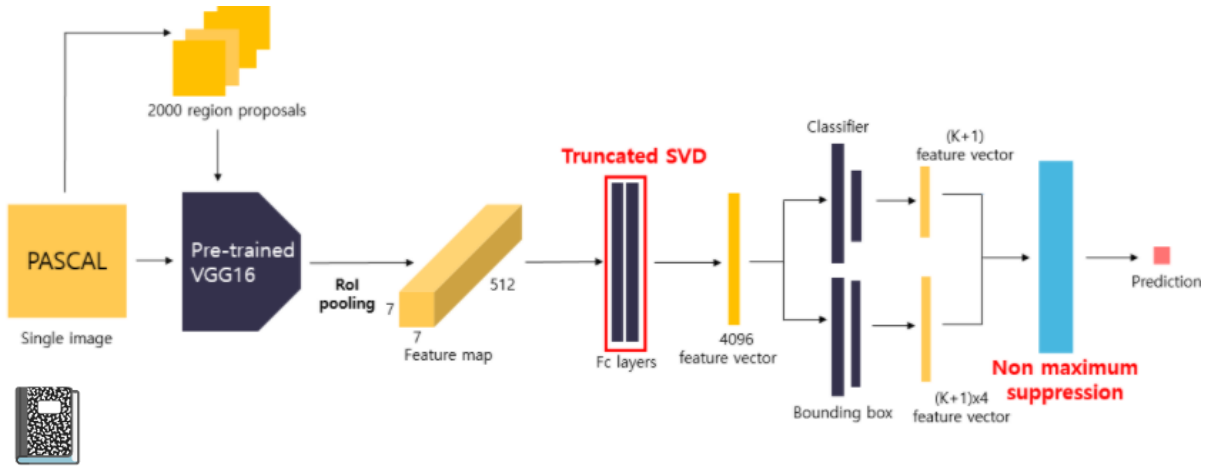


Faster RCNN Prior Learning (Fast R-CNN)

Faster RCNN

- Fast R-CNN은 R-CNN과 SPPNet의 단점을 개선한 모델
- end-to-end learning이 가능하며, 2000개의 proposals이 모두 CNN에 통과하지 않도록 구조를 개선하여 detecting 속도를 높임

R-CNN의 단점

1. 학습이 여러 단계로 나뉘어져 있음
    - R-CNN은 3가지 단계의 학습 과정을 거쳐야 함
        - CNN fine-tuning
        - SVM fine-tuning
        - learn bounding-box regression

2. 학습하는데에 시가닝 오래 걸리고 메모리 공간도 많이 차지
    - SVM과 bounding-box regression은 각 이미지에서 각 Proposal로 추출된 특징으로 학습되기 때문임

3. 느림
    - test time에서 각 이미지에서 각 Proposal로부터 특징이 추출
    - R-CNN은 Proposal을 약 2000개 제안하고 모든 Proposal을 CNN에 전달

      → 많은 시간이 소요 됨

SPPNet의 단점

1. R-CNN과 마찬가지로 학습이 여러 단계에 걸쳐 이루어짐

   1) fine-tuning network

   2) training SVM

   3) fitting bounding-box regressor

2. fc layer만 fine-tuning 가능함
   - Pre-trained된 CNN을 업데이트할 수 없기 때문에 정확도를 제한함

Fast R-CNN의 장점

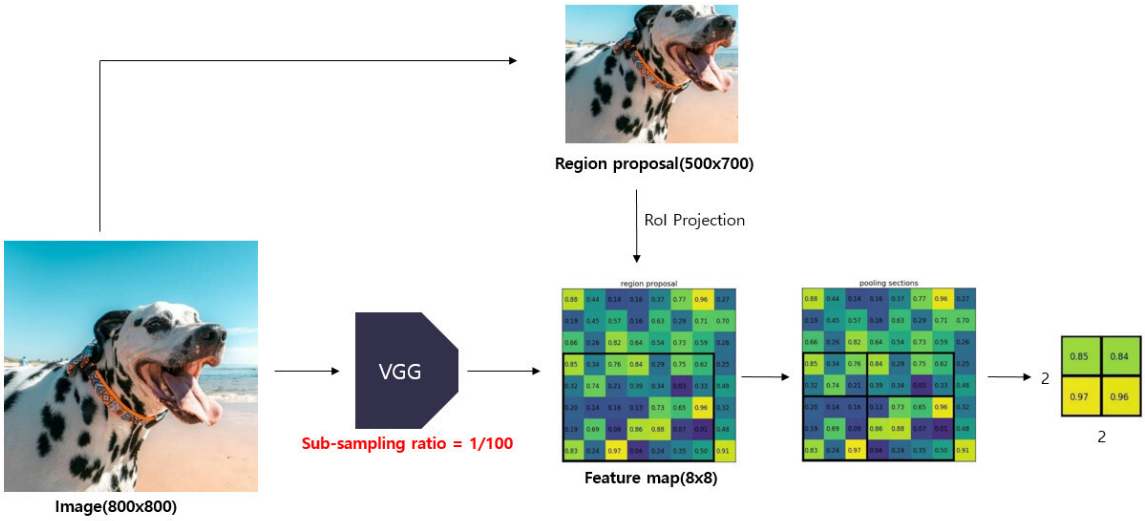
1. R-CNN과 SPPnet보다 높은 mAP을 달성함
2. multi-task loss를 사용하여 학습이 single-stage로 진행
3. 모든 network layers를 업데이트 할 수 있음
4. 특징을 저장하기 위한 추가적인 저장 공간이 요구되지 않음

Fast R-CNN Main Ideas

1. ROI(Region of Interest) Pooling

- **RoI(Region of Interest) pooling**은 feature map에서 region proposals에 해당하는 **관심 영역(Region of Interest)**을 지정한 크기의 grid로 나눈 후 max pooling을 수행하는 방법
- 각 channel별로 독립적으로 수행하며, 이 같은 방법을 통해 **고정된 크기의 feature map을 출력하는 것이 가능**

ex)



1. 먼저 원본 이미지를 CNN 모델에 통과시켜 feature map을 얻음
   - **800x800** 크기의 이미지를 VGG 모델에 입력하여 **8x8** 크기의 feature map을 얻음
   - **sub-sampling ratio = 1/100**이라고 할 수 있음 (여기서 말하는 subsampling은 pooling을 거치는 과정을 의미)

2. 동시에 원본 이미지에 대하여 Selective search 알고리즘을 적용하여 region proposals를 얻음
   - 원본 이미지에 Selective search 알고리즘을 적용하여 **500x700** 크기의 region proposal을 얻음

3. 이제 feature map에서 각 region proposals에 해당하는 영역을 추출

   → 이 과정은 **RoI Projection**을 통해 가능함.

   → Selective search를 통해 얻은 region proposals는 sub-sampling 과정을 거치지 않은 반면, 원본 이미지의 feature map은 sub-sampling 과정을 여러 번 거쳐 크기가 작아졌음.

   → **작아진 feature map에서 region proposals이 encode(표현)하고 있는 부분을 찾기 위해 작아진 feature map에 맞게 region proposals를 투영해주는 과정** 이 필요

   → 이는 region proposal의 크기와 중심 좌표를 **sub sampling ratio** 에 맞게 변경시켜줌으로써 가능

   - Region proposal의 중심점 좌표, width, height와 sub-sampling ratio를 활용하여 feature map으로 투영시킴
   - feature map에서 region proposal에 해당하는 **5x7** 영역을 추출

4. 추출한 RoI feature map을 지정한 **sub-window의 크기**에 맞게 grid로 나눠줌
   - 추출한 5x7 크기의 영역을 지정한 **2x2** 크기에 맞게 grid를 나눠줌

5. grid의 각 셀에 대하여 max pooling을 수행하여 고정된 크기의 feature map을 얻음
   - 각 grid 셀마다 max pooling을 수행하여 **2x2** 크기의 feature map을 얻음

⇒ 미리 지정한 크기의 sub-window에서 max pooling을 수행하다보니 **region proposal의 크기가 서로 달라도 고정된 크기의 feature map**을 얻을 수 있음

Multi-task loss

- Fast R-CNN 모델에서는 feature vector를 **multi-task loss**를 사용하여 Classifier와 Bounding box regressior을 동시에 학습시킴
- 각각의 RoI(=region proposal)에 대하여 multi task loss를 사용하여 학습
- 이처럼 두 모델을 한번에 학습시키기 때문에, R-CNN 모델과 같이 **각 모델을 독립적으로 학습시켜야 하는 번거로움이 없다는 장점**이 있음

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

- $P = (p0, ..., P_k)$: (k+1)개의 class score
- u: ground truth class score
- $t^u = (t^u_x, t^u_y, t^u_w, t^u_h)$: 예측한 bounding box 좌표를 조정하는 값
- $v = (v_x, v_y, v_w, v_h)$: 실제 bounding box 좌표값

$L_{cls}(p, u) = -logp_u$: classification loss(Logloss)
$L_ioc(t^u, v) = \Sigma_{i \in \{x,y,w,h\}} smooth_{L_1}(t^u_i - v_i)$: regression loss(Smooth L1 loss)

$smooth_{L_1}(t^u_i - v_i) = \left\{ \begin{matrix} 0.5x^2, if|x|<1 \\ |x|-0.5, otherwise \end{matrix} \right\}$

λ: 두 loss사이의 가중치를 조정하는 balacing hyperparameter

- K개의 class를 분류한다고할 때, 배경을 포함한 (K+1)개의 class에 대하여 Classifier를 학습시켜줘야 함
- u는 positive sample인 경우 1, negative sample인 경우 0으로 설정되는 **index parameter**
- **L1 loss**는 R-CNN, SPPnets에서 사용한 L2 loss에 비행 outlier에 덜 민감하다는 장점이 있음
- λ=1 로 사용
- multi task loss는 0.8~1.1% mAP를 상승시키는 효과가 있음

Hierarchical Sampling

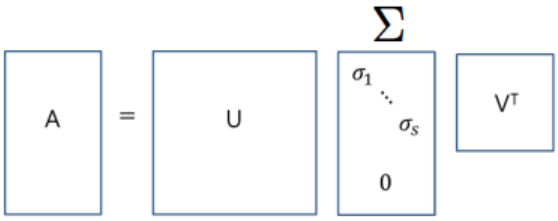
- R-CNN 모델은 학습 시 region proposal이 서로 다른 이미지에서 추출되고, 이로 인해 학습 시 연산을 공유할 수 없다는 단점이 있음

- 논문의 저자는 학습 시 **feature sharing**을 가능하게 하는 **Hierarchical sampling** 방법을 제시
- SGD mini-batch를 구성할 때 N개의 이미지를 sampling하고, 총 R개의 region proposal을 사용한다고 할 때, 각 이미지로부터 R/N개의 region proposals를 sampling하는 방법
- 이를 통해 같은 이미지에서 추출된 region proposals끼리는 forward, backward propogation 시, **연산과 메모리를 공유할 수 있음**

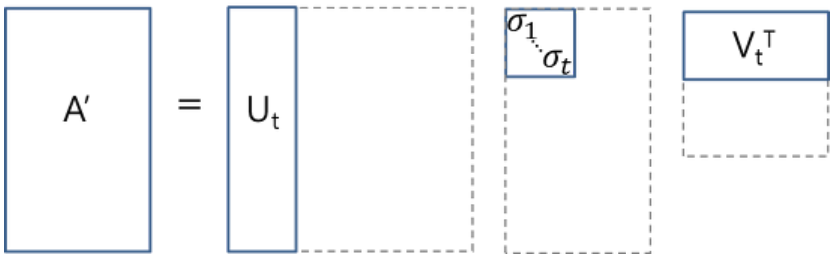
- 논문에서는 학습 시, N=2, R=128로 설정하여, 서로 다른 2장의 이미지에서 각각 64개의 region proposals를 sampling하여 mini-batch를 구성
- 각 이미지의 region proposals 중 25%(=16장)는 ground truth와의 IoU 값이 0.5 이상인 sample을 추출하고, 나머지(75%, 48장)에 대해서는 IoU 값이 0.1~0.5 사이의 sample을 추출
- **전자의 경우 positive sample로, 위에서 정의한 multi-task loss의 u=1u=1이며, 후자는 u=0u=0인 경우라고 할 수 있음**

Truncated SVD

- Fast R-CNN 모델은 detection 시, RoI를 처리할 때 fc layer에서 많은 시간을 잡아먹음
- 논문에서는 detection 시간을 감소시키기 위해 **Truncated SVD(Singular Vector Decomposition)**을 통해 fc layer를 압축하는 방법을 제시



- 행렬 A를 $m \times m$ 크기인 U, $m \times m$ 크기인 Σ, $n \times n$ 크기인 V^T로 특입값을 분해(SVD)하는 것을 FULL SVD(**Singular Vector Decomposition)**라고 함
- 하지만 실제로 이처럼 Full SVD를 하는 경우는 드물며, Truncated SVD와 같이 분해된 행렬 중 일부분만을 활용하는 reduced SVD를 일반적으로 많이 사용한다고함



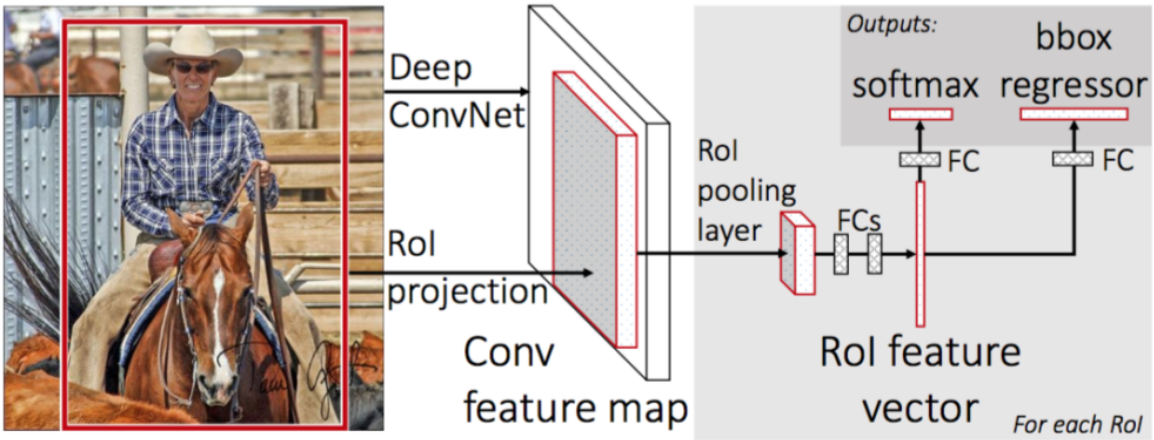
- Trucated SVD는 Σ의 비대각 부분과 데각 원소 중 특이값이 0인 부분을 모두 제거하고, Σ에 대응되는 U,V 원소도 함께 제거하여 차원을 줄이는 형태
- U_t의 크기는 $m \times t$이며, Σ_t의 크기는 $t \times t$, 그리고 V^t의 크기는 $t \times n$임
- 행렬 A를 상당히 근사하는 것이 가능

$$W \approx U\Sigma_t V^T$$

- fc layer의 가중치 행렬이 $W(= u \times u)$라고 할 때, Trucated SVD를 통해 위와 같이 근사하는 것이 가능
- 파라미터 수를 $u \times u$에서 t(u+u)로 감소시키는 것이 가능
- Trucated SVD를 fc layer의 가중치 행렬 W에 적용하면, fc layer는 두개의 fc layer로 나눠지게 됨
  1. fc layer는 $\Sigma_t V^T$가중치 행렬
  2. fc layer는 U 가중치 행렬
- 이를 통해 네트워크를 효율적으로 압축하는 것이 가능

⇒ 논문의 저자는 **Truncated SVD를 통해 detection 시간이 30% 정도 감소**되었다고 말함

Fast R-CNN 구조와 Training



Fast R-CNN 작동 방식

1. Selective Search로 region proposals를 얻음
2. 전체 이미지가 CNN을 통과하여 feature map을 얻음
3. region proposal는 feature map에 projection 되어 RoI를 생성

4. RoI pooling layer는 feature map에 생성된 RoI으로부터 고정된 길이의 특징을 추출

5. 추출된 고정된 길이의 특징은 fc layer에 전달됩니다. 그리고 fc layer는 마지막에 두 output layer로 갈라짐

(1) 첫 번째 output layer는 confidence를 지닌 K개의 class를 예측합니다.

(2) 두 번째 output layer는 각 K class에 대하여 4개 값을 출력합니다. 4개 값은 bounding box regressor를 거쳐 K class의 바운딩 박스 좌표가 됩니다.

1) Initializing pre-trained network

- feature map을 추출하기 위해 **VGG16** 모델을 사용합

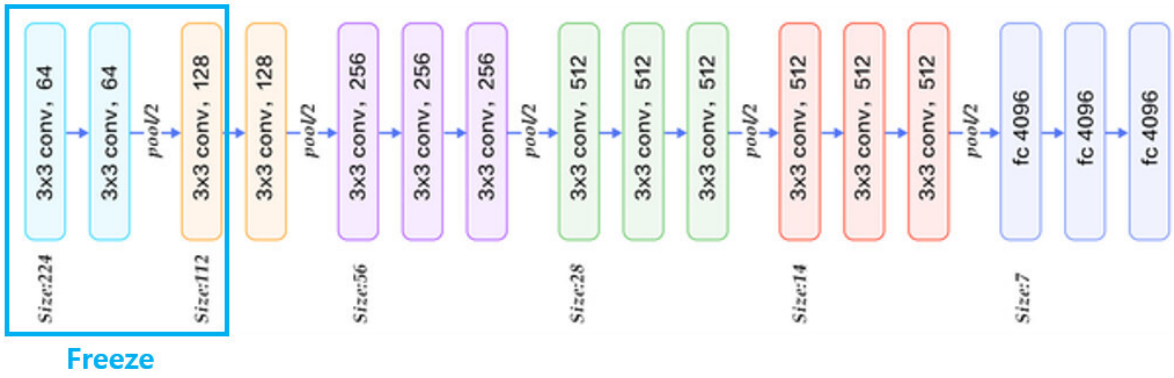
→ 먼저 네트워크를 detection task에 맞게 변형시켜주는 과정이 필요

1. VGG16 모델의 마지막 max pooling layer를 **RoI pooling layer로 대체,** 이 때 RoI pooling을 통해 출력되는 feature map의 크기인 H, W는 후속 fc layer와 호환 가능하도록 크기인 **7x7**로 설정

2. 네트워크의 마지막 fc layer를 2개의 fc layer로 대체,

첫 번째 fc layer는 K개의 class와 배경을 포함한 **(K+1)개의 output unit을 가지는 Classifier**이며

두 번째 fc layer는 각 class별로 bounding box의 좌표를 조정하여 **(K+1) * 4개의 output unit을 가지는 bounding box regressor**



3. conv layer3까지의 가중치값은 **고정(freeze)**시켜주고, 이후 layer(conv layer4~ fc layer3)까지의 가중치값이 학습될 수 있도록 **fine tuning**해줌. 논문의 저자는 fc layer만 fine tuning했을 때보다 conv layer까지 포함시켜 학습시켰을 때 더 좋은 성능을 보였다고 함

4. 네트워크가 원본 이미지와 selective search 알고리즘을 통해 추출된 region proposals 집합을 입력으로 받을 수 있도록 변환시켜 줌

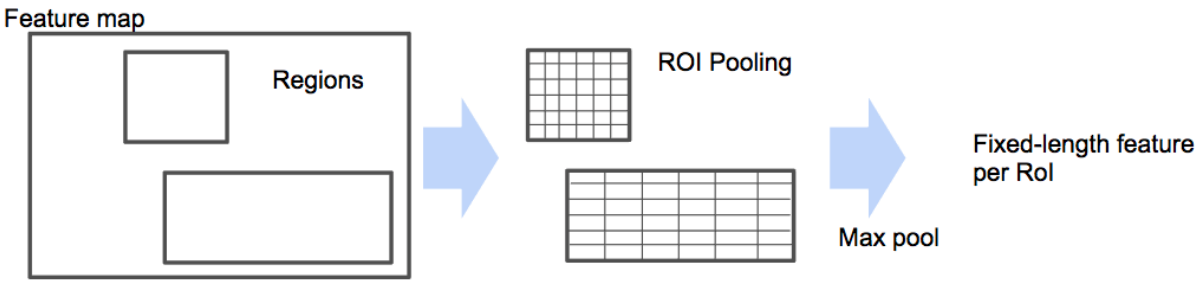
2) region proposal by Selective search

- 먼저 원본 이미지에 대하여 Selective search 알고리즘을 적용하여 미리 region proposals를 추출
  - **Input** : image
  - **Process** : Selective search
  - **Output** : 2000 region proposals

3) Feature extraction(~layer13 pre-pooling) by VGG16

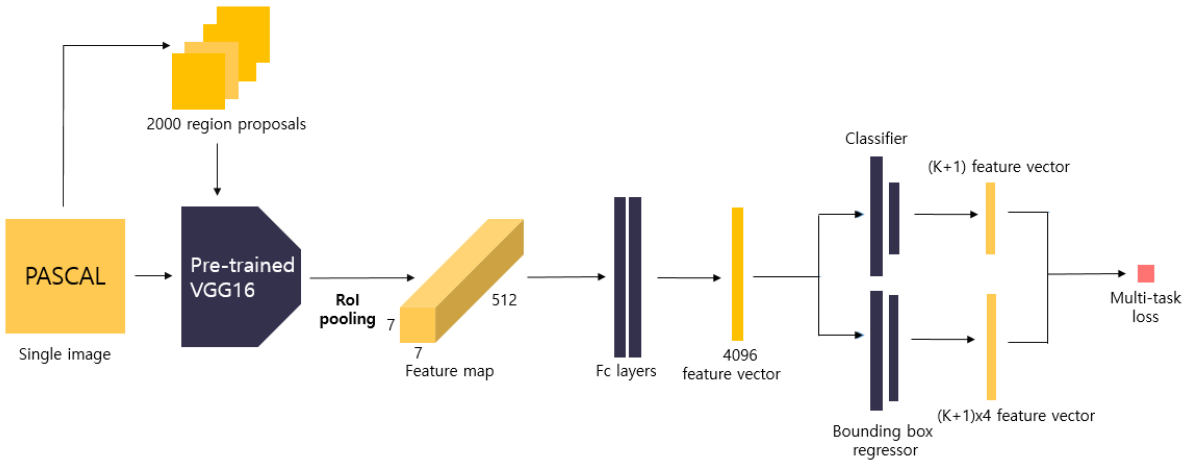
- VGG16 모델에 224x224x3 크기의 원본 이미지를 입력하고, layer13까지의 feature map을 추출
- 마지막 pooling을 수행하기 전에 14x14 크기의 feature map 512개가 출력됩니다.
  - **Input** : 224x224x3 sized image
  - **Process** : feature extraction by VGG16
  - **Output** : 14x14x512 feature maps

4) Max pooling by RoI pooling



- region proposals를 layer13을 통해 출력된 feature map에 대하여 RoI projection을 진행한 후, **RoI pooling**을 수행
- 앞서 언급했듯이, RoI pooling layer는 VGG16의 마지막 pooling layer를 대체한 것
- 이 과정을 거쳐 고정된 7x7 크기의 feature map을 추출
  - **Input** : 14x14 sized 512 feature maps, 2000 region proposals
  - **Process** : RoI pooling
  - **Output** : 7x7x512 feature maps

5) Feature vector extraction by Fc layers



- 다음으로 region proposal별로 7x7x512(=25088)의 feature map을 flatten한 후 fc layer에 입력하여 fc layer를 통해 4096 크기의 feature vector를 얻음
  - **Input** : 7x7x512 sized feature map
  - **Process** : feature extraction by fc layers
  - **Output** : 4096 sized feature vector

6) Class prediction by Classifier

- 4096 크기의 feature vector를 K개의 class와 배경을 포함하여 (K+1)개의 output unit을 가진 fc layer에 입력
- 하나의 이미지에서 하나의 region proposal에 대한 class prediction을 출력
  - **Input** : 4096 sized feature vector
  - **Process** : class prediction by Classifier
  - **Output** : (K+1) sized vector(class score)

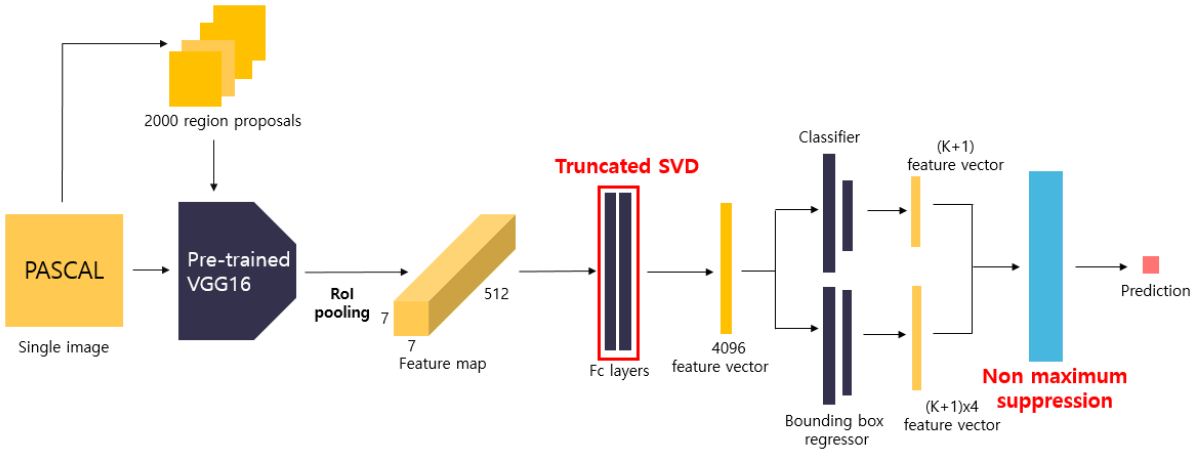
7) Detailed localization by Bounding box regressor

- 4096 크기의 feature vector를 class별로 bounding box의 좌표를 예측하도록 (K+1) x 4개의 output unit을 가진 fc layer에 입력
- 하나의 이미지에서 하나의 region proposal에 대한 class별로 조정된 bounding box 좌표값 출력
  - **Input** : 4096 sized feature vector
  - **Process** : Detailed localization by Bounding box regressor
  - **Output** : (K+1) x 4 sized vector

8) Train Classifier and Bounding box regressor by Multi-task loss

- Multi-task loss를 사용하여 하나의 region proposal에 대한 Classifier와 Bounding box regressor의 loss를 반환
- 이후 Backpropagation을 통해 두 모델(Classifier, Bounding box regressor)을 한 번에 학습
  - **Input** : (K+1) sized vector(class score), (K+1) x 4 sized vector
  - **Process** : calculate loss by Multi-task loss function
  - **Output** : loss(Log loss + Smooth L1 loss)

Detection Fast R-CNN



- Detection 시 동작 순서는 학습 과정과 크게 다르지 않음
- but, 4096 크기의 feature vector를 출력하는 fc layer에 **Truncated SVD**를 적용한다는 점에서 차이가 있음

여러가지 연구

1. Multi Scale Trining and Testing

- 입력 이미지의 크기가 1개로 고정하여 학습했을 때와 5개의 크기를 사용했을 때의 연구 결과

	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**

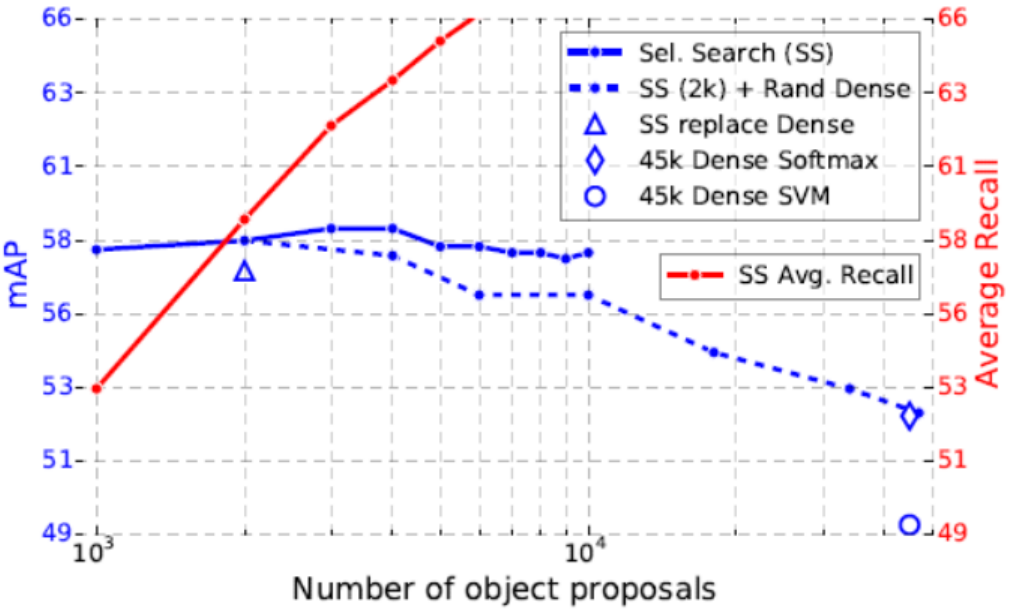
- 5-scale은 더 높은 mAP을 얻었지만, 많은 cost가 요구
- L 모델에서는 GPU 한계 때문에 5-scale 실험을 못했다고 함

2. SVM vs Softmax

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

- Fast R-CNN에서 softmax가 SVM보다 좋은 성능을 나타냄
- SVM은 수백 기가바이트의 특징 벡터가 하드디스크에 저장
- softmax는 특징 벡터를 하드디스크에 저장하지 않고 end-to-end learning이 가능

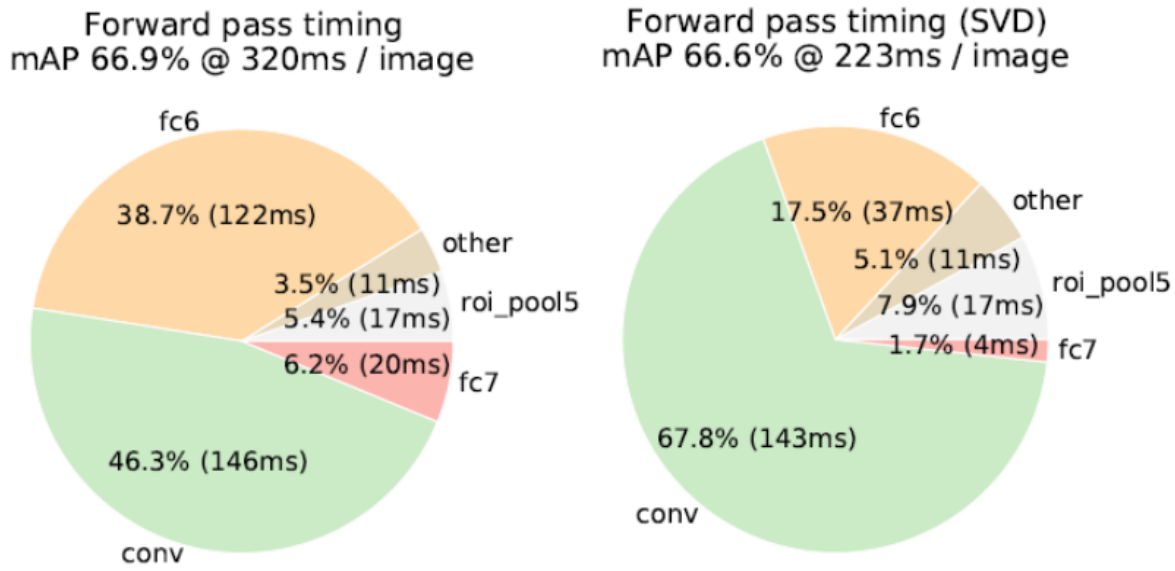
3. Region Proposals



- region proposal 수 증가는 mAP를 꼭 증가시키진 않음
- region proposal 증가에 따라 mAP은 증가하다가 특정 구간부터 감소

4. Truncated SVD for faster detection

- test time때 FC layers에서 많은 시간이 요구
- test time을 감소하기 위해서 FC layer의 weights를 SVD로 감소
- FC6 layer의 25088x4096 행렬로부터 1024개 특이값, FC7 layer의 4096x4096 행렬로부터 256개 특이값을 활용



- SVD로 weights를 압축하여 FC layer의 test time이 감소

Result

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].

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.

- Fast R-CNN을 VOC 2007, 2010, 2012 dataset으로 학습한 성능

	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	-	-	-	-

- Fast R-CNN의 training, test time임

- Fast R-CNN은 R-CNN보대 9배 빠르게 학습되었고 test time에서 213배 빠름
- SPPnet과 비교하여 학습은 3배 빠르고 test는 10배 빠름

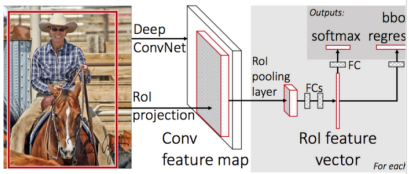
참고 자료

- 블로그

Fast R-CNN 논문 리뷰

이번 포스팅에서는 Fast R-CNN 논문()을 읽고 정리해봤습니다. 기존 R-CNN 모델은 학습 시간이 매우 오래 걸리며, detection 속도 역시, 이미지 한 장당 47초나 걸려 매우 느린 추론 속도를 보였습니다. 또한 3가지의 모델(AlexNet, linear SVM, Bounding box regressor)을 독립적으로 학습시켜, 연산을

https://herbwood.tistory.com/8



[논문 읽기] Fast R-CNN(2014) 리뷰

Fast R-CNN은 R-CNN과 SPPnet의 단점을 개선한 모델입니다. end-to-end learning이 가능하며, 2000개의 proposals이 모두 CNN에 통과하지 않도록 구조를 개선하여 detecting 속도를 높였습니다. R-CNN 단점 1. 학습이 여러 단계로 나뉘어져 있습니다. R-CNN은 3가지 단계의 학습 과정을 거쳐야

https://deep-learning-study.tistory.com/456?category=968059

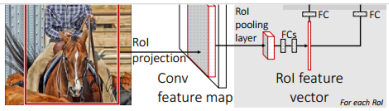
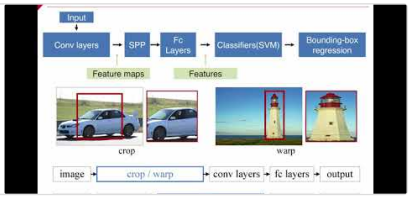


Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs).

- 유튜브

양우식 - Fast R-CNN & Faster R-CNN

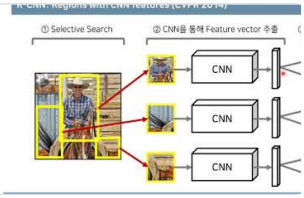
https://www.youtube.com/watch?v=Jo32zrxr6l8



객체 검출(Object Detection) 딥러닝 기술: R-CNN, Fast R-CNN, Faster R-CNN 발전 과정 핵심 요약

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&list=PLRx0vPvlEmdADpce8aoBhNnDaaHQN1Typ&index=25



[Paper Review] Introduction to Object Detection Task : Overfeat, RCNN, SPPNet, FastRCNN

1] 발표자: DSBA 연구실 석사과정 정의석[2] 발표 논문: 본 발표는 Computer Vision Task 중 Object Detection Task의 기반이 되는 논문 4개를 소개합니다.- OverFeat: Integrated Recognition, Localization...

https://www.youtube.com/watch?v=SMEtbrqJ2YI