의미.



Back Prop through RoI Pooling

즉 y_ri를 만들어내는데 유효한 특징 맵에서의 x_i에 대한 편 미분 값을 Chain-rule를 통해서 계산.

염창동 형준 킴 – 갈아 먹는 Object Detection [3] Fast RCNN (https://yeomko.tistory.com/15)

Fast R-CNN Detection

Scale Invariance

객체의 스케일에 영향을 받지 않고 탐지를 할 수 있도록 하는, 두 가지 방법 적용.

- Brute-Force : 말 그대로 정해진 크기로 이미지 스케일 재조정.
- Multi-Scale : Image Pyramid를 통해서 다양한 크기의 입력을 생성.

Pyramid of images

Fast R-CNN Detection

- Image Pyramid를 적용할 때는, 크기별로 인코딩된 이미지 리스트와 그에 따른 R개의 Proposal 리스트를 입력으로 받는다.
- 테스트 시에 R은 거의 2K정도 된다.
- Image Pyramid를 적용시에는 ROI가 거의 224x224가 되도록 한다.
- 각 ROI에 대해 순전파를 마치면 클래스 확률 분포 p와 BB offset과 관련된 값이 생성된다.
- 각 r에 다음과 같이 추정 확률 값을 계산하여 객체 클래스 k만큼 Detection Confidence를 부여한다.
- NMS를 각 클래스마다 독립적으로 수행한다.

Fast R-CNN Detection

Truncated SVD for Faster Detection

$$W \approx U \Sigma_t V^T \qquad (5)$$

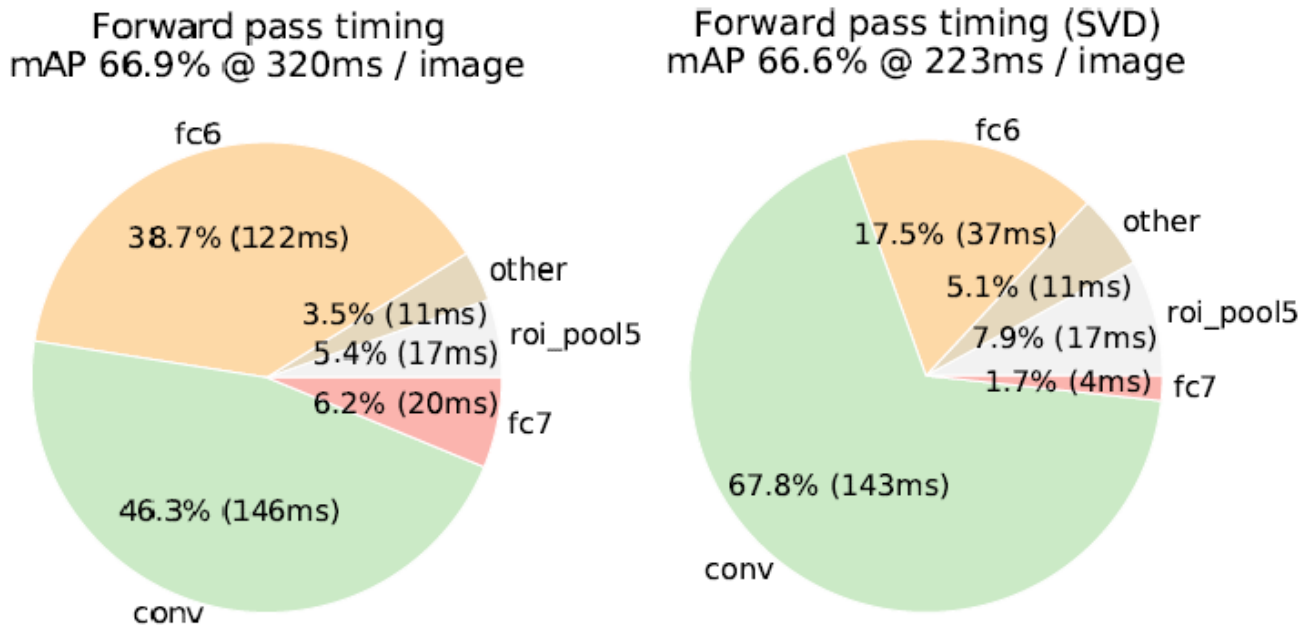


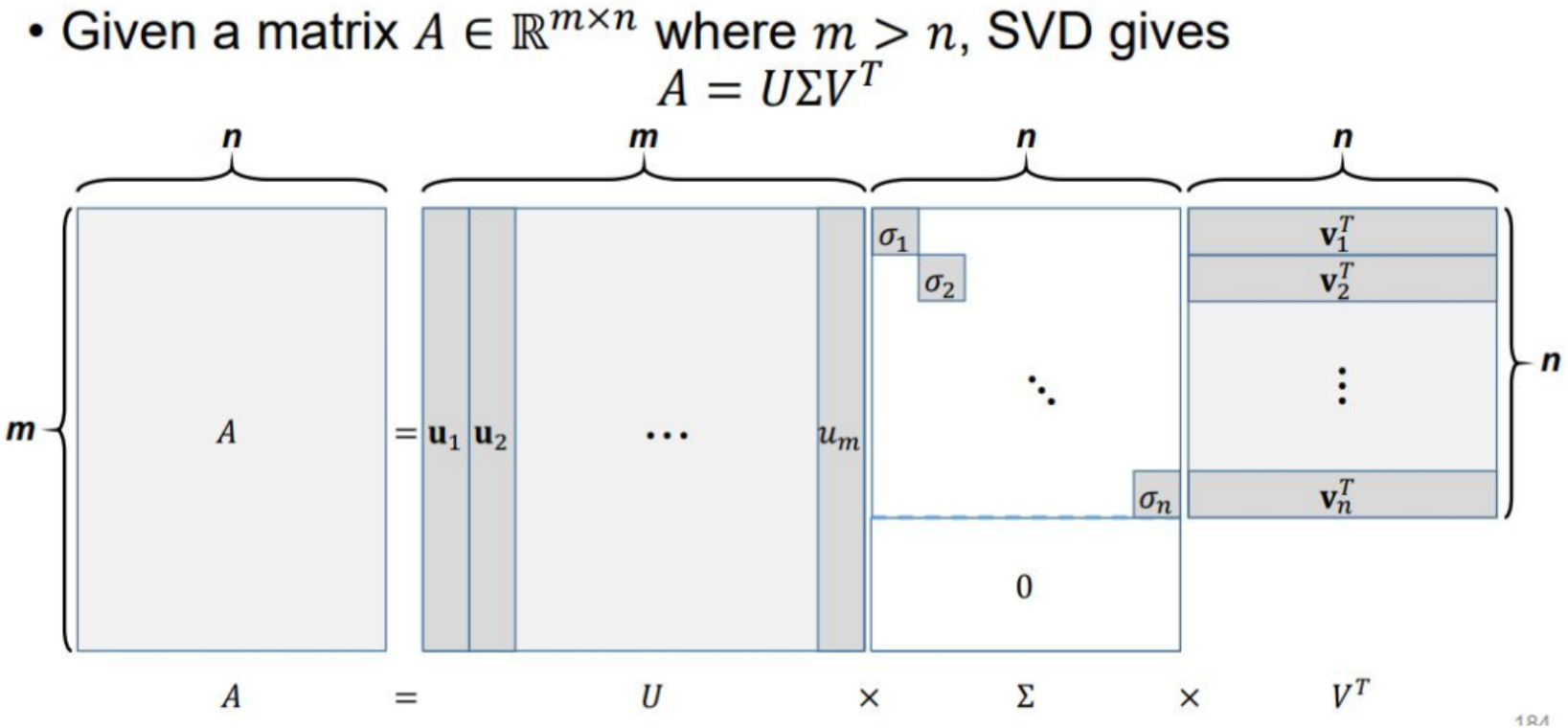
Figure 2. Timing for VGG16 before and after truncated SVD. Before SVD, fully connected layers fc6 and fc7 take 45% of the time.

- 저자들의 분석에 의하면 순전파 시간의 절반을 완전 연결 계층의 연산에 쓴다. 그래서 SVD를 통해서 완전 연결 계층에서의 연산의 시간을 줄이고자 노력했다.
- 완전 연결 계층의 가중치 행렬을 위와 같이 분해하고 층을 두개로 나눈다. 이 층 사이에 비선형성은 생략.

Fast R-CNN Detection

Truncated SVD for Faster Detection

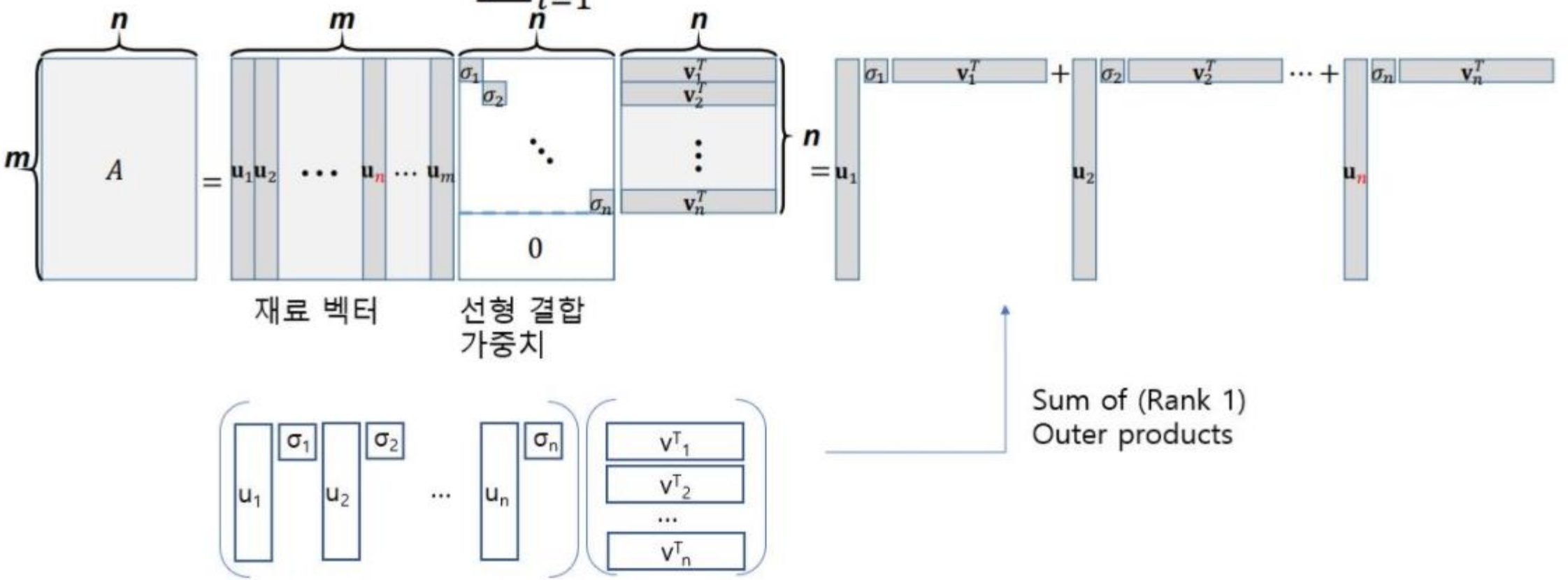
직사각행렬 $A \in \mathbb{R}^{(m \times n)}$가 있고,Orthonormal한 열들로 구성된 $U \in \mathbb{R}^{(m \times m)}$, $V \in \mathbb{R}^{(n \times n)}$가 있어서 각각 A의 열공간과 행공간에 Orthonormal한 기저들을 구성하며 대각 행렬 $\Sigma \in \mathbb{R}^{(m \times n)}$의 0이 아닌 원소들이 내림차 순으로 되어 있을 때 ($\sigma 1 \geq \sigma 2 \geq \cdots \geq \sigma_\min(m,n)$) SVD는 다음과 같이 쓸 수 있다.



Fast R-CNN Detection

Truncated SVD for Faster Detection

$$A = U\Sigma V^T = \sum_{i=1}^{n} \sigma_i \mathbf{u}_i \mathbf{v}_i^T, \quad \text{where } \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$$



위에서 몇 개의 σ을 0으로 만들어버리면 축소된 형태의(그러면서 A와 거의 비슷한) 행렬을 얻을 수 있다.

Main Results

Experimental Setup

- CaffeNet(AlextNet의 변형체)를 Small의 S로 부름.
- S와 깊이는 같으나 넓이가 더 넓은 VGG_CNN_M_1024를 Medium의 M으로 부름.
- Very deep VGG16를 Largest의 L로 부름.
- 모든 실험은 Single-Scale Training and Testing이 원칙.

VOC 2007 Results

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
SPPnet BB [11]†	07 \ diff	73.9	72.3	62.5	51.5	44.4	74.4	73.0	74.4	42.3	73.6	57.7	70.3	74.6	74.3	54.2	34.0	56.4	56.4	67.9	73.5	63.1
R-CNN BB [10]	07	73.4	77.0	63.4	45.4	**44.6**	75.1	78.1	79.8	40.5	73.7	62.2	79.4	78.1	73.1	64.2	**35.6**	66.8	67.2	70.4	**71.1**	66.0
FRCN [ours]	07	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8	66.9
FRCN [ours]	07 \ diff	74.6	**79.0**	68.6	57.0	39.3	79.5	**78.6**	81.9	**48.0**	74.0	67.4	80.5	80.7	74.1	69.6	31.8	67.1	68.4	75.3	65.5	68.1
FRCN [ours]	07+12	**77.0**	78.1	**69.3**	**59.4**	38.3	**81.6**	**78.6**	**86.7**	42.8	**78.8**	**68.9**	**84.7**	**82.0**	**76.6**	**69.9**	31.8	**70.1**	**74.8**	**80.4**	70.4	**70.0**

Table 1. **VOC 2007 test** detection average precision (%). All methods use VGG16. Training set key: **07**: VOC07 trainval, **07 \ diff**: **07** without "difficult" examples, **07+12**: union of **07** and VOC12 trainval. †SPPnet results were prepared by the authors of [11].

Main Results

VOC 2010 and 2012 Results

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
BabyLearning	Prop.	77.7	73.8	62.3	48.8	45.4	67.3	67.0	80.3	41.3	70.8	49.7	79.5	74.7	78.6	64.5	36.0	69.9	55.7	70.4	61.7	63.8
R-CNN BB [10]	12	79.3	72.4	63.1	44.0	44.4	64.6	66.3	84.9	38.8	67.3	48.4	82.3	75.0	76.7	65.7	35.8	66.2	54.8	69.1	58.8	62.9
SegDeepM	12+seg	**82.3**	75.2	67.1	50.7	**49.8**	71.1	69.6	88.2	42.5	71.2	50.0	85.7	76.6	81.8	69.3	**41.5**	**71.9**	62.2	73.2	**64.6**	67.2
FRCN [ours]	12	80.1	74.4	67.7	49.4	41.4	74.2	68.8	87.8	41.9	70.1	50.2	86.1	77.3	81.1	70.4	33.3	67.0	63.3	77.2	60.0	66.1
FRCN [ours]	07++12	82.0	**77.8**	**71.6**	**55.3**	42.4	**77.3**	**71.7**	**89.3**	**44.5**	**72.1**	**53.7**	**87.7**	**80.0**	**82.5**	**72.7**	36.6	68.7	**65.4**	**81.1**	62.7	**68.8**

Table 2. **VOC 2010 test** detection average precision (%). BabyLearning uses a network based on [17]. All other methods use VGG16. Training set key: **12**: VOC12 trainval, **Prop.**: proprietary dataset, **12+seg**: **12** with segmentation annotations, **07++12**: union of VOC07 trainval, VOC07 test, and VOC12 trainval.

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
BabyLearning	Prop.	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6	63.2
NUS_NIN_c2000	Unk.	80.2	73.8	61.9	43.7	**43.0**	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	**38.6**	**68.3**	58.0	68.7	63.3	63.8
R-CNN BB [10]	12	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3	62.4
FRCN [ours]	12	80.3	74.7	66.9	46.9	37.7	73.9	68.6	87.7	41.7	71.1	51.1	86.0	77.8	79.8	69.8	32.1	65.5	63.8	76.4	61.7	65.7
FRCN [ours]	07++12	**82.3**	**78.4**	**70.8**	**52.3**	38.7	**77.8**	**71.6**	**89.3**	**44.2**	**73.0**	**55.0**	**87.5**	**80.5**	**80.8**	**72.0**	35.1	**68.3**	**65.7**	**80.4**	**64.2**	**68.4**

Table 3. **VOC 2012 test** detection average precision (%). BabyLearning and NUS_NIN_c2000 use networks based on [17]. All other methods use VGG16. Training set key: see Table 2, **Unk.**: unknown.

Main Results

Training and Testing Time

	Fast R-CNN			R-CNN			SPPnet
	S	**M**	**L**	**S**	**M**	**L**	†**L**
train time (h)	**1.2**	2.0	9.5	22	28	84	25
train speedup	**18.3**×	14.0×	8.8×	1×	1×	1×	3.4×
test rate (s/im)	0.10	0.15	0.32	9.8	12.1	47.0	2.3
▷ with SVD	**0.06**	0.08	0.22	-	-	-	-
test speedup	98×	80×	146×	1×	1×	1×	20×
▷ with SVD	169×	150×	**213**×	-	-	-	-
VOC07 mAP	57.1	59.2	**66.9**	58.5	60.2	66.0	63.1
▷ with SVD	56.5	58.7	66.6	-	-	-	-

Table 4. Runtime comparison between the same models in Fast R-CNN, R-CNN, and SPPnet. Fast R-CNN uses single-scale mode. SPPnet uses the five scales specified in [11]. †Timing provided by the authors of [11]. Times were measured on an Nvidia K40 GPU.

Main Results

Which Layers to Fine-Tune?

	layers that are fine-tuned in model **L**			SPPnet **L**
	$\geq$ fc6	$\geq$ conv3_1	$\geq$ conv2_1	$\geq$ fc6
VOC07 mAP	61.4	66.9	**67.2**	63.1
test rate (s/im)	**0.32**	**0.32**	**0.32**	2.3

Table 5. Effect of restricting which layers are fine-tuned for VGG16. Fine-tuning $\geq$ fc6 emulates the SPPnet training algorithm [11], but using a single scale. SPPnet **L** results were obtained using five scales, at a significant (7×) speed cost.

SPP net에서는 완전 연결 계층만 Fine-tuning을 해도 충분하다고 했는데 저자들은 이 주장이 아주 깊은 네트워크에서는 맞지 않는다고 생각했다. 그래서 이를 검증하기 위해서 Fast R-CNN에서 컨볼루션 계층을 Freeze를 하고 완전 연결 계층만 Fine-tuning을 한 결과 mAP가 66.9%에서 61.4%로 하락했다.

Design Evaluation

Does Multi-Task Training Help?

	S				M				L			
multi-task training?		✓		✓		✓		✓		✓		✓
stage-wise training?			✓				✓				✓	
test-time bbox reg?			✓	✓			✓	✓			✓	✓
VOC07 mAP	52.2	53.3	54.6	**57.1**	54.7	55.5	56.6	**59.2**	62.6	63.4	64.0	**66.9**

Table 6. Multi-task training (forth column per group) improves mAP over piecewise training (third column per group).

End-To-End로 Jointly하게 각 네트워크의 컴포넌트들을 훈련시키는 것이 정말로 유효한 가를 실험.

결과적으로 따로 훈련 시키는 것보다 더 효율적임이 실험을 통해서 드러남.

Design Evaluation

Scale Invariance: To Brute Force or Finesse?

	SPPnet **ZF**		**S**		**M**		**L**
scales	1	5	1	5	1	5	1
test rate (s/im)	0.14	0.38	**0.10**	0.39	0.15	0.64	0.32
VOC07 mAP	58.0	59.2	57.1	58.4	59.2	60.7	**66.9**

Table 7. Multi-scale vs. single scale. SPPnet **ZF** (similar to model **S**) results are from [11]. Larger networks with a single-scale offer the best speed / accuracy tradeoff. (**L** cannot use multi-scale in our implementation due to GPU memory constraints.)

- Brute Force한 방법으로 스케일을 정해 놓고 하는 방법과 Image pyramid를 통해서 여러 스케일로 하는 방법의 mAP는 큰 차이가 없으나 시간에서 큰 차이가 발생함.
- 따라서 Map는 조금 떨어져도 시간이 짧은 Singe-Scale의 Brute Force한 방법으로 실험 진행.

Design Evaluation

Do we need more Training Data?

- 더 많은 데이터로 더 많은 시간을 들여 학습 했을 때, mAP가 상승하는 것을 확인함.

Do SVMs outperform Softmax?

method	classifier	S	M	L
R-CNN [9, 10]	SVM	**58.5**	**60.2**	66.0
FRCN [ours]	SVM	56.3	58.7	66.8
FRCN [ours]	softmax	57.1	59.2	**66.9**

Table 8. Fast R-CNN with softmax vs. SVM (VOC07 mAP).

- Fast R-CNN에서 SVM + Hard negative mining의 방법과 Softmax 방법 사이에 큰 차이는 없으나 Softmax는 네트워크를 Jointly하게 학습시킬 수 있으므로 Softmax 방법 적용.

Design Evaluation

Are more Proposals Always Better?

· 더 많은 Proposal이 성능을 개선시키는 확인하기 위한 실험.

· 파란색 실선을 보면 SS에서 Proposal 수를 늘리면 성능이 조금 오르다가 오히려 떨어지는 것을 알 수 있다.

· Object proposal의 Quality를 측정하는 방법으로 Average Recall을 많이 사용한다. AR은 mAP와 Proposal의 수가 이미지 당 고정되어 있을 때, 상관 관계가 높은 것으로 알려져 있는데 Proposal의 수가 다양해짐에 따라 상관 관계가 거의 없는 것을 확인할 수 있다(붉은 실선과 파란색 선들의 추세).

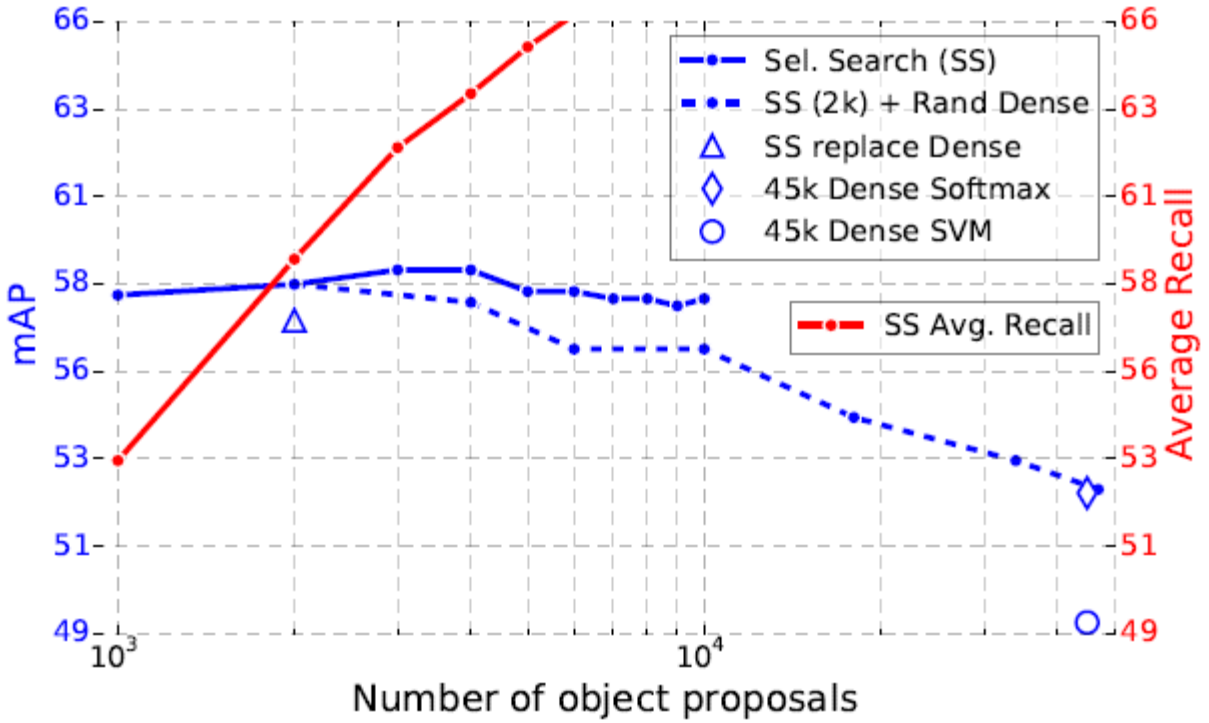


Figure 3. VOC07 test mAP and AR for various proposal schemes.

· 더 많은 Dense box들을 추가할 경우, SS box들을 사용할 때보다 성능이 많이 떨어지는 것을 알 수있다(파란색 점선).

Design Evaluation

Preliminary MS COCO Results

MS COCO 데이터셋에 적용한 Fast R-CNN의 성능은 다음과 같다.

· The PASCAL-style mAP is 35.9%; the new COCO-style AP, which also averages over IoU thresholds, is 19.7%.

Conclusion

저자들은 R-CNN과 SPP net에서의 업데이트 버전인 Fast R-CNN을 제시했다. 실험을 통해서 Sparse object proposal이 탐지 성능을 높이는데 더 좋음을 확인했다.