

# SegNet : A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation

- paper : <https://arxiv.org/pdf/1511.00561.pdf>

## 0. Abstract

1. VGG16에서 FC-Layer 3개를 제외한 13개의 Layer으로 구성된 Encoder를 사용
2. Encoder와 정반대의 Decoder를 이용해서 resolution을 증가시킴
  - 이때, UnMaxpool을 이용해서 non-linear한 Upsampling을 진행함
  - Unpooling의 경우 학습할 필요가 없어서 파라미터 및 속도의 장점이 있음
  - UnMaxpool만 사용할 경우에 Upsampled maps이 Sparse한 단점이 있어서 Conv를 같이 사용해서 Dense하게 만듬
3. FCN / DeepLab-LargeFOV, DeconvNet과 비교
  - 메모리 및 정확도, 속도의 관점
4. 특히, scene understanding의 분야에서 동기를 얻고 이에 대한 테스크를 수행하려고 함
  - Scene understanding application 분야는 속도와 메모리가 중요
5. 다른 모델에 비해 파라미터가 적어서 속도, 메모리 측면에서 효율적임

## 1. Introduction

두 가지의 동기를 통해서 SegNet이라는 모델의 아키텍처를 고안했음

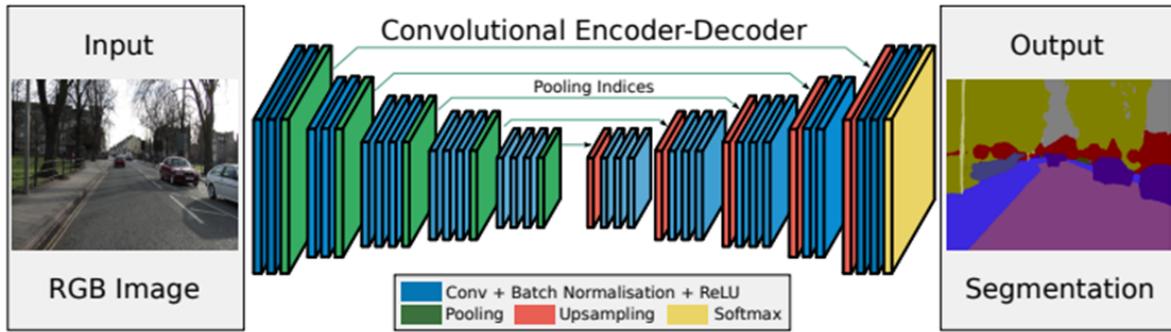
1. Max Pooling 및 Sub Sampling을 통해서 줄어든 특징 맵의 해상도를 증가시킬지에 대한 고민
2. Road Scene Understanding applications라는 분야에서 Semantic Segmentation을 수행하기 위해서 모델이 필요한 능력에 대한 고민

특히, Road Scene Understanding 분야의 경우 몇 가지 특징이 존재함

1. Scene understanding에서 objects간의 관계를 이해하는데에 Semantic segmentation 사용
2. Scene understanding의 경우 appearance(road, building)과 shape(cars, pedestrians)를 이해하고 그들간의 관계를 이해할 필요가 있음 (road - cars, pedestrians - side-walk)



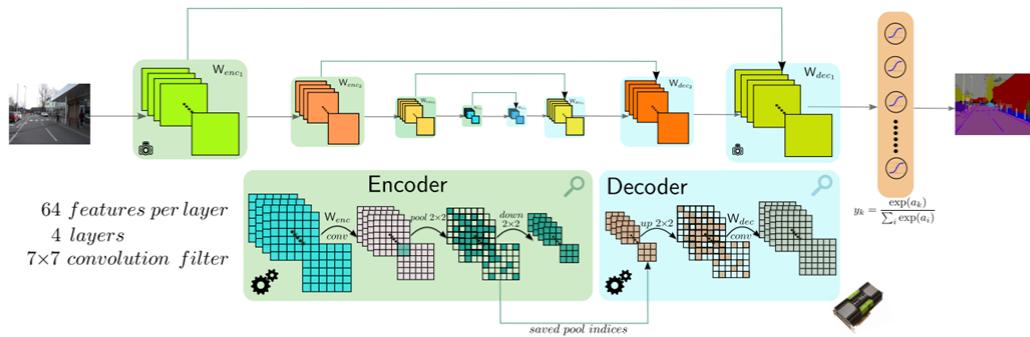
## 3. Architecture



1. VGG16의 13개 층을 Encoder로 사용하고 뒤집은 부분을 Decoder로 사용
  - 중간의 Fully Connected Layer 부분을 제거해서 학습 파라미터를 줄임 ( $134M \rightarrow 14.7M$ )
2. Encoder 부분은 pretrained 된 네트워크를 사용
  - Conv + BN + ReLU
3. MaxPooling의 장점과 단점
  - Max Pooling –  $2 \times 2$  window with stride 2 (overlapped 안되도록)
  - Max Pooling을 통해 translation invariance + large context를 each pixel에 담음
  - spatial resolution이 사라지는 문제가 발생 (디테일한 부분이 사라짐)
  - 위의 문제점을 잡아 주기위한 장치가 필요하고 이게 UnMaxPooling 기법
4. UnMaxPooling의 장점과 단점
  - 메모리의 효율성 + 학습 및 인퍼런스 속도가 빠름
  - 경계에 대한 정보를 효율적으로 저장할 수 있음
5. 다른 Architectures과의 비교 (DeconvNet / UNet)
  - DeconvNet : FC Layer + 2 stage training + larger parameterization
  - UNet : Use Feature map / don't use pretrained weight of VGG net

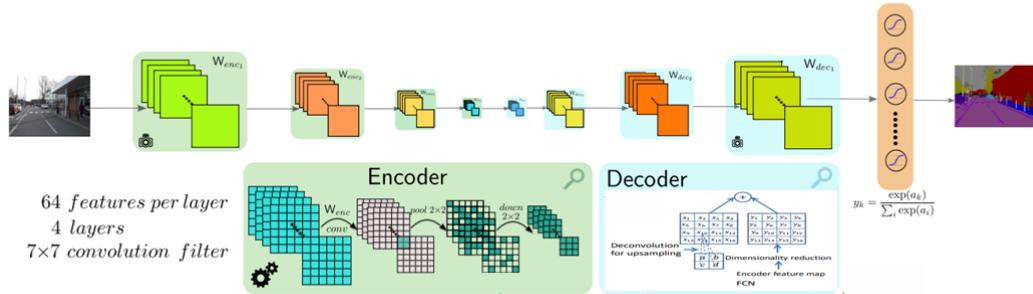
### 3.1 Decoder Variants

#### ✓ SegNet-Basic



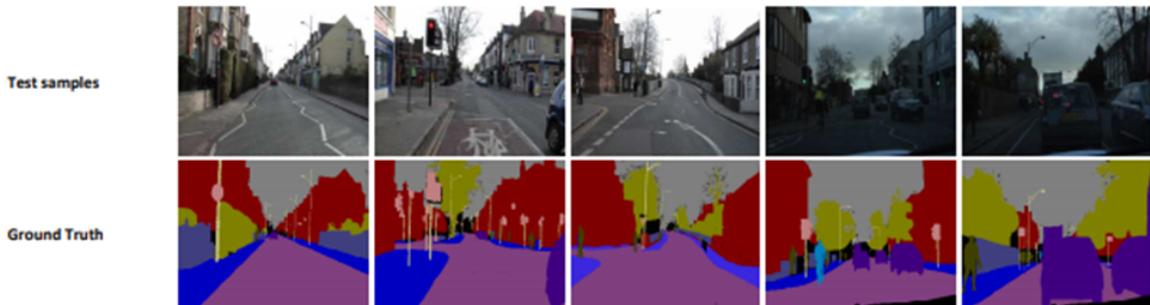
1. SegNet의 small version으로 4 encoders와 4 decoders으로 구성 (SegNet은 5개씩으로 구성)
2. Encoders는 max-pooling 과 sub-sampling을 수행하고 이때의 indice를 받아서 unmaxsampling 을 수행
3. Conv layer 이후에는 BN 이 사용되고 Bias는 없음
4. Decoder Network에는 Relu, Bias 를 사용하지 않음
5. 7x7 의 conv을 이용해서 wide context를 잡으려고 함

## FCN-Basic



1. SegNet-Basic과 Encoder는 동일하게 사용
2. Decoder에는 UnMaxPooling이 아닌 Transposed Convolution으로 전부 대체

## 3.2 Training



1. Camvid Dataset
  - Training Dataset : 367
  - Test Dataset : 233
  - Class : 12 (배경포함)
2. Initialized (He initialized)
3. Optimization : SGD (lr : 0.1, momentum 0.9)
4. validation score가 가장 높은 Epoch 선택
5. Cross Entropy Loss
6. Medium Frequency Balancing
  - larger Class의 weight를 1보다 작게 설정

## 3.3 Analysis

0. 실험을 위한 평가함수와 테스트 데이터셋 세팅
  - Performance Metric
    - Global Accuracy (Pixel Accuracy - G)
    - Class average accuracy (Mean Pixel Accuracy - C)
    - mean Intersection over union (mIoU)
    - Boundary F1-measure (BF)
      - mIoU의 경우 계산방법과 사람이 인식하는 경계부분이 좀 다른데, 이를 보완하기 위해 경계부분에 대해서 정확도를 측정하는 방법
  - Test Set (CAMVID)
    - CamVid validation set을 기준으로 global accuracy가 가장 높은 Epoch로 추론
    - 하지만, 도로와 건물, 하늘, 보도처럼 대부분의 이미지를 차지하는 클래스 때매 클래스의 평균 정확도가 높다고해서 Global Accuracy가 높은게 아님

Variant	Params (M)	Storage multiplier	Infer time (ms)	Median frequency balancing				Natural frequency balancing			
				Test		Train		Test		Train	
G	C	mIoU	BF	G	C	mIoU	BF	G	C	mIoU	
Fixed upsampling											
Bilinear-Interpolation	0.625	0	24.2	77.9	61.1	43.3	20.83	89.1	90.2	82.7	82.7 52.5 43.8 23.08 93.5 74.1 59.9
Upsampling using max-pooling indices											
SegNet-Basic	1.425	1	52.6	82.7	62.0	47.7	35.78	94.7	96.2	92.7	84.0 54.6 46.3 36.67 96.1 83.9 73.3
SegNet-Basic-EncoderAddition	1.425	64	53.0	83.4	<b>63.6</b>	48.5	35.92	94.3	95.8	92.0	<b>84.2</b> 56.5 <b>47.7</b> 36.27 95.3 80.9 68.9
SegNet-Basic-SingleChannelDecoder	0.625	1	33.1	81.2	60.7	46.1	31.62	93.2	94.8	90.3	83.5 53.9 45.2 32.45 92.6 68.4 52.8
Learning to upsample (bilinear initialisation)											
FCN-Basic	0.65	11	24.2	81.7	62.4	47.3	<b>38.11</b>	92.8	93.6	88.1	83.9 55.6 45.0 <b>37.33</b> 92.0 66.8 50.7
FCN-Basic-NoAddition	0.65	n/a	23.8	80.5	58.6	44.1	31.96	92.5	93.0	87.2	82.3 53.9 44.2 29.43 93.1 72.8 57.6
FCN-Basic-NoDimReduction	1.625	64	44.8	<b>84.1</b>	63.4	<b>50.1</b>	37.37	<b>95.1</b>	<b>96.5</b>	<b>93.2</b>	83.5 57.3 47.0 37.13 <b>97.2</b> <b>91.7</b> <b>84.8</b>
FCN-Basic-NoAddition-NoDimReduction	1.625	0	43.9	80.5	61.6	45.9	30.47	92.5	94.6	89.9	83.7 54.8 45.5 33.17 95.0 80.2 67.8

TABLE 1

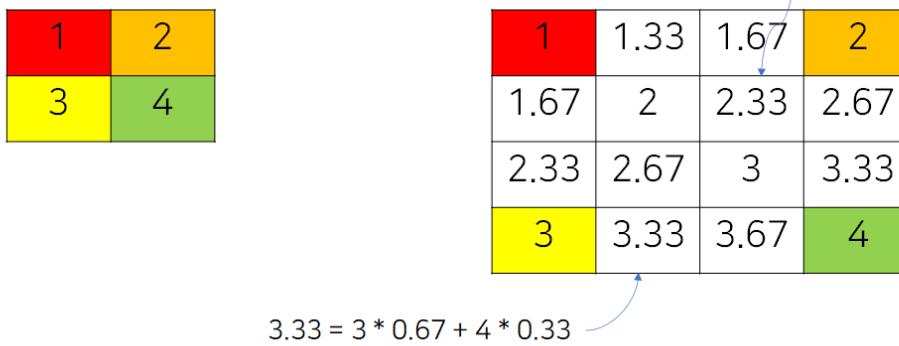
1. 3가지 경우에 대해서 각각 성능을 비교 (파라미터 / 추론속도 / 성능)

- unsampling의 종류 (Bilinear / Upsampling - indices / Learning to upsample)
- Median frequency balancing
- Natural frequency balancing

2. Bilinear-Interpolation without any learning performs의 경우

### ⑤ Bilinear – Interpolation

$$2.33 = 1.67 * 0.33 + 2.67 * 0.66$$



```
nn.Upsample(scale_factor=2, mode='bilinear', align_corners=True)
```

3. SegNet-Basic vs FCN-Basic

Variant	Params (M)	Storage multiplier	Infer time (ms)	Median frequency balancing				Natural frequency balancing			
				Test		Train		Test		Train	
G	C	mIoU	BF	G	C	mIoU	BF	G	C	mIoU	
Fixed upsampling											
Bilinear-Interpolation	0.625	0	24.2	77.9	61.1	43.3	20.83	89.1	90.2	82.7	82.7 52.5 43.8 23.08 93.5 74.1 59.9
Upsampling using max-pooling indices											
SegNet-Basic	1.425	1	52.6	82.7	62.0	47.7	35.78	94.7	96.2	92.7	84.0 54.6 46.3 36.67 96.1 83.9 73.3
SegNet-Basic-EncoderAddition	1.425	64	53.0	83.4	<b>63.6</b>	48.5	35.92	94.3	95.8	92.0	<b>84.2</b> 56.5 <b>47.7</b> 36.27 95.3 80.9 68.9
SegNet-Basic-SingleChannelDecoder	0.625	1	33.1	81.2	60.7	46.1	31.62	93.2	94.8	90.3	83.5 53.9 45.2 32.45 92.6 68.4 52.8
Learning to upsample (bilinear initialisation)											
FCN-Basic	0.65	11	24.2	81.7	62.4	47.3	<b>38.11</b>	92.8	93.6	88.1	83.9 55.6 45.0 <b>37.33</b> 92.0 66.8 50.7
FCN-Basic-NoAddition	0.65	n/a	23.8	80.5	58.6	44.1	31.96	92.5	93.0	87.2	82.3 53.9 44.2 29.43 93.1 72.8 57.6
FCN-Basic-NoDimReduction	1.625	64	44.8	<b>84.1</b>	63.4	<b>50.1</b>	37.37	<b>95.1</b>	<b>96.5</b>	<b>93.2</b>	83.5 57.3 47.0 37.13 <b>97.2</b> <b>91.7</b> <b>84.8</b>
FCN-Basic-NoAddition-NoDimReduction	1.625	0	43.9	80.5	61.6	45.9	30.47	92.5	94.6	89.9	83.7 54.8 45.5 33.17 95.0 80.2 67.8

TABLE 1

- SegNet-Basic : Less memory (Storage multiplier)
- FCN-Basic : encoder feature maps
  - encoder 층마다 feature maps 필요
  - Faster Inference time (Deconvolution layer가 적음)

4. Others

Variant	Params (M)	Storage multiplier	Infer time (ms)	Median frequency balancing				Natural frequency balancing			
				Test		Train		Test		Train	
G	C	mIoU	BF	G	C	mIoU	BF	G	C	mIoU	
Fixed upsampling											
Bilinear-Interpolation	0.625	0	24.2	77.9	61.1	43.3	20.83	89.1	90.2	82.7	82.7 52.5 43.8 23.08 93.5 74.1 59.9
Upsampling using max-pooling indices											
SegNet-Basic	1.425	1	52.6	82.7	62.0	47.7	35.78	94.7	96.2	92.7	84.0 54.6 46.3 36.67 96.1 83.9 73.3
SegNet-Basic-EncoderAddition	1.425	64	53.0	83.4	<b>63.6</b>	48.5	35.92	94.3	95.8	92.0	<b>84.2</b> 56.5 <b>47.7</b> 36.27 95.3 80.9 68.9
SegNet-Basic-SingleChannelDecoder	0.625	1	33.1	81.2	60.7	46.1	31.62	93.2	94.8	90.3	83.5 53.9 45.2 32.45 92.6 68.4 52.8
Learning to upsample (bilinear initialisation)											
FCN-Basic	0.65	11	24.2	81.7	62.4	47.3	<b>38.11</b>	92.8	93.6	88.1	83.9 55.6 45.0 <b>37.33</b> 92.0 66.8 50.7
FCN-Basic-NoAddition	0.65	n/a	23.8	80.5	58.6	44.1	31.96	92.5	93.0	87.2	82.3 53.9 44.2 29.43 93.1 72.8 57.6
FCN-Basic-NoDimReduction	1.625	64	44.8	<b>84.1</b>	63.4	<b>50.1</b>	37.37	<b>95.1</b>	<b>96.5</b>	<b>93.2</b>	83.5 57.3 47.0 37.13 <b>97.2</b> <b>91.7</b> <b>84.8</b>
FCN-Basic-NoAddition-NoDimReduction	1.625	0	43.9	80.5	61.6	45.9	30.47	92.5	94.6	89.9	83.7 54.8 45.5 33.17 95.0 80.2 67.8

TABLE 1

- SegNet-Basic-SingleChannelDecoder : 추론속도와 파라미터의 수가 엄청 작음
- SegNet-Basic-EncoderAddition : UnSampling시에 Max-pooling Indices를 사용
- FCN-Basic-NoDimReduction : 성능이 가장 높고 Infer time도 작음. 파라미터와 메모리의 경우 1.625M과 64로 가장 큼
- FCN-Basic-NoAddition-NoDimReduction : Addition을 제거시에 정확도가 많이 감소 84.8 -> 67.8
- SegNet-Basic-EncoderAddition, FCN-Basic-NoDimReduction과 같이 무거운 모델이 성능이 높음
  - FCN에서는 차원축소를 제거하는게 성능과 BF 측면에서 성능향상이 큼
  - Memory와 정확도 사이에는 Trade-off 관계를 보임

## 5. Summary

- 인코더 기능 맵이 완전히 저장되어 있을 때 최상의 성능을 얻을 수 있음. 특히 의미론적 등고선 설명 메트릭(BF)에 확하게 반영됨
- 추론 중 메모리가 제한될 때, 압축된 형태의 인코더 특징 맵(차원 감소, 최대 풀링 지수)을 적절한 디코더(예: SegNet 유형)와 함께 저장 및 사용하여 성능을 개선할 수 있음
- Decoder의 깊이가 커지면 성능이 향상함

## 4. Benchmarking

### 4.1 Road Scene Segmentation

Method	Building	Tree	Sky	Car	Sign-Symbol	Road	Pedestrian	Fence	Column-Pole	Side-walk	Bicyclist	Class avg.	Global avg.	mIoU	BF
SfM+Appearance [28]	46.2	61.9	89.7	68.6	42.9	89.5	53.6	46.6	0.7	60.5	22.5	53.0	69.1	n/a*	
Boosting [29]	61.9	67.3	91.1	71.1	58.5	92.9	49.5	37.6	25.8	77.8	24.7	59.8	76.4	n/a*	
Dense Depth Maps [32]	85.3	57.3	95.4	69.2	46.5	<b>98.5</b>	23.8	44.3	22.0	38.1	28.7	55.4	82.1	n/a*	
Structured Random Forests [31]						n/a						51.4	72.5	n/a*	
Neural Decision Forests [64]						n/a						56.1	82.1	n/a*	
Local Label Descriptors [65]	80.7	61.5	88.8	16.4	n/a	98.0	1.09	0.05	4.13	12.4	0.07	36.3	73.6	n/a*	
Super Parsing [33]	87.0	67.1	96.9	62.7	30.1	95.9	14.7	17.9	1.7	70.0	19.4	51.2	83.3	n/a*	
SegNet (3.5K dataset training - 140K)	<b>89.6</b>	<b>83.4</b>	96.1	<b>87.7</b>	52.7	96.4	<b>62.2</b>	<b>53.45</b>	<b>32.1</b>	<b>93.3</b>	<b>36.5</b>	<b>71.20</b>	<b>90.40</b>	60.10	46.84
CRF based approaches															
Boosting + pairwise CRF [29]	70.7	70.8	94.7	74.4	55.9	94.1	45.7	37.2	13.0	79.3	23.1	59.9	79.8	n/a*	
Boosting+Higher order [29]	84.5	72.6	<b>97.5</b>	72.7	34.1	95.3	34.2	45.7	8.1	77.6	28.5	59.2	83.8	n/a*	
Boosting+Detectors+CRF [30]	81.5	76.6	96.2	78.7	40.2	93.9	43.0	47.6	14.3	81.5	33.9	62.5	83.8	n/a*	

TABLE 2

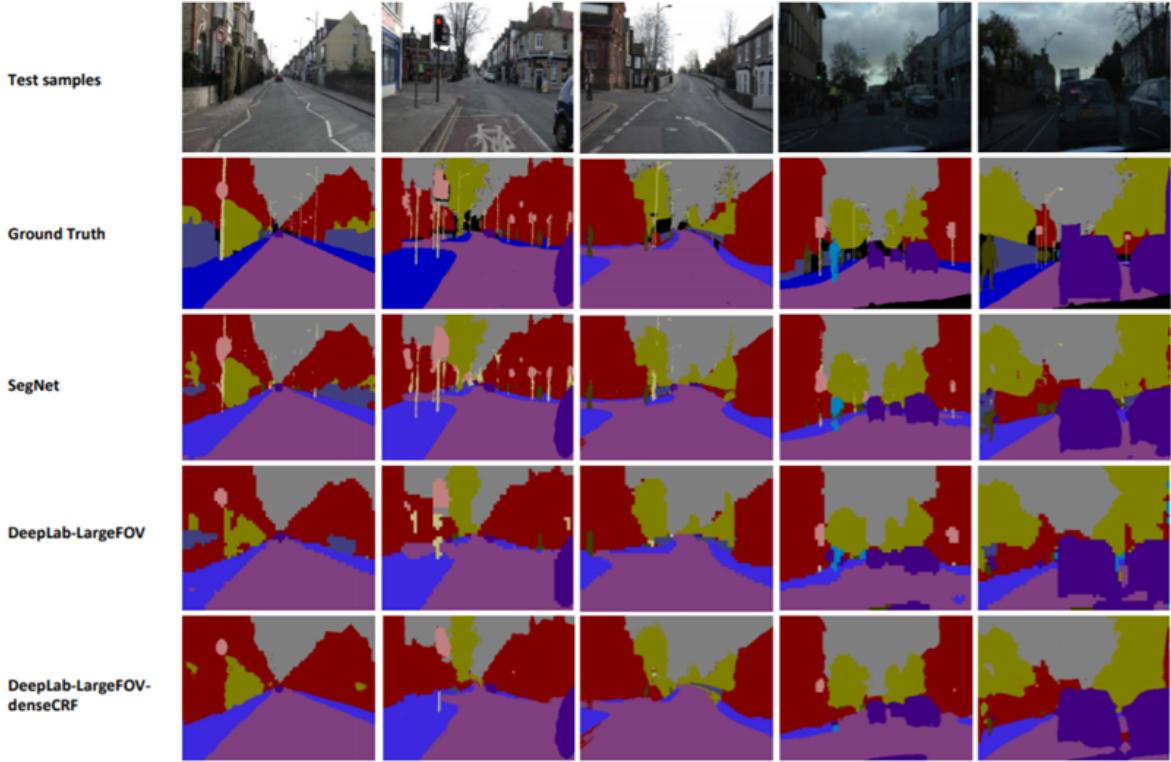
Quantitative comparisons of SegNet with traditional methods on the CamVid 11 road class segmentation problem [22]. SegNet outperforms all the other methods, including those using depth, video and/or CRF's on the majority of classes. In comparison with the CRF based methods SegNet predictions are more accurate in 8 out of the 11 classes. It also shows a good  $\approx 10\%$  improvement in class average accuracy when trained on a large dataset of 3.5K images. Particularly noteworthy are the significant improvements in accuracy for the smaller/thinner classes. \* Note that we could not access predictions for older methods for computing the mIoU, BF metrics.

Network/Iterations	40K				80K				>80K				Max iter
	G	C	mIoU	BF	G	C	mIoU	BF	G	C	mIoU	BF	
SegNet	88.81	59.93	50.02	35.78	89.68	69.82	57.18	42.08	90.40	71.20	60.10	46.84	140K
DeepLab-LargeFOV [3]	85.95	60.41	50.18	26.25	87.76	62.57	53.34	32.04	88.20	62.53	53.88	32.77	140K
DeepLab-LargeFOV-denseCRF [3]				not computed					89.71	60.67	54.74	40.79	140K
FCN	81.97	54.38	46.59	22.86	82.71	56.22	47.95	24.76	83.27	59.56	49.83	27.99	200K
FCN (learnt deconv) [2]	83.21	56.05	48.68	27.40	83.71	59.64	50.80	31.01	83.14	64.21	51.96	33.18	160K
DeconvNet [4]	85.26	46.40	39.69	27.36	85.19	54.08	43.74	29.33	89.58	70.24	59.77	52.23	260K

TABLE 3

Quantitative comparison of deep networks for semantic segmentation on the CamVid test set when trained on a corpus of 3433 road scenes without class balancing. When end-to-end training is performed with the same and fixed learning rate, smaller networks like SegNet learn to perform better in a shorter time. The BF score which measures the accuracy of inter-class boundary delineation is significantly higher for SegNet, DeconvNet as compared to other competing models. DeconvNet matches the metrics for SegNet but at a much larger computational cost. Also see Table 2 for individual class accuracies for SegNet.

- FCN과 DeconvNet에 비해서 SegNet과 DeepLabv1은 적은 iteration에서 높은 성능을 보임
  - 40K, 80K에서 성능의 차이가 특히 발생하고 >80K에서는 DeconvNet하고 성능은 비슷한 모습을 보이고, 오히려 BF 부분은 DeconvNet의 성능이 더 높음
  - SegNet과 DeepLabv1은 G, C, mIoU에 대해서는 초기에는 성능이 거의 비슷하고 Iteration이 늘어날 수록 차이가 발생 (BF의 경우 40K에서도 SegNet이 우수)



## 4.2 SUN RGB-D Indoor Scenes

Network/Iterations	80K				140K				>140K				Max iter
	G	C	mIoU	BF	G	C	mIoU	BF	G	C	mIoU	BF	
SegNet	70.73	30.82	22.52	9.16	71.66	37.60	27.46	11.33	72.63	44.76	31.84	12.66	240K
DeepLab-LargeFOV [3]	70.70	41.75	30.67	7.28	71.16	42.71	31.29	7.57	71.90	42.21	32.08	8.26	240K
DeepLab-LargeFOV-denseCRF [3]	not computed								66.96	33.06	24.13	9.41	240K
FCN (learnt deconv) [2]	67.31	34.32	24.05	7.88	68.04	37.2	26.33	9.0	68.18	38.41	27.39	9.68	200K
DeconvNet [4]	59.62	12.93	8.35	6.50	63.28	22.53	15.14	7.86	66.13	32.28	22.57	10.47	380K

TABLE 4

Quantitative comparison of deep architectures on the SUNRGB-D dataset when trained on a corpus of 5250 indoor scenes. Note that only the RGB modality was used in these experiments. In this complex task with 37 classes all the architectures perform poorly, particularly because of the smaller sized classes and skew in the class distribution. DeepLab-Large FOV, the smallest and most efficient model has a slightly higher mIoU but SegNet has a better G,C,BF score. Also note that when SegNet was trained with *median frequency class balancing* it obtained 71.75, 44.85, 32.08, 14.06 (180K) as the metrics.

- Deep Architectures (SegNet, DeconvNet)가 80K에서 낮은 성능을 보임 (G, C, mIoU)
- SegNet의 경우 G와 BF가 DeepLab-LargeFOV 보다 높은 경향을 보이지만 C와 mIoU는 높은 경향을 보이고 CamVid 데이터셋에 비해 성능이 많이 감소함
  - 첫번째 원인은 Class의 수가 증가했고, small class가 이미지에 많이 등장하는 경향이 있음
  - 두번째 원인은 VGG를 사용하는 Deep한 Architecture때문에 발생 (파라미터가 많아서 수렴 x, 정보손실이 큼)



## 5. Discussion and Future Work

1. 보통의 Deep Learning은 모델이 깊고, 데이터가 많고, 학습이 길어지면 성능이 좋아짐
  - 실험 결과처럼 학습 시간의 상승이 성능의 향상이 크게 관련 없을 경우 학습 시간이 중요함(?)
  - 인퍼런스 시간과 메모리 역시 AR, 자율주행과 같은 특수한 분야에서 매우 중요
  - 전반적인 효율성(성능, 메모리, 학습 및 테스트 시간)에서 SegNet은 효율적임
2. Pascal, MS-COCO 처럼 Class가 적은 경우보다 Scene segmentation은 Class가 훨씬 많고 동시에 등장해서 더 어려움
3. 속도 및 메모리, 성능에 대해서 동일한 비교를 하기 위해서 End-to-End의 학습 및 추론을 사용 (DeconvNet의 경우 Instance-wise Segmentation을 하는데 그렇지 않았다는 의미)

## 6. Conclusion

1. road and indoor scene understanding 분야에서 동기를 얻어서 memory와 computational time을 효율적으로 만드려고 시도
2. SegNet을 다른 논문인 FCN, DeconvNet 등과 비교해서 architectures에 따라서 어떤 식으로 정확도, 학습 및 추론 속도 등이 trade-offs 관계를 가지는지 파악
3. Encoder network feature maps를 저장해서 활용하는게 성능은 가장 좋지만 inference time과 memory 측면에서 좋지 않음
4. 위의 3의 성능을 보완하기 위해서 max-pooling indices를 사용하면 충분히 좋은 성능에 도달
5. 추후, End-to-End의 학습이 잘 되도록 개선할 예정
6. SegNet은 FCN, DeepLab v1 보다는 느리지만 DeconvNet 보다는 빠름

Network	Forward pass(ms)	Backward pass(ms)	GPU training memory (MB)	GPU inference memory (MB)	Model size (MB)
SegNet	422.50	488.71	6803	<b>1052</b>	117
DeepLab-LargeFOV [3]	<b>110.06</b>	<b>160.73</b>	<b>5618</b>	1993	83
FCN (learn deconv) [2]	317.09	484.11	9735	1806	539
DeconvNet [4]	474.65	602.15	9731	1872	877

TABLE 6

A comparison of computational time and hardware resources required for various deep architectures. The caffe time command was used to compute time requirement averaged over 10 iterations with mini batch size 1 and an image of  $360 \times 480$  resolution. We used nvidia-smi unix command to compute memory consumption. For training memory computation we used a mini-batch of size 4 and for inference memory the batch size was 1. Model size was the size of the caffe models on disk. SegNet is most memory efficient during inference model.

- FCN, DeepLabv1에 비해 SegNet은 Decoder Architecture가 있어서 느릴 수밖에 없음
- DeconvNet에서 FC Layer를 제거한 구조여서 DeconvNet 보다는 속도가 빠름

7. SegNet은 Training, Inference memory가 작은 편이고 Model Size 또한 FCN, DeconvNet에 비해 작음

8. Object의 크기가 큰 경우에 대해서는 잘 맞추는 모습을 보이지만, 반대의 경우에 대해서는 성능이 떨어지는 모습을 보임

Wall	Floor	Cabinet	Bed	Chair	Sofa	Table	Door	Window	Bookshelf	Picture	Counter	Blinds
83.42	93.43	63.37	73.18	75.92	59.57	64.18	52.50	57.51	42.05	56.17	37.66	40.29
Desk	Shelves	Curtain	Dresser	Pillow	Mirror	Floor mat	Clothes	Ceiling	Books	Fridge	TV	Paper
11.92	11.45	66.56	52.73	43.80	26.30	0.00	34.31	74.11	53.77	29.85	33.76	22.73
Towel	Shower curtain	Box	Whiteboard	Person	Night stand	Toilet	Sink	Lamp	Bathtub	Bag		
19.83	0.03	23.14	60.25	27.27	29.88	76.00	58.10	35.27	48.86	16.76		

TABLE 5

Class average accuracies of SegNet predictions for the 37 indoor scene classes in the SUN RGB-D benchmark dataset. The performance correlates well with size of the classes in indoor scenes. Note that class average accuracy has a strong correlation with mIoU metric.

## 6.1 Advantages

1. Scene Understanding이라는 특수한 분야의 Task에 초점을 맞춰서 문제를 해결하려고 함

- Object의 크기가 크다는 점
- Object간에 동시에 발생하고 서로 관계가 있다는 점 (보행자-인도, 자동차-도로)
- 자율주행의 경우 Inference 속도가 빨라야 한다는 점

2. 구조 자체가 DeconvNet에서 Fully Connected Layer만 뺀 구조라서 SegNet의 장점을 어필하기 위해 다양한 시도와 초점을 맞춰서 진행

- Scene 분야 / Memory 및 Training, Inference Time / Parameter에 의한 오버피팅
- DeconvNet vs SegNet / FCN vs SegNet / Unet vs Segnet
- Iterations이 적은 경우와 많은 경우에 대해 성능이 어떻게 나오는지 비교
- 클래스의 크기가 작은 경우와 많은 경우에 대해 성능이 어떻게 나오는지 비교

## 6.2 Disadvantages

1. DeconvNet에 Fully Connected Layer만 뺀 논문치고는 굉장히 화려한게 아닌가 생각이 듬 (시점으로 보면 미리 준비하고 있었는데 DeconvNet이 나와서 arxiv에 낸게 아닌가 불쌍하기도 함)

2. CamVid와 SUNRGB-D라는 특수한 데이터셋에 대해서는 성능이 제일 좋았지만 다른 데이터셋에 대해서는 어떻게 나왔을지 비교가 필요(일반화성능)

- 참고로 FCN, DeconvNet, DeepLab 논문의 경우 위의 2개 데이터에 대해서 실험을 한 내용이 없어서 직접 구현해서 돌렸을텐데 튜닝을 어느정도 했을지에 대한 의문도 남음
- PASCAL VOC 2012에 대해서도 비교를 해야 정확한 결과였을 것 같음
- Global Accuracy가 가장 높은거로 선택했다고 했는데 mIoU나 다른 지표로 weight를 선택해서 실험했을때 어떤 결과가 나왔을지에 대해 궁금

3. Large Object를 잡고 Object간의 관계를 파악하는 것을 수행하는게 MaxPooling -> UnMaxPooling인데 과연 이걸로 충분한가? 사실 DeconvNet도 위의 장치가 있는게 결과 차이가 너무 나는게 이상함. Fully Connected Layer 있고 없고에 대해서 어떤 차이가 나오는지 등에 대해 정확한 분석이 들어가야 되고 위의 원인이 맞는지도 의심스러움

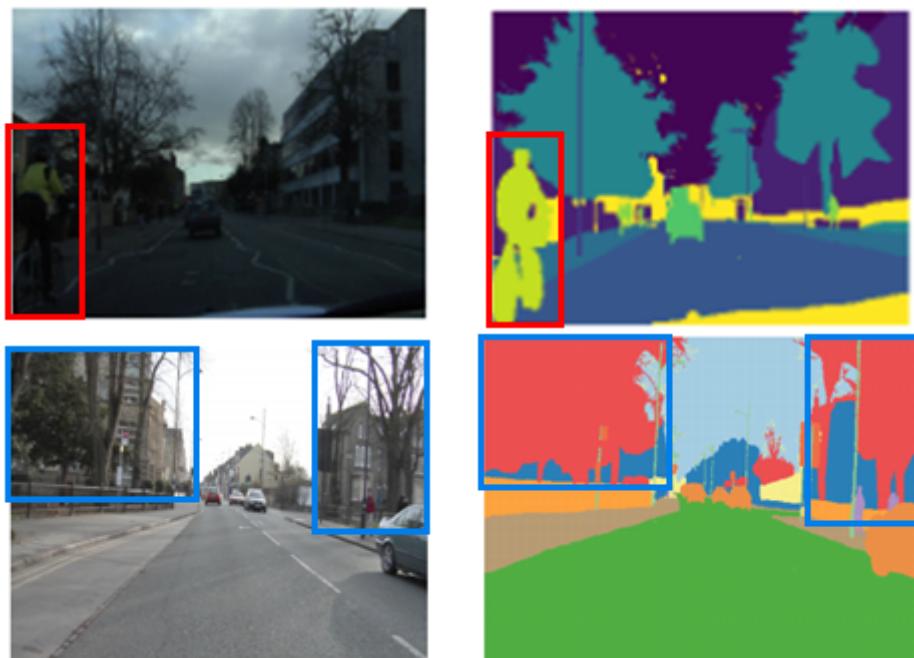
4. 같은 얘기 너무 반복함. 모든 내용에 Training Time / Computation / Memory 중요하다는 얘기가 반복되고 FCN이나 다른 방법에 비해 크게 상승했는지도 의문임

5. CamVid Data가 가지고 있는 문제점

- SegNet의 실험 대부분이 CamVid에서 비교했는데 해당 데이터는 약간의 문제가 있음
  - Input / Output data가 Time Correlated됨



- Dataset의 수가 작고 이상한 라벨들이 존재
  - Train 367 / Test 233
  - Weird Label이 존재 (Bycycle == Person / 라벨이 자세하지 않음)



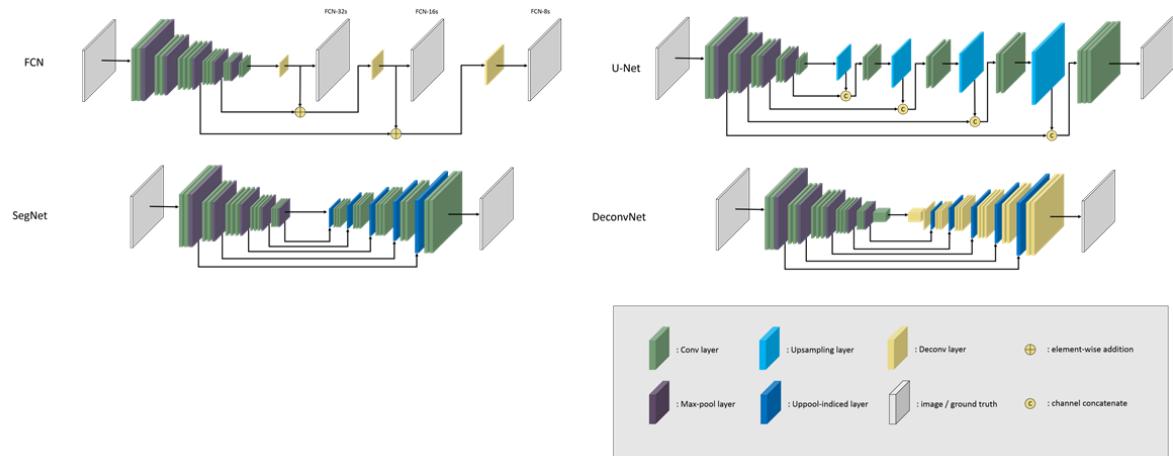
## 7. Appendix

---

### 7.1 참고자료

1. J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015
2. A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012
3. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 2014
4. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. CoRR, abs/1409.4842, 2014
5. B. Hariharan, P. Arbelaez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In ICCV, 2011
6. C. L. Zitnick and P. Dollar. Edge boxes: Locating object proposals from edges. In ECCV, 2014
7. <https://medium.com/@sunnerli/simple-introduction-about-hourglass-like-model-11ee7c30138>

## 7.2 기존의 네트워크와의 비교



이미지 출처 : <https://medium.com/@sunnerli/simple-introduction-about-hourglass-like-model-11ee7c30138>