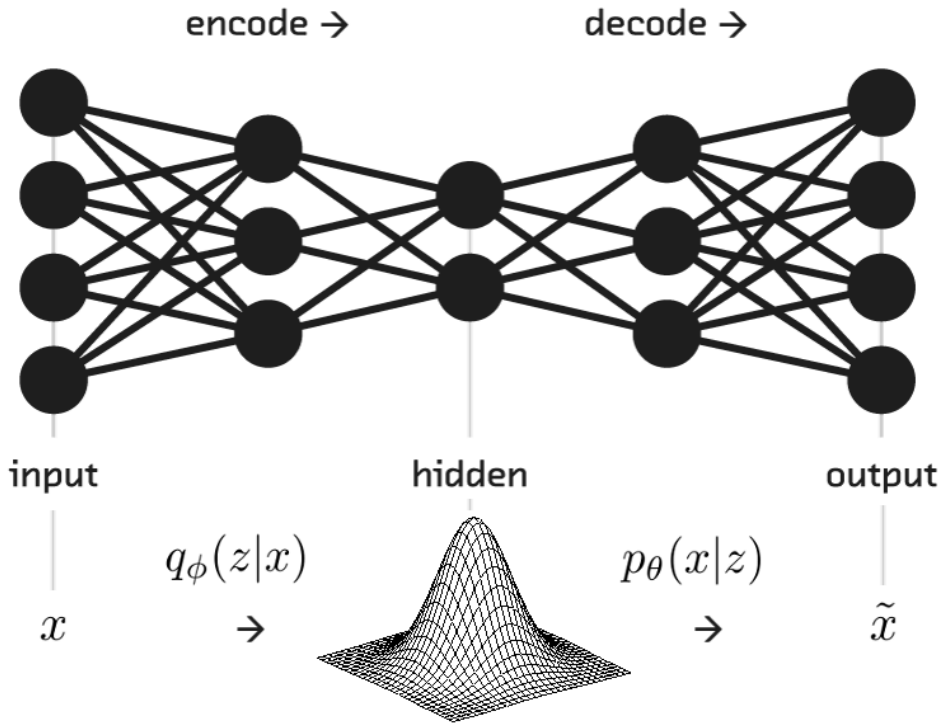
Week 9
- Semantic Image Segmentation & Brain Tumor Segmentation

2019.07.13

Solaris
(http://solarisailab.com)

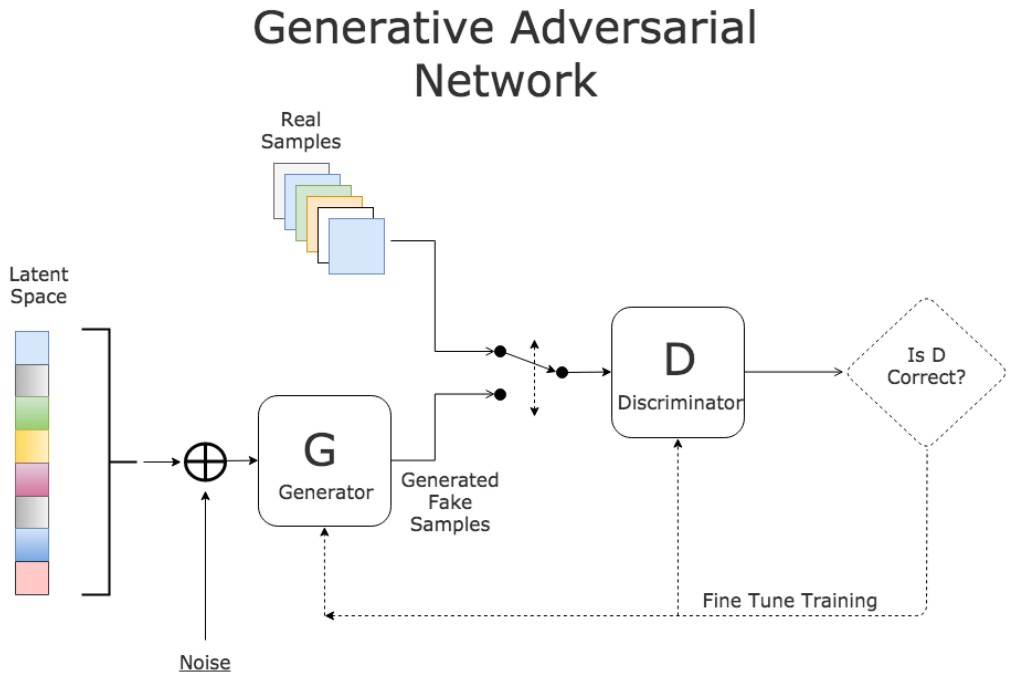
Week 8 복습 – Variational AutoEncoder(VAE)

- 확률 분포를 학습하고, 학습한 확률 분포로부터 새로운 데이터를 생성한다.(Generative Model)



Week 8 복습 – Generative Adversarial Networks(GAN)

- 경찰(Discriminator)과 위조지폐생성도둑(Generator)
- **Generator(생성자)** – Discriminator를 속이기 위한 이미지를 생성하도록 학습 된다.
- **Discriminator(구분자)** – 주어진 이미지가 진짜 이미지 인지 Generator가 생성한 가짜 이미지인지를 구분하도록 학습 된다.



Week 8 복습 – Learning a Driving Simulator

Figure 1: 80×160 samples from the driving dataset.

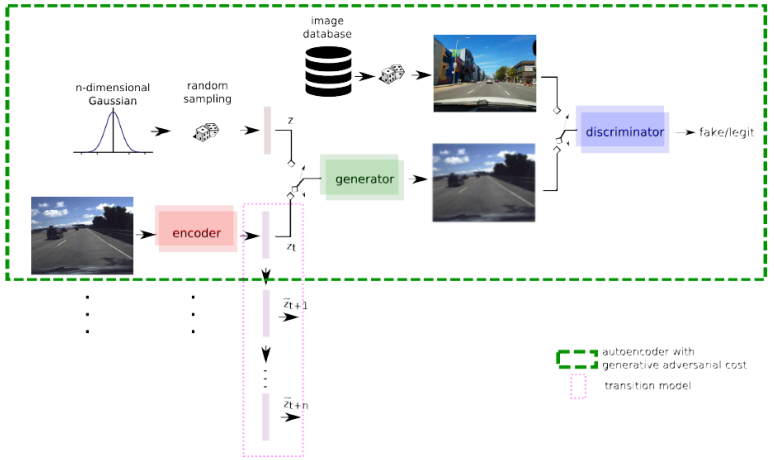


Figure 2: Driving simulator model: an autoencoder trained with generative adversarial costs coupled with a recurrent neural network transition model

Week 8 복습 – Learning a Driving Simulator

$$\mathcal{L} = \mathcal{L}_{prior} + \mathcal{L}_{llike}^{Dis_l} + \mathcal{L}_{GAN}.$$

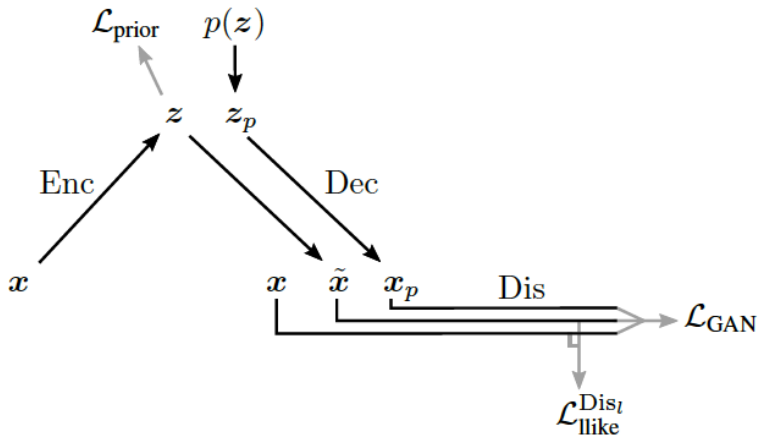


Figure 2. Flow through the combined VAE/GAN model during training. Gray lines represent terms in the training objective.

Algorithm 1 Training the VAE/GAN model

$\boldsymbol{\theta}_{\text{Enc}}, \boldsymbol{\theta}_{\text{Dec}}, \boldsymbol{\theta}_{\text{Dis}} \leftarrow$ initialize network parameters
repeat
  $\boldsymbol{X} \leftarrow$ random mini-batch from dataset
  $\boldsymbol{Z} \leftarrow \text{Enc}(\boldsymbol{X})$
  $\mathcal{L}_{\text{prior}} \leftarrow D_{\text{KL}}(q(\boldsymbol{Z}|\boldsymbol{X})\|p(\boldsymbol{Z}))$
  $\tilde{\boldsymbol{X}} \leftarrow \text{Dec}(\boldsymbol{Z})$
  $\mathcal{L}_{\text{llike}}^{\text{Dis}_l} \leftarrow -\mathbb{E}_{q(\boldsymbol{Z}|\boldsymbol{X})}\left[p(\text{Dis}_l(\boldsymbol{X})|\boldsymbol{Z})\right]$
  $\boldsymbol{Z}_p \leftarrow$ samples from prior $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
  $\boldsymbol{X}_p \leftarrow \text{Dec}(\boldsymbol{Z}_p)$
  $\mathcal{L}_{\text{GAN}} \leftarrow \log(\text{Dis}(\boldsymbol{X})) + \log(1 - \text{Dis}(\tilde{\boldsymbol{X}})) + \log(1 - \text{Dis}(\boldsymbol{X}_p))$
  // Update parameters according to gradients
  $\boldsymbol{\theta}_{\text{Enc}} \overset{+}{\leftarrow} -\nabla_{\boldsymbol{\theta}_{\text{Enc}}}(\mathcal{L}_{\text{prior}} + \mathcal{L}_{\text{llike}}^{\text{Dis}_l})$
  $\boldsymbol{\theta}_{\text{Dec}} \overset{+}{\leftarrow} -\nabla_{\boldsymbol{\theta}_{\text{Dec}}}(\gamma\mathcal{L}_{\text{llike}}^{\text{Dis}_l} - \mathcal{L}_{\text{GAN}})$
  $\boldsymbol{\theta}_{\text{Dis}} \overset{+}{\leftarrow} -\nabla_{\boldsymbol{\theta}_{\text{Dis}}}\mathcal{L}_{\text{GAN}}$
until deadline

Week 8 복습 – Learning a Driving Simulator

$$\mathcal{L} = \mathcal{L}_{prior} + \mathcal{L}_{llike}^{Dis_l} + \mathcal{L}_{GAN}.$$

- **L_prior = D_KL(q(z|x)||p(z))**
- Assuming y_l= **Dis_l(x)** and $\tilde{y}_l$ **= Dis_l(Gen(Enc(x)))**, we have $L_{llikie}^{Dis_l} = \mathbf{E}[(y_l - \tilde{y}_l)^2]$.

Week 8 복습 – Learning a Driving Simulator

$$\mathcal{L} = \mathcal{L}_{prior} + \mathcal{L}_{llike}^{Dis_l} + \mathcal{L}_{GAN}.$$

- Finally, **L_GAN is the generative adversarial network cost** [22]. That cost function represents the game between Gen and Dis.

- **When training Dis, both Enc and Gen are kept fixed and we have**

$$\mathcal{L}_{GAN}^{Dis} = log(Dis(x)) + log(1 - Dis(Gen(u))) + log(1 - Dis(Gen(Enc(x))),$$

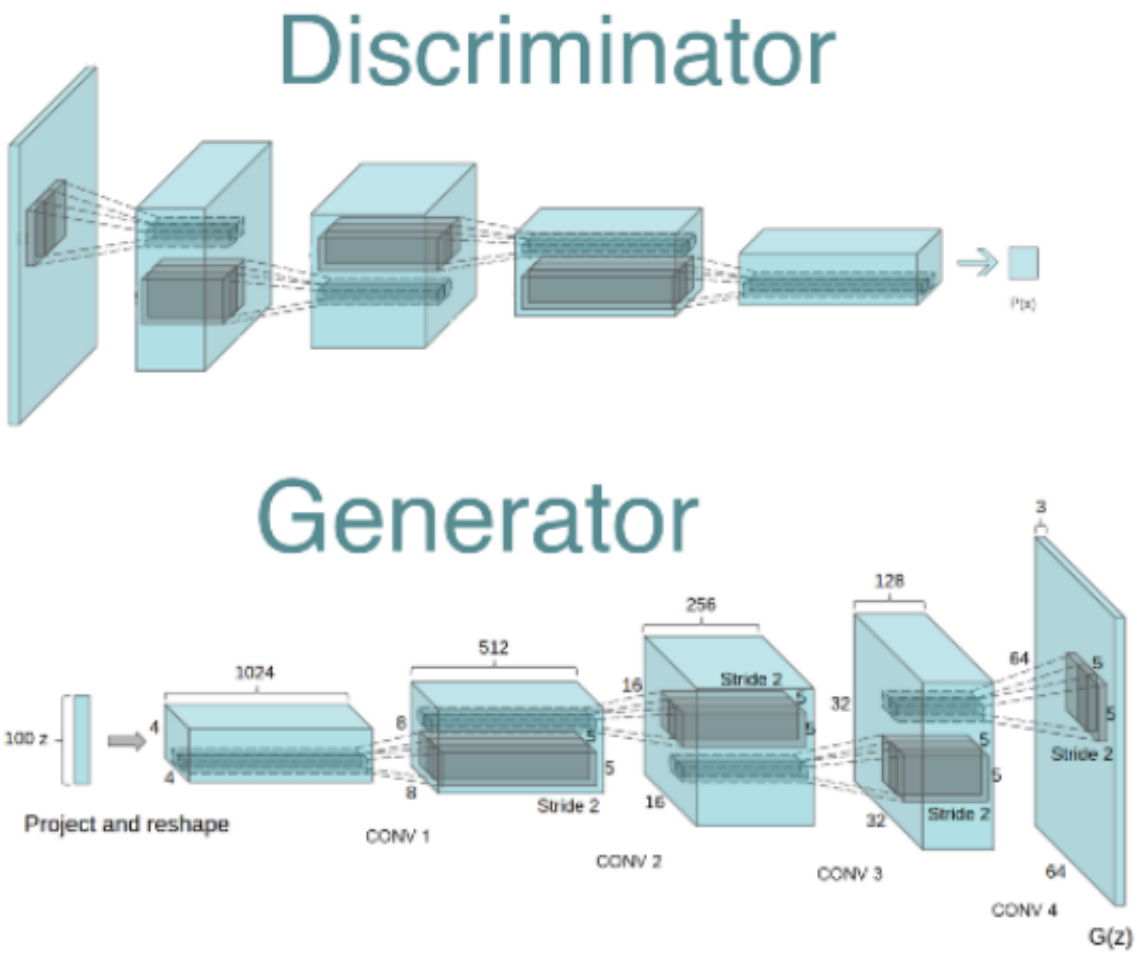
- where **u ~ N(0, 1)** is another random variable. The first r.h.t of (2) represents the log-likelihood of Dis recognizing legit samples and the other two terms represents its log-likelihood to tell fake samples generated from both random vectors u or codes z = Enc(x).

- **When training Gen, we keep both Dis and Enc fixed and we have**

$$\mathcal{L}_{GAN}^{Gen} = log(Dis(Gen(u))) + log(Dis(Gen(Enc(x))),$$

- which accounts for **Gen being able to fool Dis.**

Week 8 복습 – Deep Convolutional Generative Adversarial Networks(DCGAN)



Week 8 복습 - Transition model - RNNs

- After training the autoencoder described above we obtain the transformed dataset applying Enc : $x_t \mapsto z_t$. We train RNN : $x_t, h_t, c_t \mapsto z_{t+1}$ to represent the **transitions in the code space**:

$$h_{t+1} = tanh\left(Wh_t + Vz_t + Uc_t\right),$$
$$\tilde{z}_{t+1} = Ah_{t+1}$$

- Where W, V, U, A are trainable weights, h_t is the **hidden state of the RNN** and c_t are the **concatenated control speed and angle signal**. We leave LSTMs, GRU and multiplicative iterations between c_t and z_t for **future investigations**.

- The cost function for optimizing the trainable weights is simply the mean square error:

$$\mathcal{L}_{RNN} = \mathbb{E}\left[(z_{t+1} - \tilde{z}_{t+1})^2\right].$$

- It is easy to see that (5) is optimal because we impose a Gaussian constraint L_{prior} in the distribution of the codes z when training the autoencoder. In other words, MSE equals up to a constant factor the log-likelihood of a normally distributed random variable.

- **Given a predicted code $\tilde{z}_{t+1}$ we can estimate future frames as $\tilde{x}_{t+1} = Gen(\tilde{z}_{t+1})$.**

Week 8 복습 – Learning a Driving Simulator

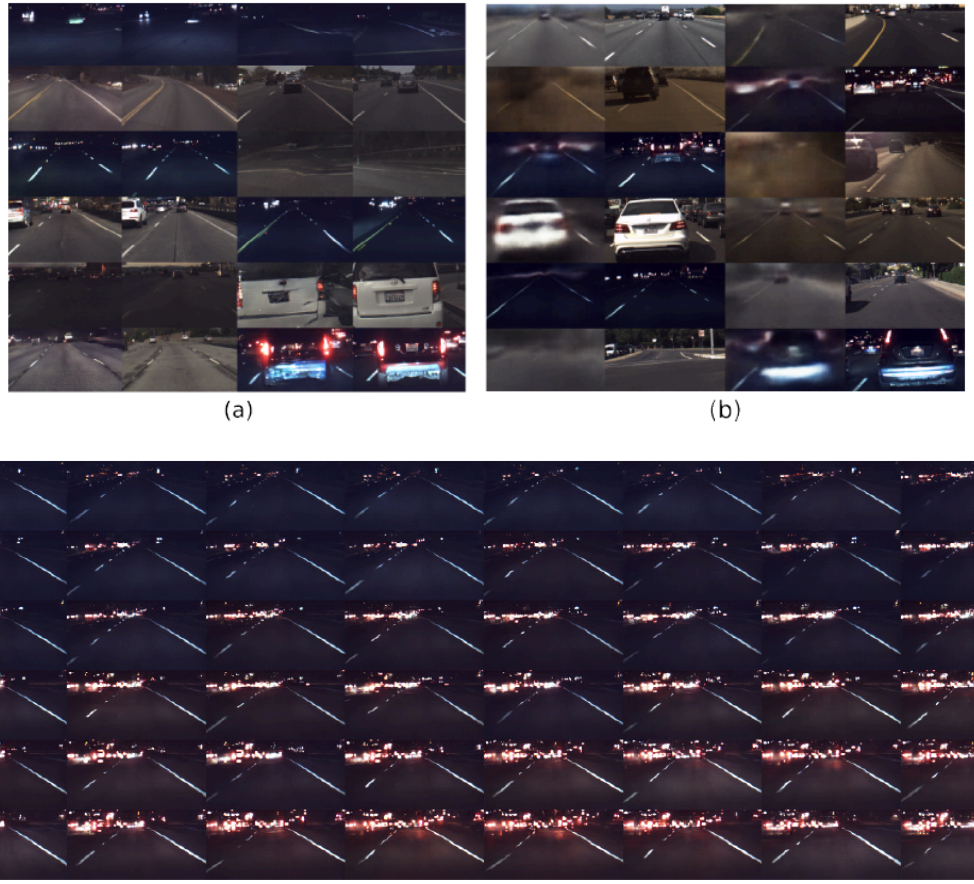
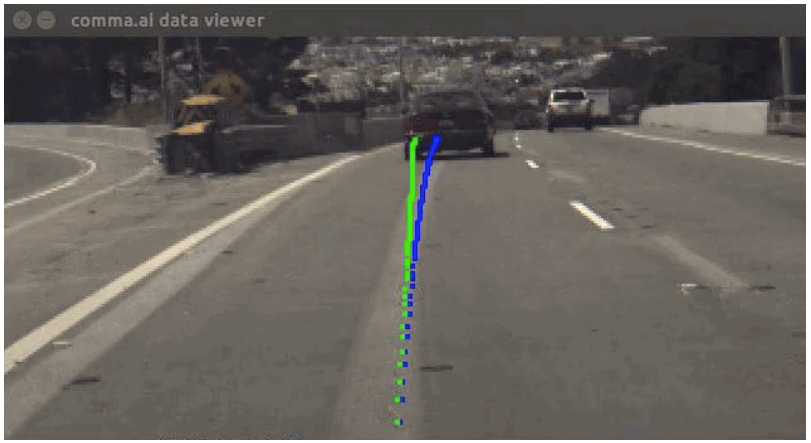


Figure 4: Samples generated by letting the transition model *hallucinate* and decoding $\tilde{z}$ using Gen. Note that Gen was not optimized to make these samples realistic, which support our assumption that the transition model did not leave the code space.

Week 8 복습 – Learning a Driving Simulator



Week 8 복습 – End to End Learning for Self-Driving Cars

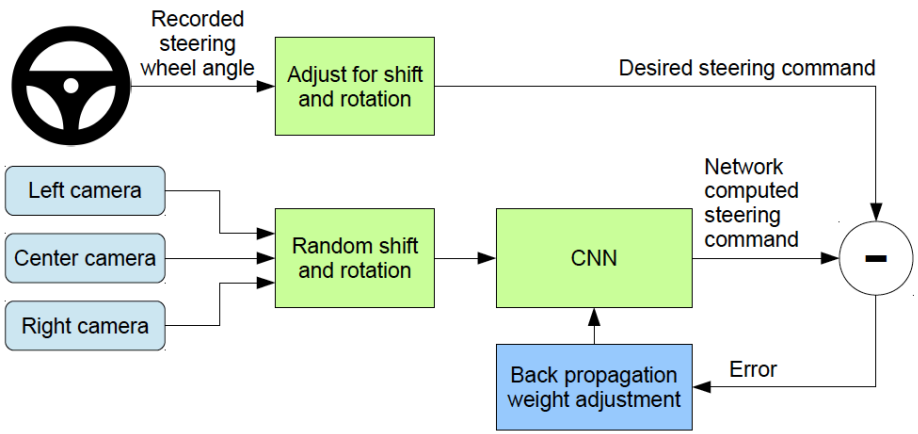
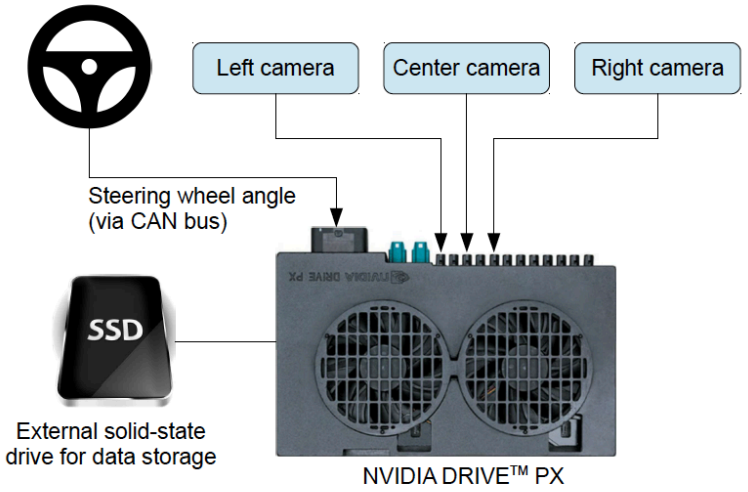


Figure 2: Training the neural network.



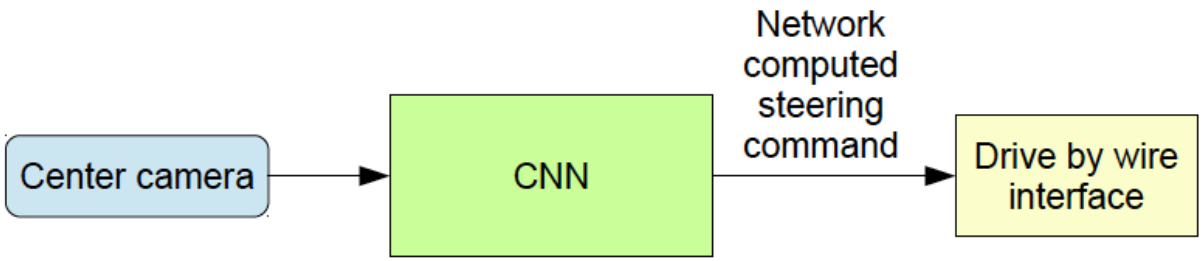


Figure 3: The trained network is used to generate steering commands from a single front-facing center camera.

Week 8 복습 – End to End Learning for Self-Driving Cars

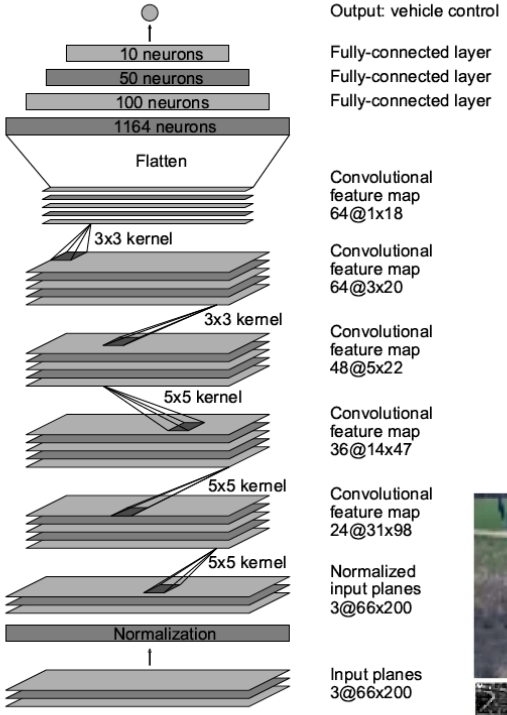


Figure 4: CNN architecture. The network has about 27 million connect parameters.

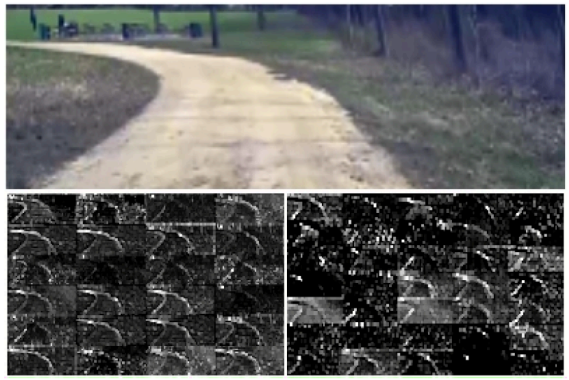


Figure 7: How the CNN "sees" an unpaved road. Top: subset of the camera image sent to the CNN. Bottom left: Activation of the first layer feature maps. Bottom right: Activation of the second layer feature maps. This demonstrates that the CNN learned to detect useful road features on its own, i. e., with only the human steering angle as training signal. We never explicitly trained it to detect the outlines of roads.

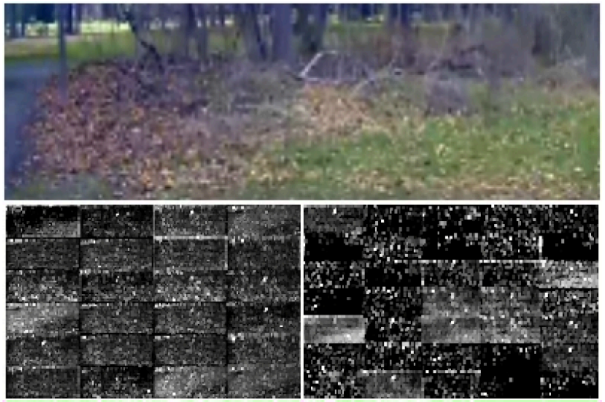


Figure 8: Example image with no road. The activations of the first two feature maps appear to contain mostly noise, i. e., the CNN doesn't recognize any useful features in this image.

Week 8 복습 – End-to-end Learning of Driving Models from Large-scale Video Datasets

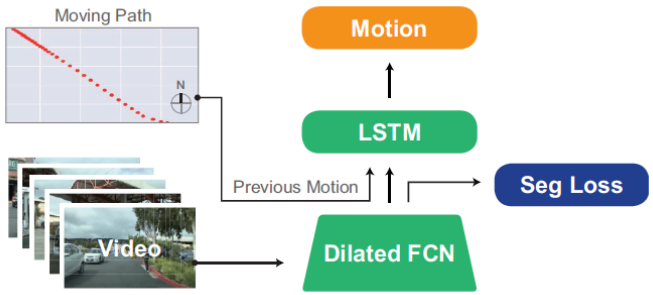


Figure 1: Autonomous driving is formulated as a future egomotion prediction problem. Given a large-scale driving video dataset, an end-to-end FCN-LSTM network is trained to predict multi-modal discrete and continuous driving behaviors. Using semantic segmentation as a side task further improves the model.

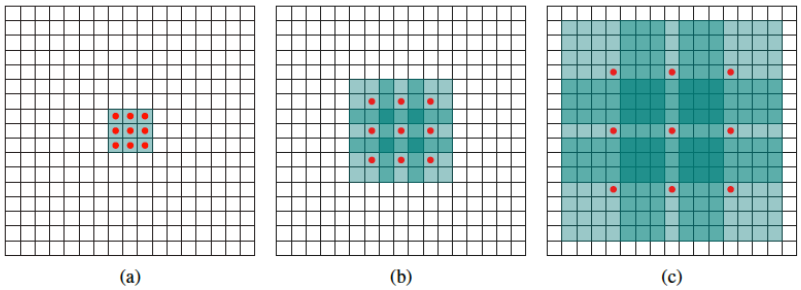
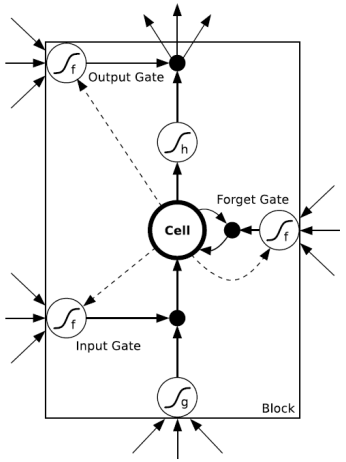


Figure 1: Systematic dilation supports exponential expansion of the receptive field without loss of resolution or coverage. (a) F_1 is produced from F_0 by a 1-dilated convolution; each element in F_1 has a receptive field of 3×3. (b) F_2 is produced from F_1 by a 2-dilated convolution; each element in F_2 has a receptive field of 7×7. (c) F_3 is produced from F_2 by a 4-dilated convolution; each element in F_3 has a receptive field of 15×15. The number of parameters associated with each layer is identical. The receptive field grows exponentially while the number of parameters grows linearly.



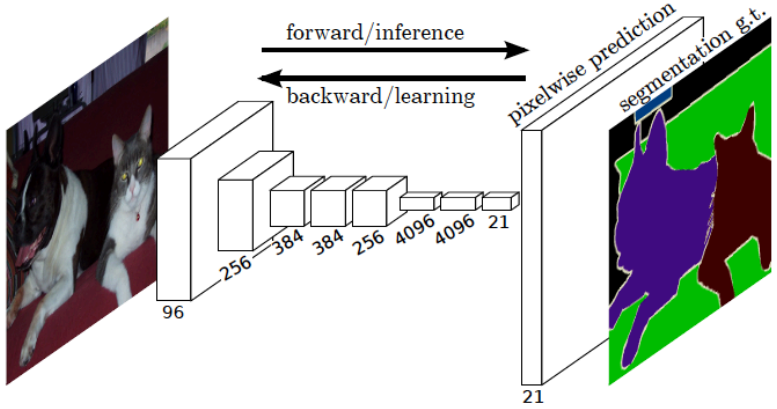


Figure 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

Week 8 복습 – End-to-end Learning of Driving Models from Large-scale Video Datasets

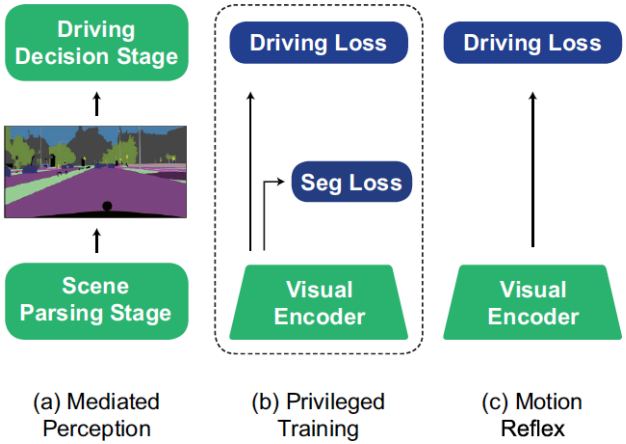


Figure 3: Comparison of learning approaches. Mediated Perception relies on semantic-class labels at the pixel level alone to drive motion prediction. The Motion Reflex method learns a representation based on raw pixels. Privileged Training learns from raw pixels but allows side-training on semantic segmentation tasks.

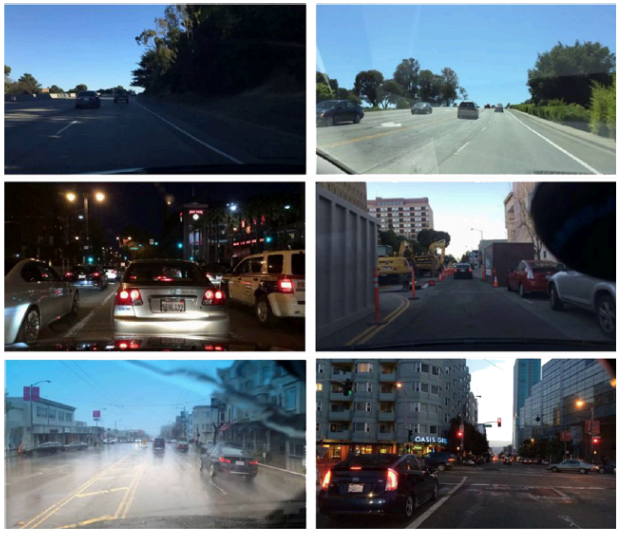
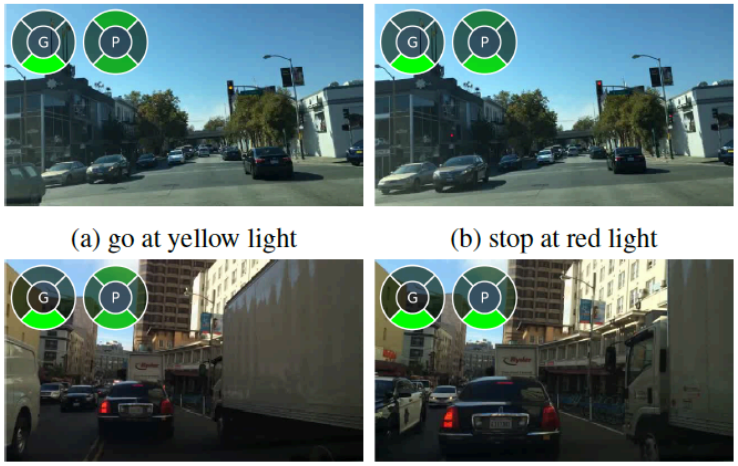


Figure 5: Sample frames from the BDDV dataset.

Datasets	settings	type	Approx scale	Diversity	Specific Car Make
KITTI	**city, rural area, highway**	**real**	less than 1 hour	one city, one weather condition, daytime	Yes
Cityscape	city	**real**	less than 100 hours	German cities, **multiple weather conditions**, daytime	Yes
Comma.ai	mostly highway	**real**	7.3 hours	highway, N.A. , **daytime and night**	Yes
Oxford	city	**real**	214 hours	one city (Oxford), **multiple weather conditions**, daytime	Yes
Princeton Torcs	highway	synthesis	13.5 hours	N.A.	N.A.
GTA	city, highway	synthesis	N.A.	N.A.	N.A.
BDDV(ours)	**city, rural area, highway**	**real**	**10k hours**	**multiple cities, multiple weather conditions,daytime and night**	**No**

Table 1: Comparison of our dataset with other driving datasets.

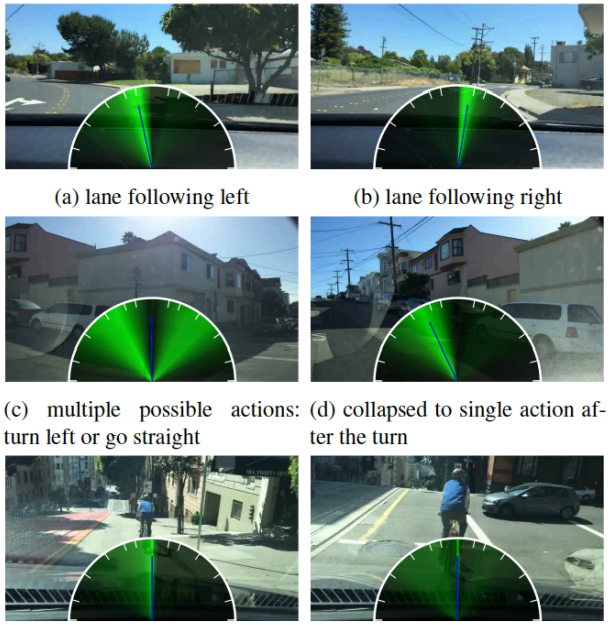
Week 8 복습 – End-to-end Learning of Driving Models from Large-scale Video Datasets



(a) go at yellow light (b) stop at red light

(c) stop & go equal weight at medium distance (d) stop when too close to vehicle ahead

Figure 6: Discrete actions predicted by our FCN-LSTM model. Each row of 2 images show how the prediction changes by time. The green bars shows the probability of doing that action at that time. The red bars are the driver's action. The four actions from top to bottom are going straight, slow or stop, turn left and turn right.



(a) lane following left (b) lane following right

(c) multiple possible actions: turn left or go straight (d) collapsed to single action after the turn

(e) single sided prediction due to side walk (f) right turn becomes available at intersection

Figure 7: Continuous actions predicted by our model. The green sector with different darkness shows the probability map of going to a particular direction. The blue line shows the driver's action.

Week 8 복습 – End-to-end Learning of Driving Models from Large-scale Video Datasets

method	perplexity	accuracy
Motion Reflex Approach	0.718	71.31%
Mediated Perception Approach	0.8887	61.66
Privileged Training Approach	**0.697**	**72.4%**

Table 4: Comparison of the privileged training with other methods.

Figure 8: We show one example result in each column from each of the three models. (a) is the Behavior Reflex Approach. (b) is the Mediated Perception Approach and (c) the Privileged Training Approach.

Week8 복습 - Week8의 학습목표

- **8강의 학습목표 :**

1. Autopilot을 위한 딥러닝 모델들을 살펴본다.
2. TensorFlow + Keras를 이용해서 Driving Scene Simulator & Steering Angle Prediction을 구현해본다.

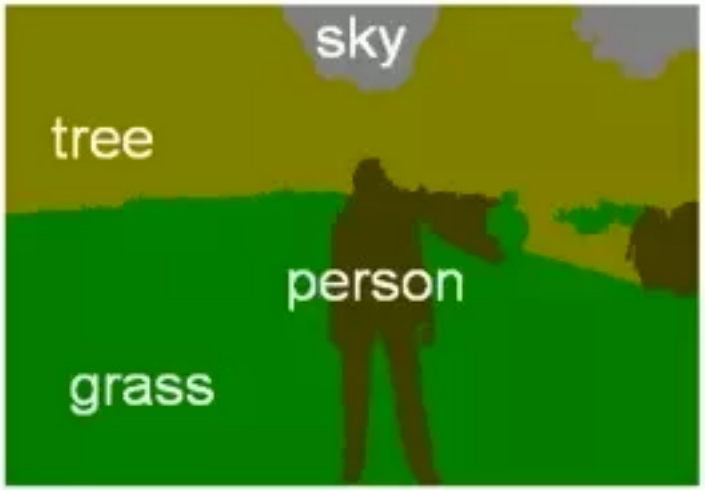
Outline

- Semantic Image Segmentation 문제 소개
- 논문 리뷰 - Fully Convolutional Networks for Semantic Segmentation(FCN)
- TensorFlow를 이용한 Fully Convolutional Networks for Semantic Segmentation(FCN) 구현
- 논문 리뷰 - MULTI-SCALE CONTEXT AGGREGATION BY DILATED CONVOLUTIONS
- 논문 리뷰 - Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation.

Semantic Image Segmentation 문제 소개

- **Semantic Image Segmentation** : 전체 이미지 pixel에서 의미있는 부분끼리 묶어서 prediction하는 기법 (Dense prediction)

Input



Segmentation [9]

Week9의 학습목표

- **9강의 학습목표 :**

1. Semantic Image Segmentation 문제를 이해한다.
2. Semantic Image Segmentation 문제를 위한 딥러닝 모델들을 살펴본다.
3. TensorFlow를 이용해서 Fully Convolutional Networks(FCN) 을 구현해본다.

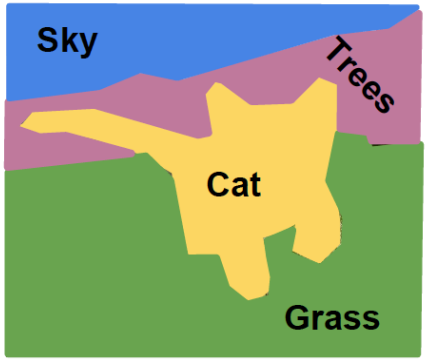
Semantic Image Segmentation 문제 소개

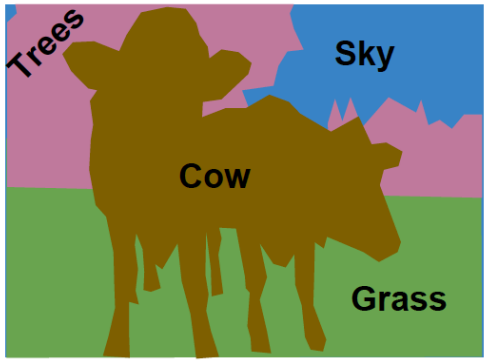
- **Semantic Image Segmentation** : 전체 이미지 pixel에서 의미있는 부분끼리 묶어서 prediction하는 기법 (Dense prediction)

Semantic Segmentation

Label each pixel in the image with a category label

Don't differentiate instances, only care about pixels

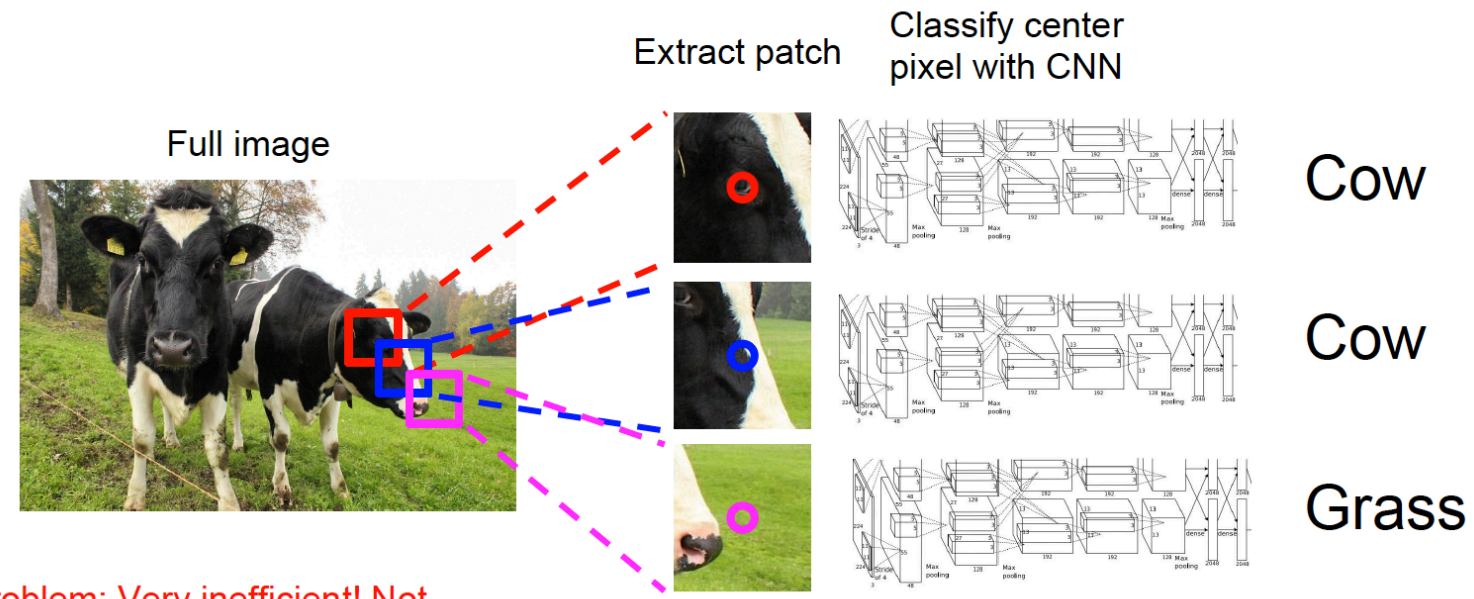




Slide credit: CS231n (https://goo.gl/QSA36t)

Semantic Image Segmetation을 위한 가장 기본적인 아이디어 – Sliding Windows

Semantic Segmentation Idea: Sliding Window



Problem: Very inefficient! Not reusing shared features between overlapping patches

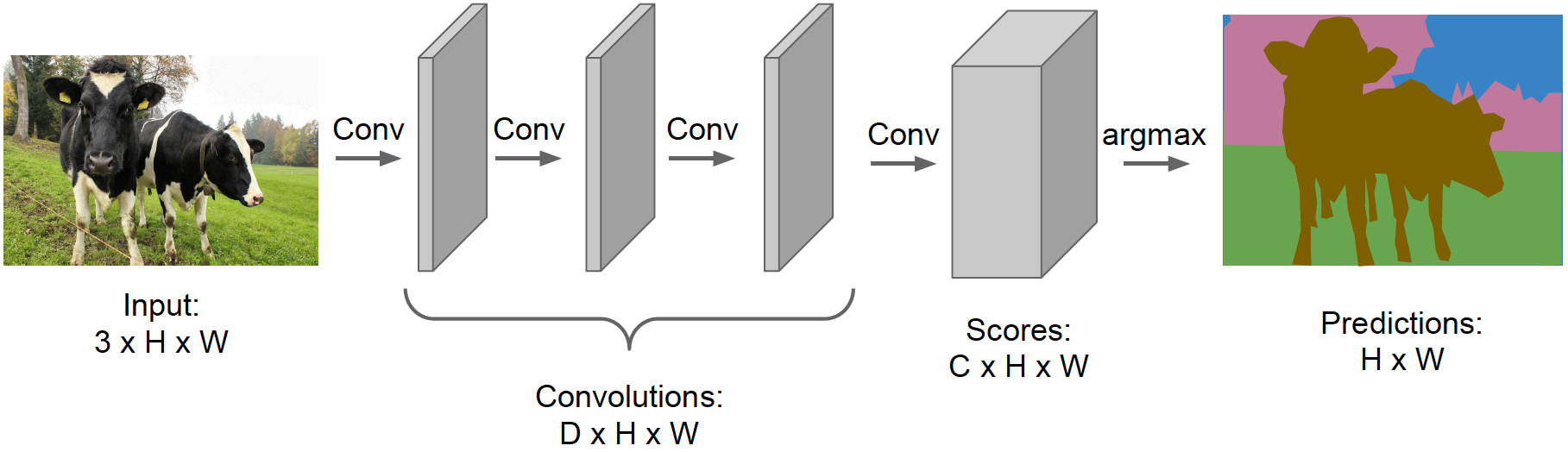
Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

Slide credit: CS231n (https://goo.gl/QSA36t)

최근에 사용되는 방법 – Fully Convolutional Networks(FCN)

Semantic Segmentation Idea: Fully Convolutional

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



Slide credit: CS231n (https://goo.gl/QSA36t)

논문 리뷰 - Fully Convolutional Networks for Semantic Segmentation(FCN)

- "Fully convolutional networks for semantic segmentation.", Long, Jonathan, Evan Shelhamer, and Trevor Darrell, CVPR 2015.

- **핵심 아이디어** : Classification Task에 사용된 CNNs을 변환해서 Segmentation Task로 Transfer Learning해서 사용한다.

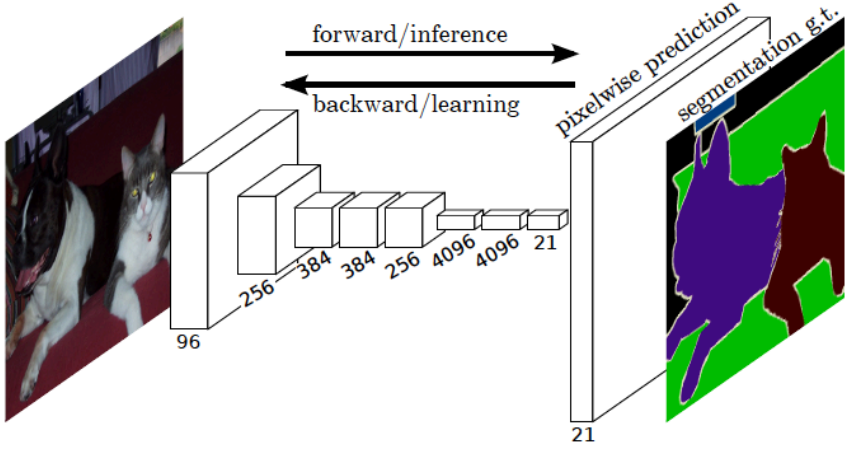
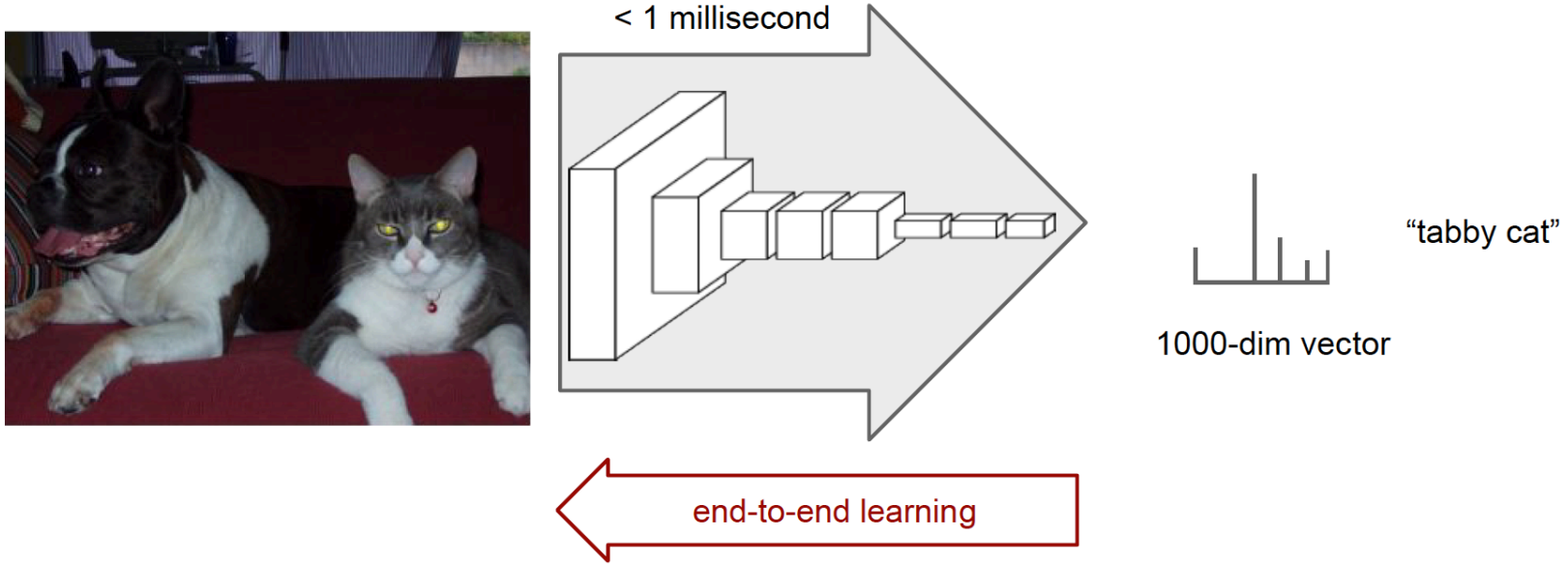


Figure 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

Fully Convolutional Networks (FCN) for Semantic Segmentation

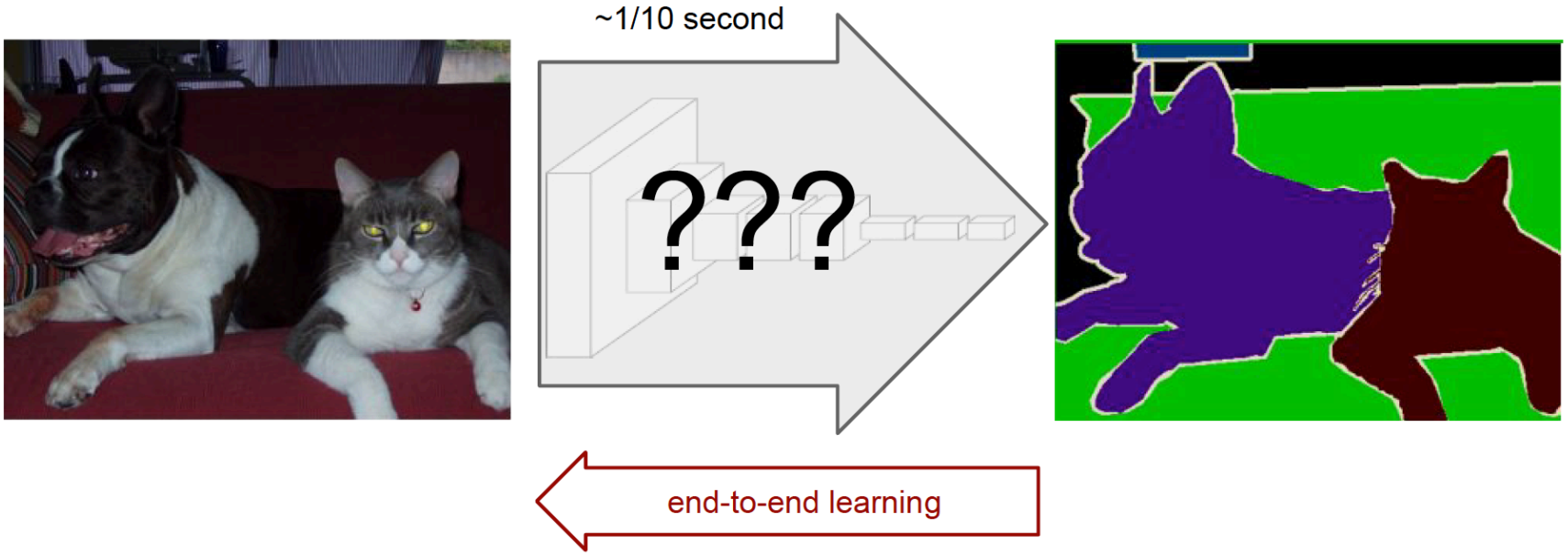
convnets perform classification



Slide credit: Jonathan Long (https://goo.gl/S92NA9)

A classification network

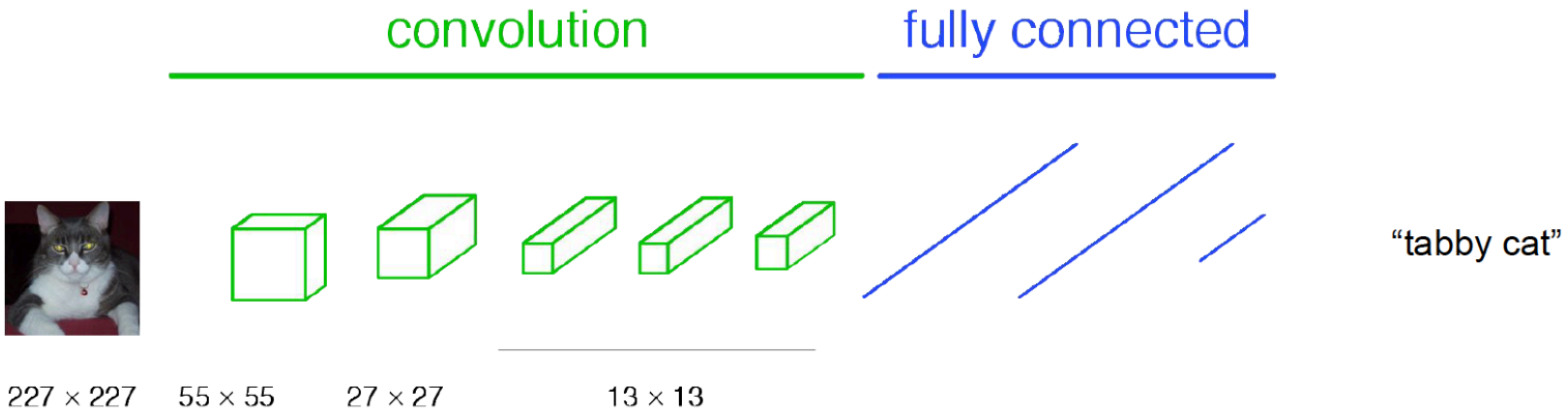
lots of pixels, little time?



Slide credit: Jonathan Long (https://goo.gl/S92NA9)

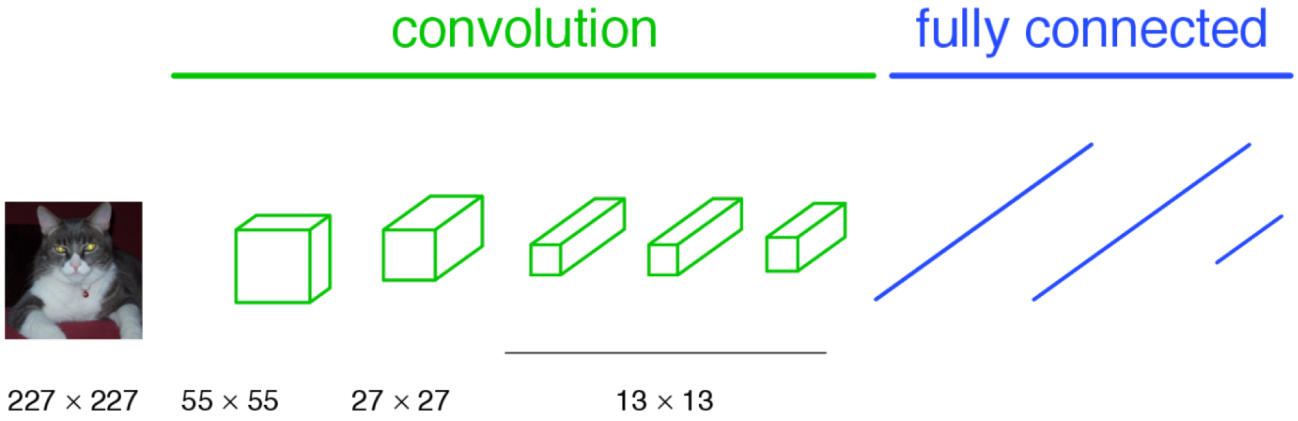
A classification network

a classification network



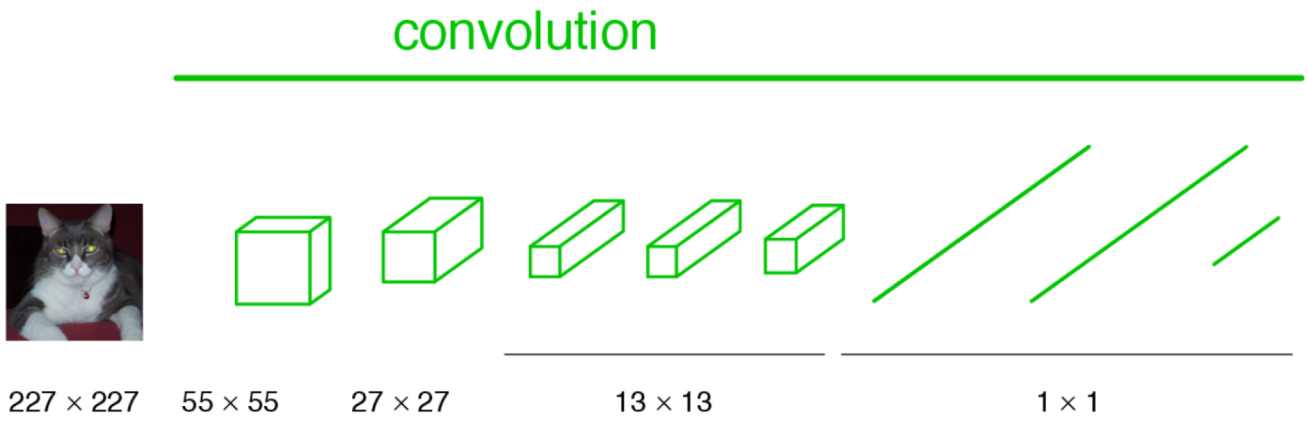
Slide credit: Jonathan Long
(https://goo.gl/S92NA9)

A classification network



Slide credit: Jonathan Long (https://goo.gl/S92NA9)

Fully connected -> 1x1 Convolution

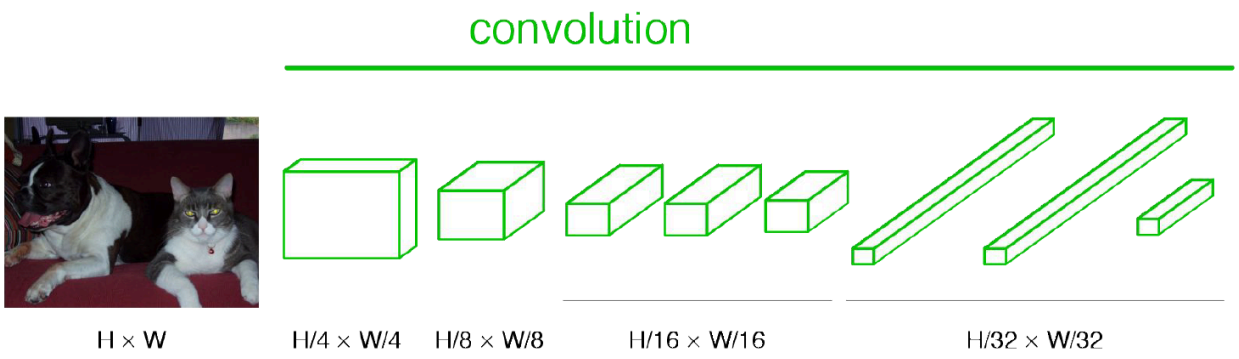


Slide credit: Jonathan Long
(https://goo.gl/S92NA9)

Fully connected -> 1x1 Convolution

- Fully Connected 부분을 1x1 conv로 바꿈으로어떤 어떤 크기 (any size)의 input 이미지도 처리할 수 있다.
- Fully Connected를 이용할 경우 **첫번째 Fully Connected Layer로 들어오는 parameter의 개수가 일정해야 하므로** 고정된 크기(Fixed Size)의 Input Image만을 받을수 있다. 하지만, Convolution Layer일 경우, **이미지 크기만큼 Filter를 Sliding해주면 되므로** 어떤 크기(Any Size)의 Input Image도 받을 수 있다.

becoming fully convolutional



Slide credit: Jonathan Long (https://goo.gl/S92NA9)

Fully connected -> 1x1 Convolution

- Fully Connected -> 1x1 Convolution
- Classification -> output heatmap

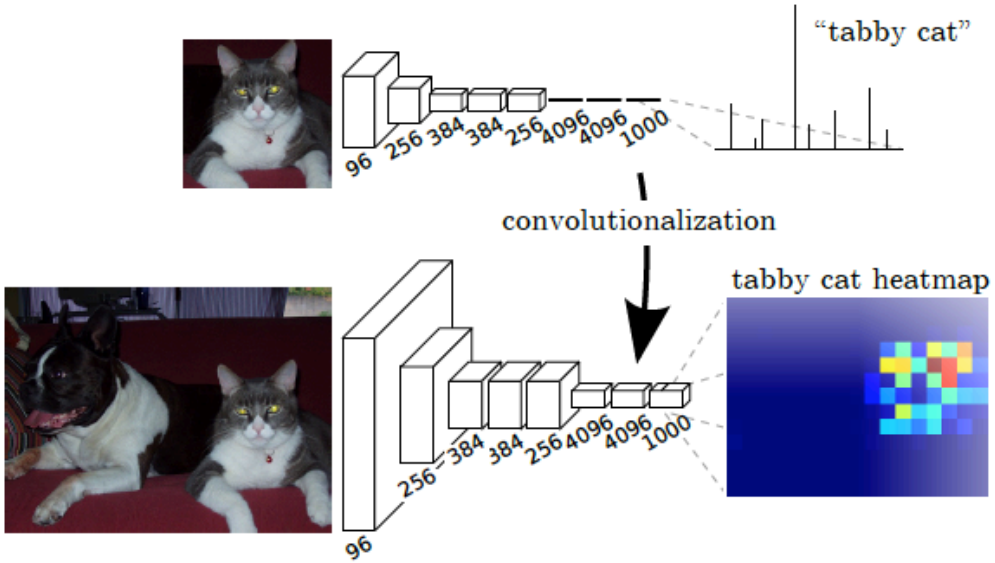
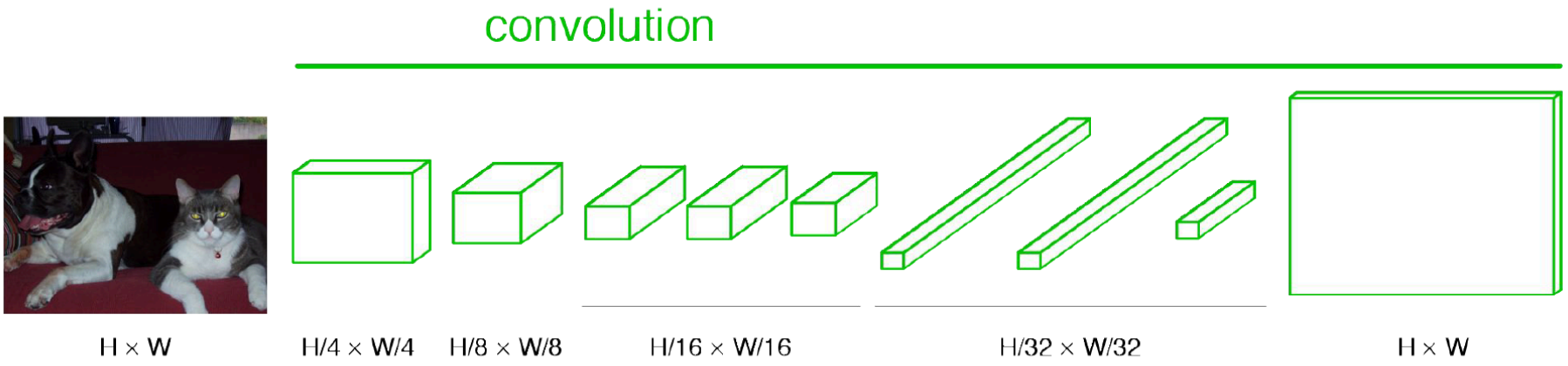


Figure 2. Transforming fully connected layers into convolution layers enables a classification net to output a heatmap. Adding layers and a spatial loss (as in Figure 1) produces an efficient machine for end-to-end dense learning.

Becoming fully convolutional

- 크기가 작아진 Feature Map을 Upsampling을 통해 다시 이미지 사이즈만큼 키운다.

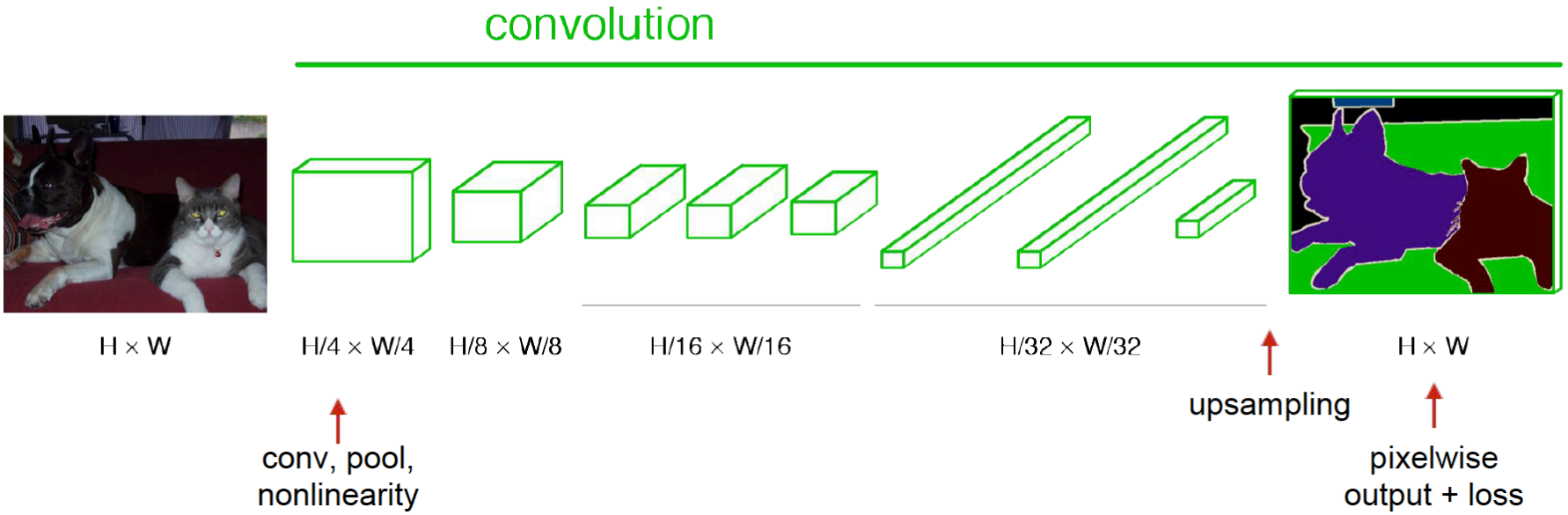
upsampling output



Slide credit: Jonathan Long (https://goo.gl/S92NA9)

Becoming fully convolutional

end-to-end, pixels-to-pixels network



Slide credit: Jonathan Long
(https://goo.gl/S92NA9)

Upsampling is backwards strided convolution

- So long as f is integral, a natural way to upsample is therefore **backwards convolution (sometimes called deconvolution)** with an output stride of f.

- Upsampling 과정을 통합적인 end-to-end backpropagation 과정에 포함시킬 수 있다.(Deconvolution에 사용할 적합한 Filter를 Backpropagation으로 학습)

- 따라서 non-linear upsampling도 가능해지므로 성능이 더욱 향상된다.

Skip Layers

- Upsampling의 정확도를 높이기 위해 마지막 pooling layer인 pool5(가장 coarse한 feature)뿐만 아니라, pool4, pool3(좀더 finer한 feature)도 함께 이용한다.

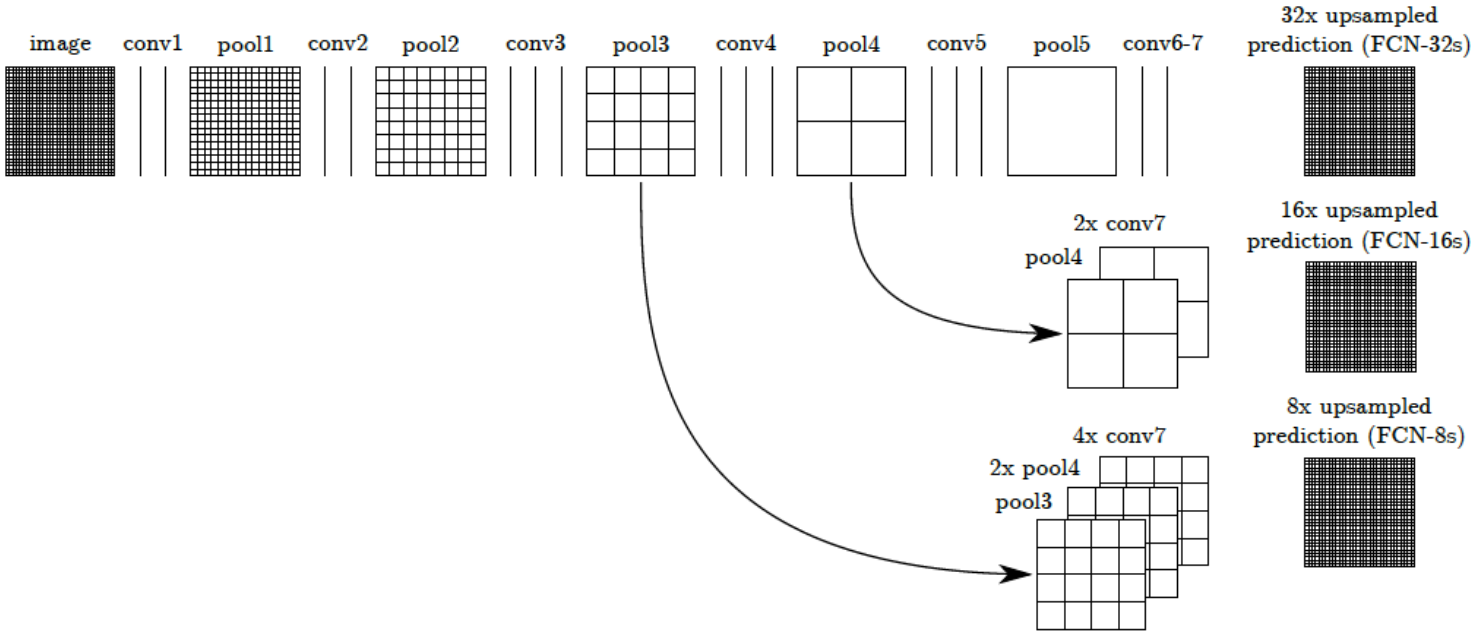
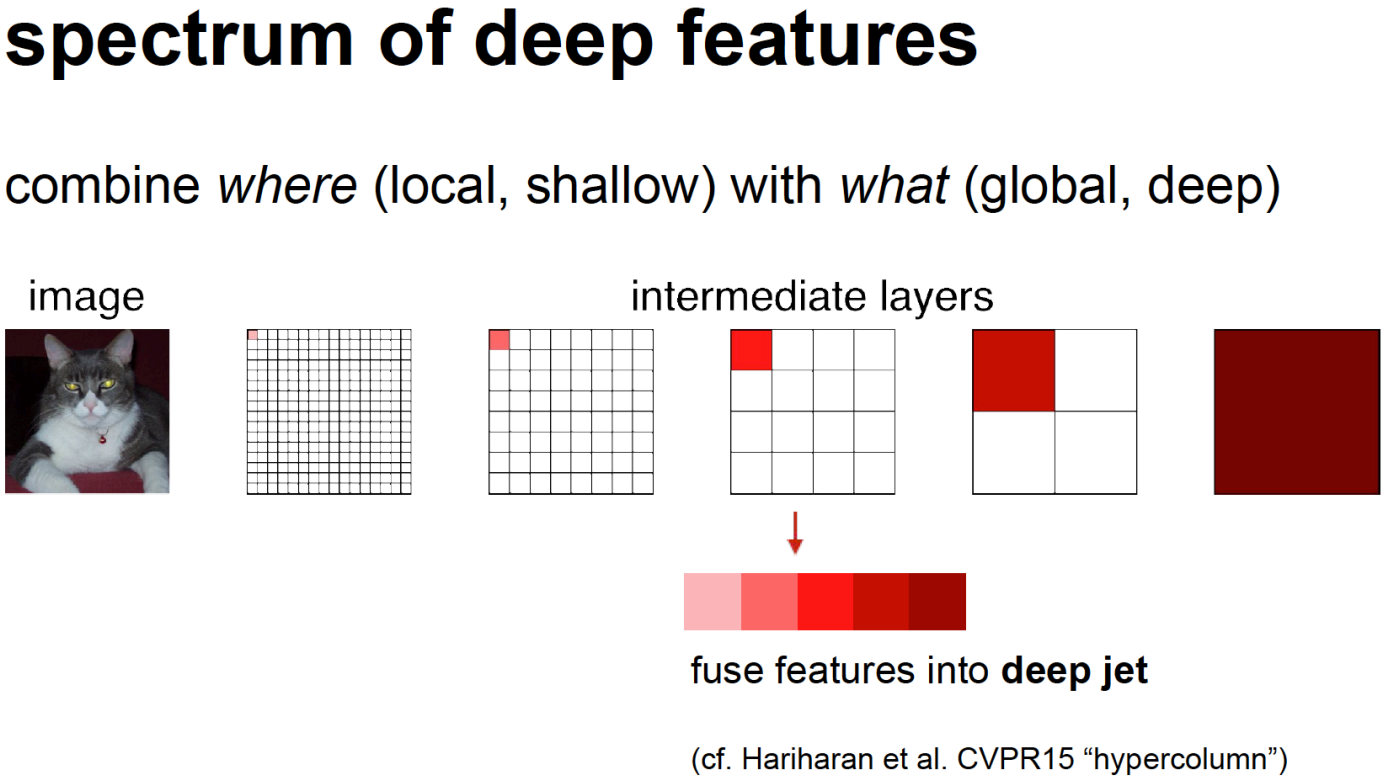


Figure 3. Our DAG nets learn to combine coarse, high layer information with fine, low layer information. Pooling and prediction layers are shown as grids that reveal relative spatial coarseness, while intermediate layers are shown as vertical lines. First row (FCN-32s): Our single-stream net, described in Section 4.1, upsamples stride 32 predictions back to pixels in a single step. Second row (FCN-16s): Combining predictions from both the final layer and the `pool4` layer, at stride 16, lets our net predict finer details, while retaining high-level semantic information. Third row (FCN-8s): Additional predictions from `pool3`, at stride 8, provide further precision.

Fuse features

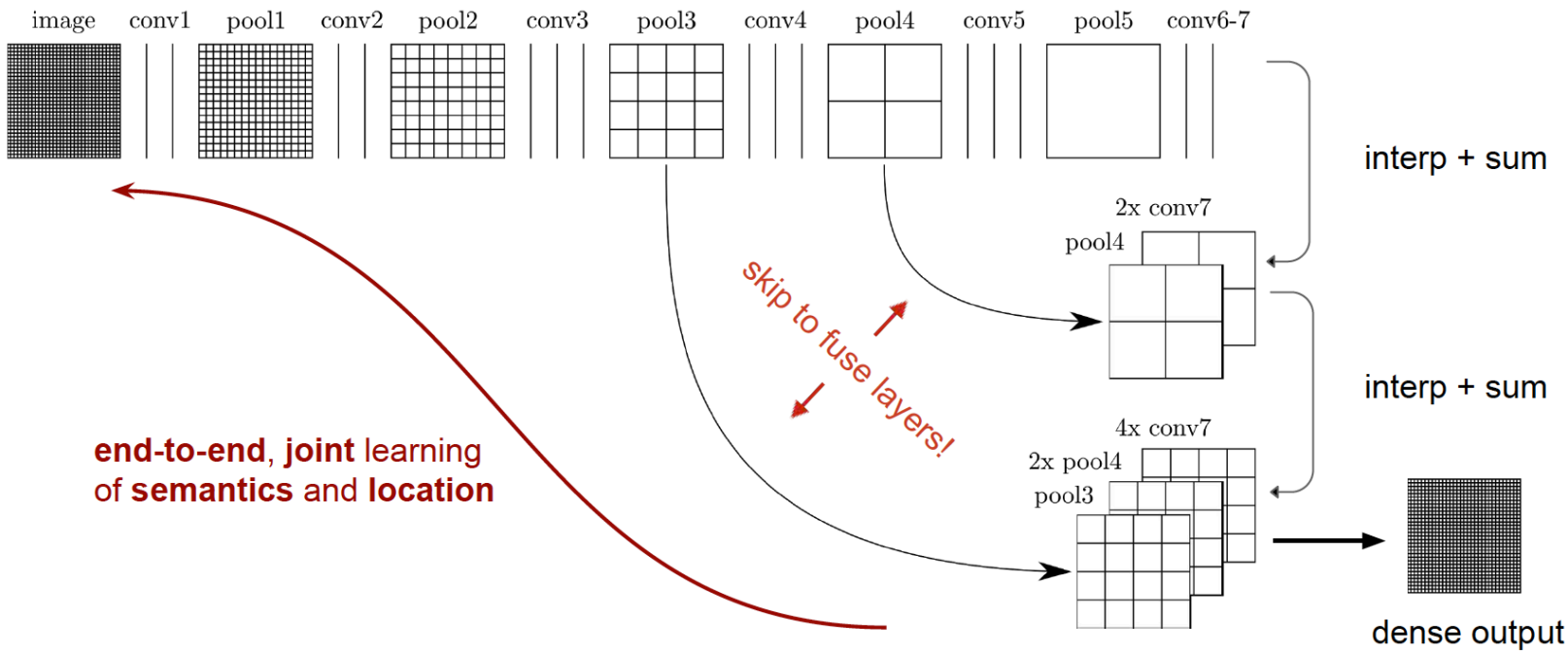
- 이전 Layer의 조밀한 정보(Dense features)를 합치면(fuse) 성능을 더욱 향상 시킬수 있다.



Slide credit: Jonathan Long
(https://goo.gl/S92NA9)

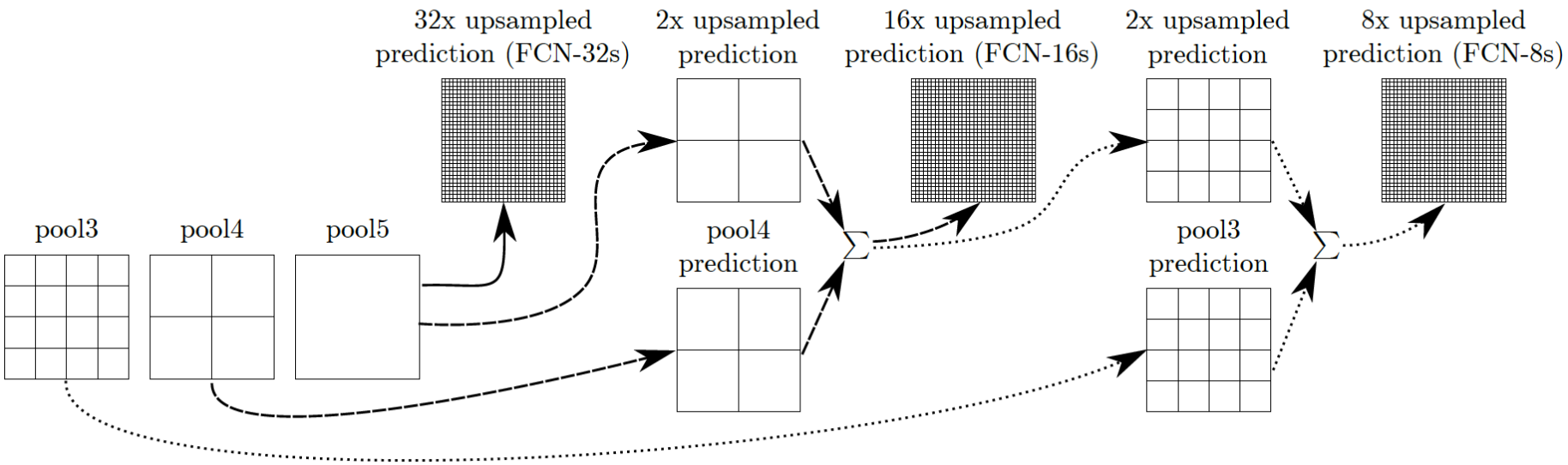
Fuse features

- 이전 Layer의 조밀한 정보(Dense features)를 합치면(fuse) 성능을 더욱 향상 시킬수 있다.



Slide credit: Jonathan Long
(https://goo.gl/S92NA9)

Skip Layers



Slide credit: Gordon Christie (https://goo.gl/phF7fy)

PASCAL VOC 2012 Dataset

- **P**attern **A**nalysis, **S**tatistical Modelling and **C**omput **A**tional **L**earning **V**isual **O**bject **C**lasses
- 아래와 같이 총 20개의 class가 존재

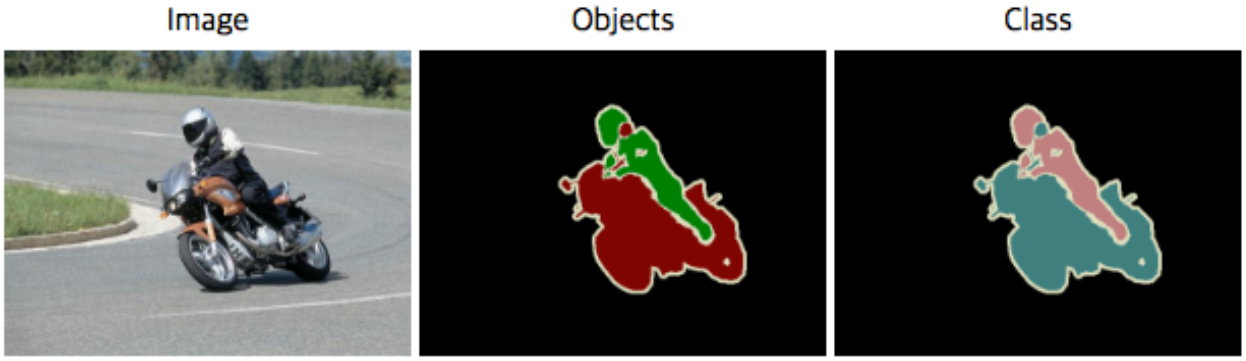
Person: person

Animal: bird, cat, cow, dog, horse, sheep

Vehicle: aeroplane, bicycle, boat, bus, car, motorbike, train

Indoor: bottle, chair, dining table, potted plant, sofa, tv/monitor

- http://host.robots.ox.ac.uk/pascal/VOC/voc2012/index.html



Metrics

- 모델의 성능을 평가하기 위해 다음과 같은 4가지 Metric을 사용한다.

- pixel accuracy: $\sum_i n_{ii} / \sum_i t_i$
- mean accuraccy: $(1/n_{\text{cl}}) \sum_i n_{ii}/t_i$
- mean IU: $(1/n_{\text{cl}}) \sum_i n_{ii} / \left(t_i + \sum_j n_{ji} - n_{ii}\right)$
- frequency weighted IU: $(\sum_k t_k)^{-1} \sum_i t_i n_{ii} / \left(t_i + \sum_j n_{ji} - n_{ii}\right)$

- 가장 많이 쓰이는 metric은 IU(Intersection over Union)-IoU라고도 표기함-

Intersection over Union(IoU) Metric

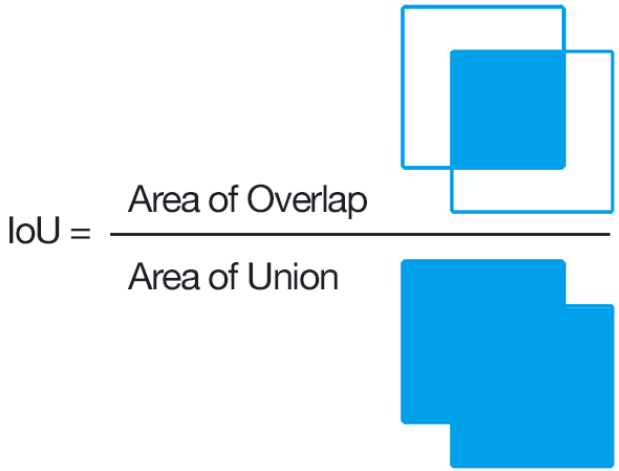
- IoU = true positive / (true positive + false positive + false negative)

n_cl : the number of classes

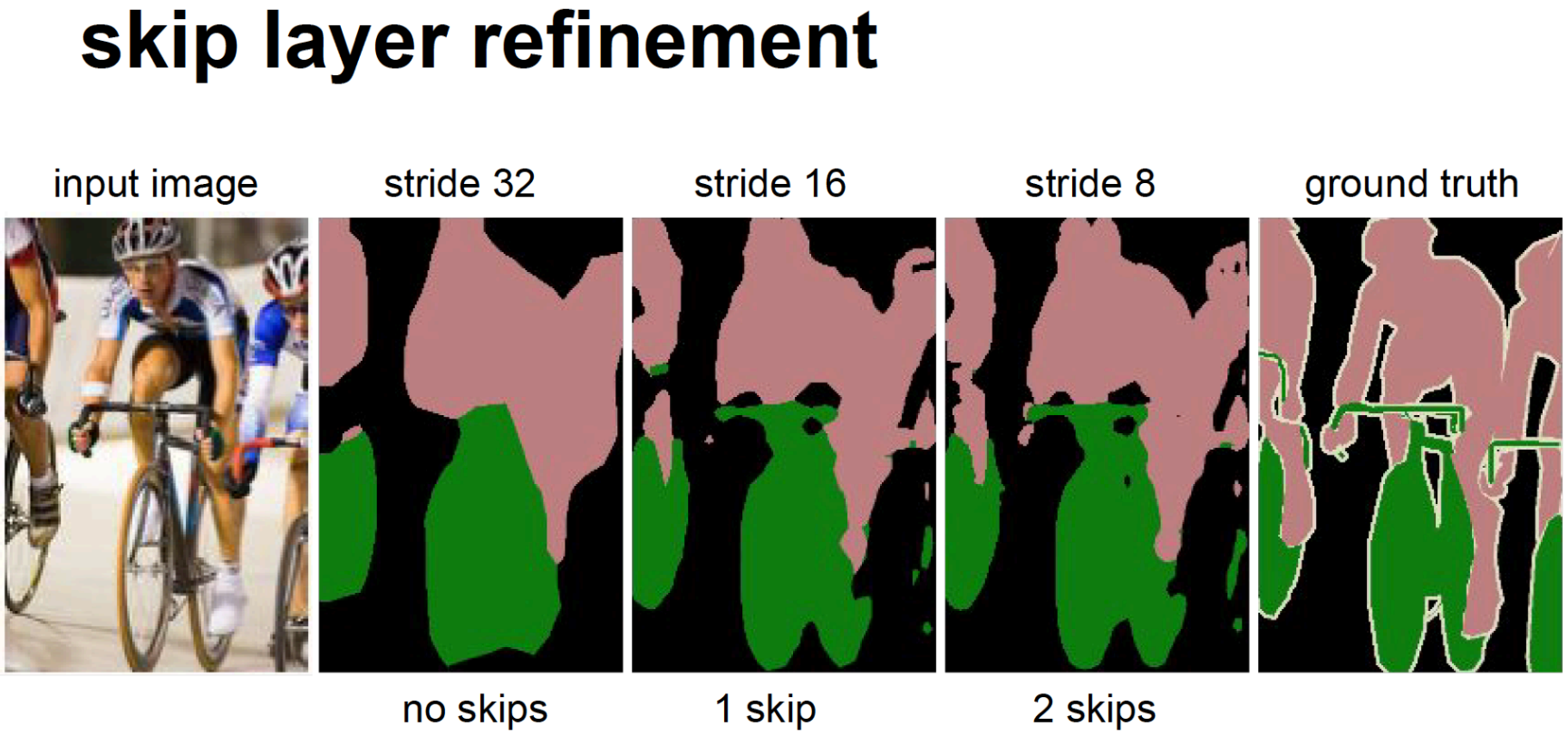
t_i : the total number of pixels in class *I*

n_ij : the number of pixels of class *i* predicted to belong to class *j*. So for class *i*:

- n_ii : the number of correctly classified pixels (true positives)
- n_ij : the number of pixels wrongly classified (false positives)
- n_ji : the number of pixels wrongly not classified (false negatives)



Skip Layers의 효과



Slide credit: Jonathan Long (https://goo.gl/S92NA9)

Skip Layers의 효과

Table 2. Comparison of skip FCNs on a subset[7] of PASCAL VOC 2011 segval. Learning is end-to-end, except for FCN-32s-fixed, where only the last layer is fine-tuned. Note that FCN-32s is FCN-VGG16, renamed to highlight stride.

	pixel acc.	mean acc.	mean IU	f.w. IU
FCN-32s-fixed	83.0	59.7	45.4	72.0
FCN-32s	89.1	73.3	59.4	81.4
FCN-16s	90.0	75.7	62.4	83.0
FCN-8s	**90.3**	**75.9**	**62.7**	**83.2**

Slide credit: Jonathan Long (https://goo.gl/S92NA9)

CNNs Architecture별 성능

- VGGNet이 Segmentation Task로 변환 했을때 가장 좋은 성능을 보여주었다.

Table 1. We adapt and extend three classification convnets. We compare performance by mean intersection over union on the validation set of PASCAL VOC 2011 and by inference time (averaged over 20 trials for a 500 × 500 input on an NVIDIA Tesla K40c). We detail the architecture of the adapted nets with regard to dense prediction: number of parameter layers, receptive field size of output units, and the coarsest stride within the net. (These numbers give the best performance obtained at a fixed learning rate, not best performance possible.)

	FCN-AlexNet	FCN-VGG16	FCN-GoogLeNet[4]
mean IU	39.8	**56.0**	42.5
forward time	50 ms	210 ms	59 ms
conv. layers	8	16	22
parameters	57M	134M	6M
rf size	355	404	907
max stride	32	32	32

다른 방법들과의 비교 – 정성적 비교

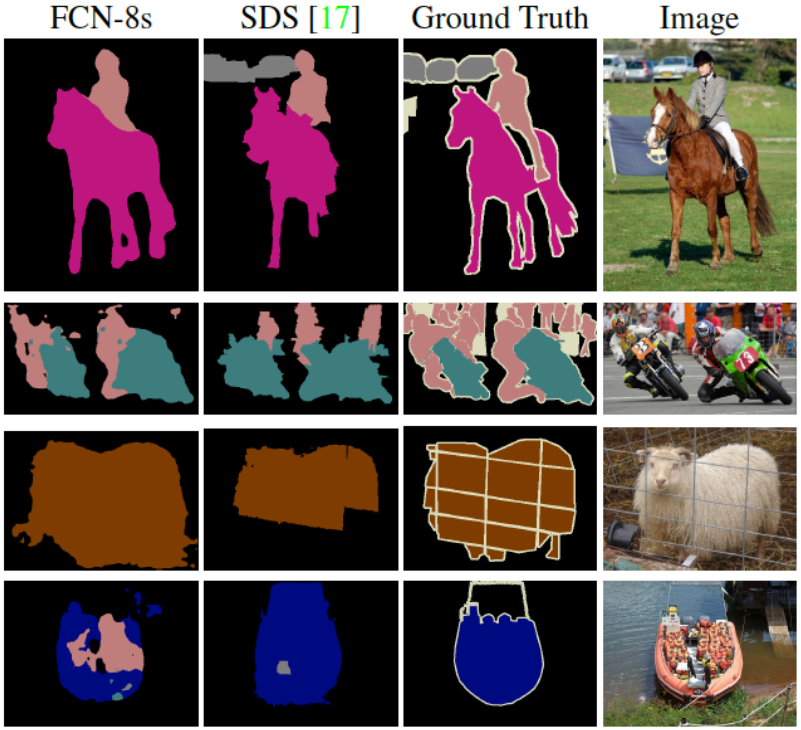


Figure 6. Fully convolutional segmentation nets produce state-of-the-art performance on PASCAL. The left column shows the output of our highest performing net, FCN-8s. The second shows the segmentations produced by the previous state-of-the-art system by Hariharan *et al*. [17]. Notice the fine structures recovered (first row), ability to separate closely interacting objects (second row), and robustness to occluders (third row). The fourth row shows a failure case: the net sees lifejackets in a boat as people.

다른 방법들과의 비교 – 정량적 비교

Table 3. Our fully convolutional net gives a 20% relative improvement over the state-of-the-art on the PASCAL VOC 2011 and 2012 test sets and reduces inference time.

	mean IU VOC2011 test	mean IU VOC2012 test	inference time
R-CNN [12]	47.9	-	-
SDS [17]	52.6	51.6	~ 50 s
FCN-8s	**62.7**	**62.2**	~ **175 ms**

TensorFlow를 이용한 Fully Convolutional Networks for Semantic Segmentation(FCN) 구현

- https://github.com/solaris33/dl_cv_tensorflow_10weeks/tree/master/week8/FCN.tensorflow

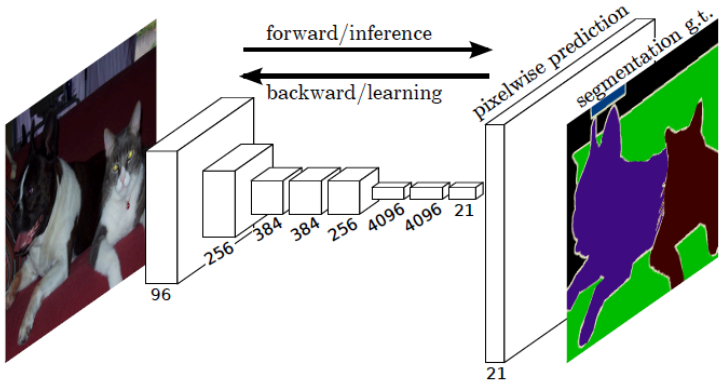
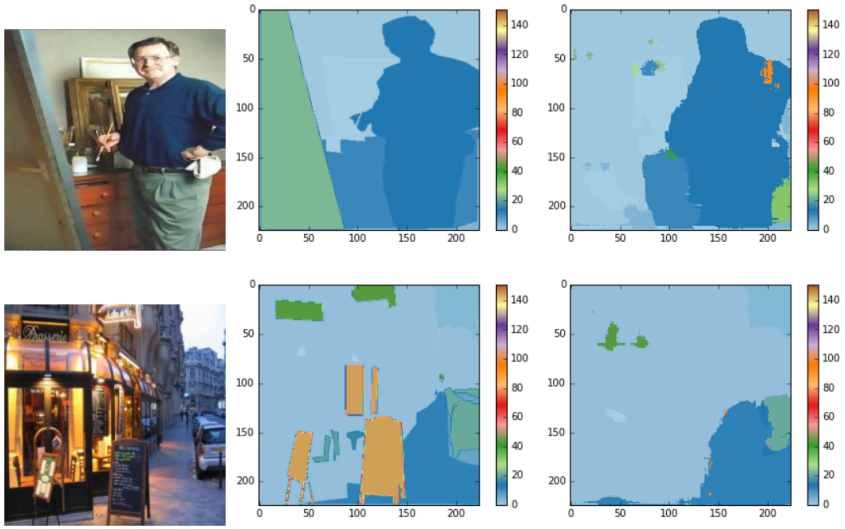


Figure 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.



논문 리뷰 - MULTI-SCALE CONTEXT AGGREGATION BY DILATED CONVOLUTIONS

- Yu, Fisher, and Vladlen Koltun. "Multi-scale context aggregation by dilated convolutions." arXiv preprint arXiv:1511.07122 (2015).

- **핵심 아이디어** : Dilated Convolution과 FCN 구조개선을 통해 더욱 좋은 성능의 Sematic Image Segmentation 결과를 보여줌

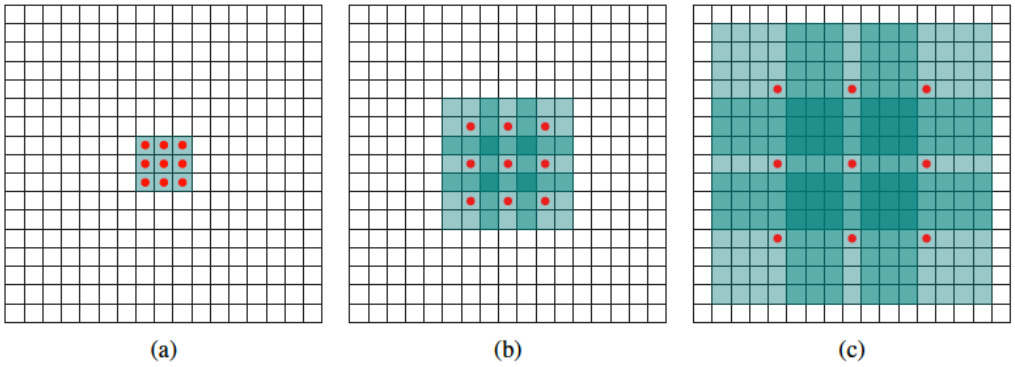


Figure 1: Systematic dilation supports exponential expansion of the receptive field without loss of resolution or coverage. (a) F_1 is produced from F_0 by a 1-dilated convolution; each element in F_1 has a receptive field of 3×3. (b) F_2 is produced from F_1 by a 2-dilated convolution; each element in F_2 has a receptive field of 7×7. (c) F_3 is produced from F_2 by a 4-dilated convolution; each element in F_3 has a receptive field of 15×15. The number of parameters associated with each layer is identical. The receptive field grows exponentially while the number of parameters grows linearly.

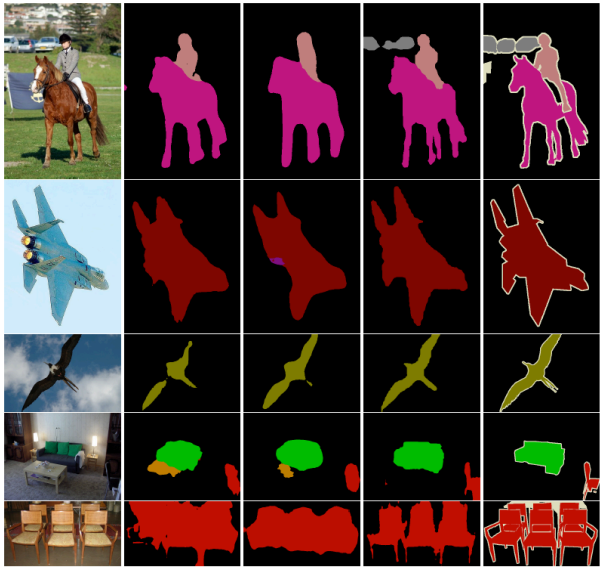


Figure 2: Semantic segmentations produced by different adaptations of the VGG-16 classification network. From left to right: (a) input image, (b) prediction by FCN-8s (Long et al., 2015), (c) prediction by DeepLab (Chen et al., 2015a), (d) prediction by our simplified front-end module, (e) ground truth.

Dilated Convolution

- **팽창된(Dilated) Convolution**
- Filter의 크기가 커지더라도 유지해야할 Parameter개수는 그대로 유지한다. (필터의 크기가 커진 부분은 모두 0으로 채운다.)
- **Dilated Convolution의 장점** : 해상도(Resolution) 손실없이 Receptive Field의 크기를 키울 수 있다.

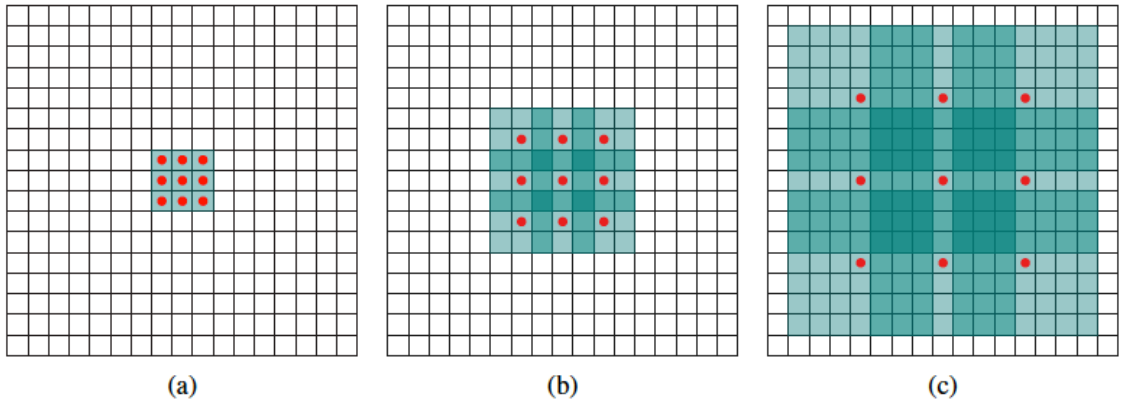
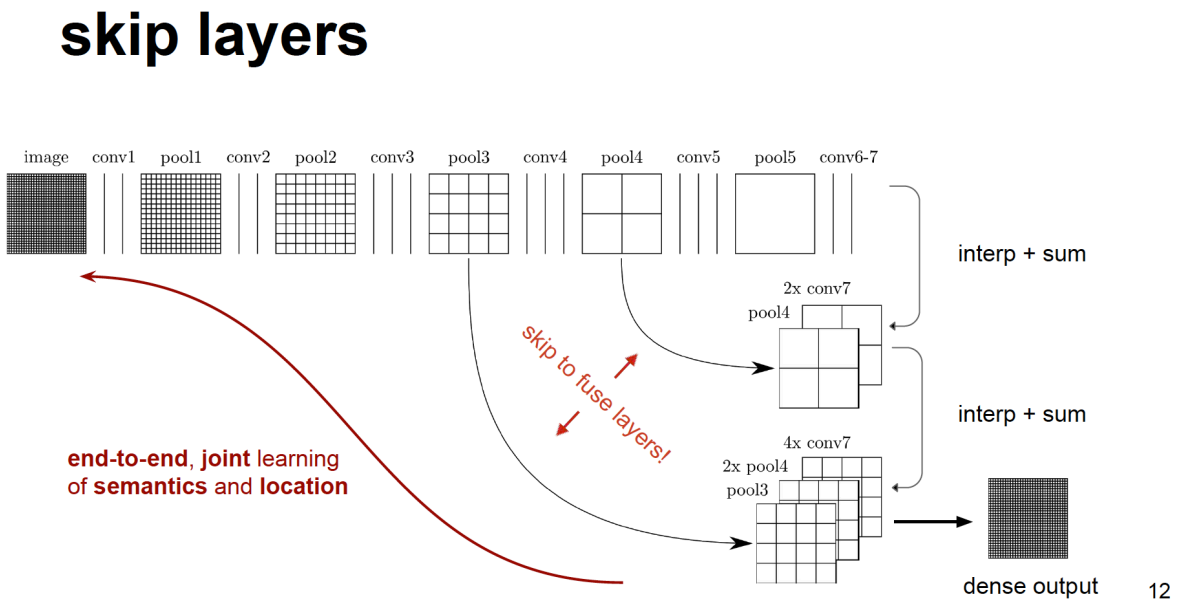


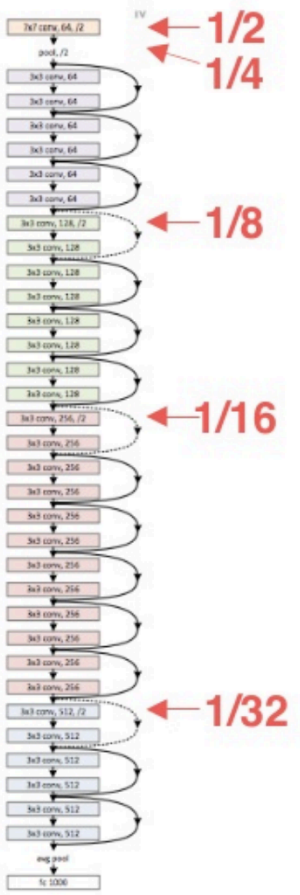
Figure 1: Systematic dilation supports exponential expansion of the receptive field without loss of resolution or coverage. (a) F_1 is produced from F_0 by a 1-dilated convolution; each element in F_1 has a receptive field of 3×3. (b) F_2 is produced from F_1 by a 2-dilated convolution; each element in F_2 has a receptive field of 7×7. (c) F_3 is produced from F_2 by a 4-dilated convolution; each element in F_3 has a receptive field of 15×15. The number of parameters associated with each layer is identical. The receptive field grows exponentially while the number of parameters grows linearly.

Front End Prediction Module

- We implemented and trained a front-end prediction module that takes a color image as input and produces C = 21 feature maps as output.
- pool4, pool5는 제거
- conv5, conv6에는 2-dilated convolution을 적용하고 마지막 conv7은 4-dilated convolution을 적용
- Training was performed by stochastic gradient descent (SGD) with mini-batch size 14, learning rate 10^-3, and momentum 0:9. The network was trained for 60K iterations.

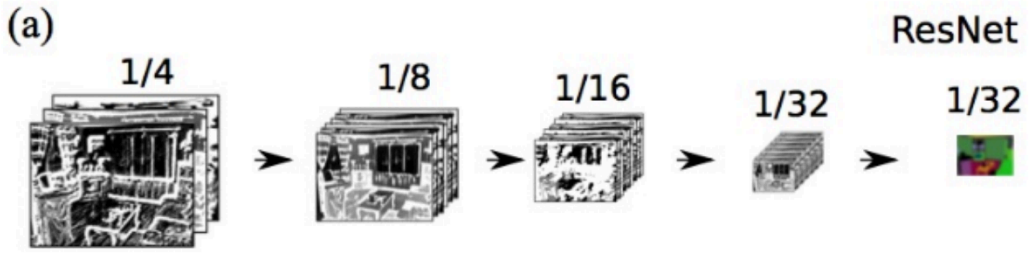


FCN vs Front End Module



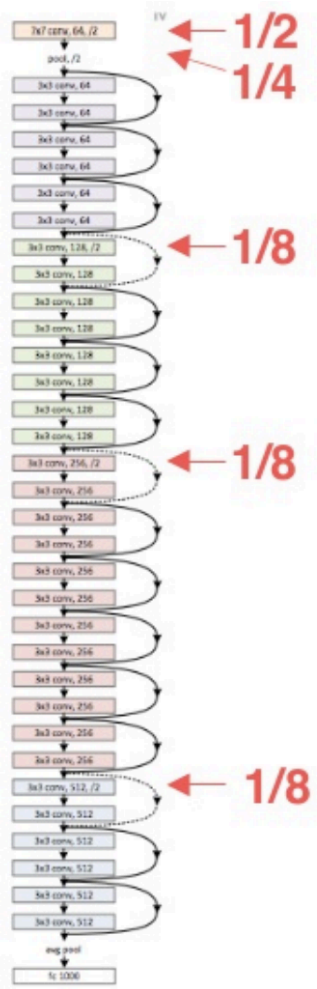
Dilated convolutions

- For example, the feature maps of ResNet are downsampled 5 times, and 4 times in the 5 are done by convolutions with stride of 2 (only the first one is by pooling with stride of 2)



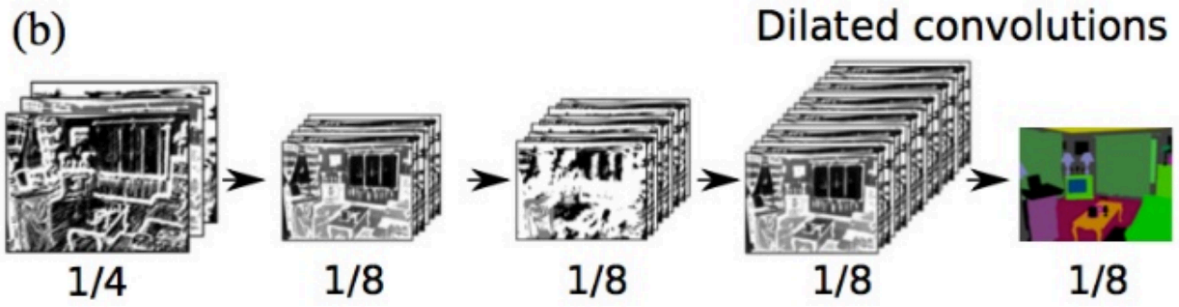
Slide credit: Shunta Saito (https://goo.gl/FMZLn5)

FCN vs Front End Module



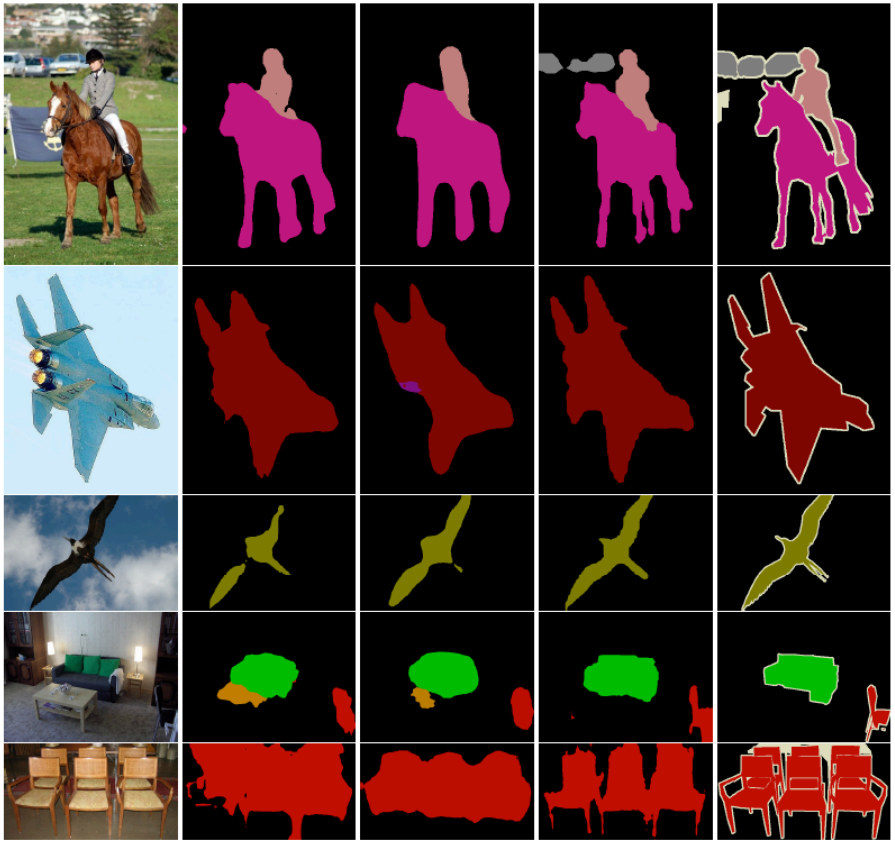
Dilated convolutions

- By using dilated convolutions instead of vanilla convolutions, the resolution after the first pooling can be kept as the same to the end



Slide credit: Shunta Saito (https://goo.gl/FMZLn5)

다른 방법들과의 비교 – 정성적 비교 (Front End Prediction Module만 사용)



(a) Image (b) FCN-8s (c) DeepLab (d) Our front end (e) Ground truth

Figure 2: Semantic segmentations produced by different adaptations of the VGG-16 classification network. From left to right: (a) input image, (b) prediction by FCN-8s (Long et al., 2015), (c) prediction by DeepLab (Chen et al., 2015a), (d) prediction by our simplified front-end module, (e) ground truth.

다른 방법들과의 비교 – 정량적 비교 (Front End Prediction Module만 사용)

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean IoU
FCN-8s	76.8	34.2	68.9	49.4	60.3	75.3	74.7	77.6	21.4	62.5	46.8	71.8	63.9	76.5	73.9	45.2	72.4	37.4	70.9	55.1	62.2
DeepLab	72	31	71.2	53.7	60.5	77	71.9	73.1	25.2	62.6	49.1	68.7	63.3	73.9	73.6	50.8	72.3	42.1	67.9	52.6	62.1
DeepLab-Msc	74.9	34.1	72.6	52.9	61.0	77.9	73.0	73.7	26.4	62.2	49.3	68.4	64.1	74.0	75.0	51.7	72.7	42.5	67.2	55.7	62.9
Our front end	**82.2**	**37.4**	**72.7**	**57.1**	**62.7**	**82.8**	**77.8**	**78.9**	**28**	**70**	**51.6**	**73.1**	**72.8**	**81.5**	**79.1**	**56.6**	**77.1**	**49.9**	**75.3**	**60.9**	**67.6**

Table 2: Our front-end prediction module is simpler and more accurate than prior models. This table reports accuracy on the VOC-2012 test set.

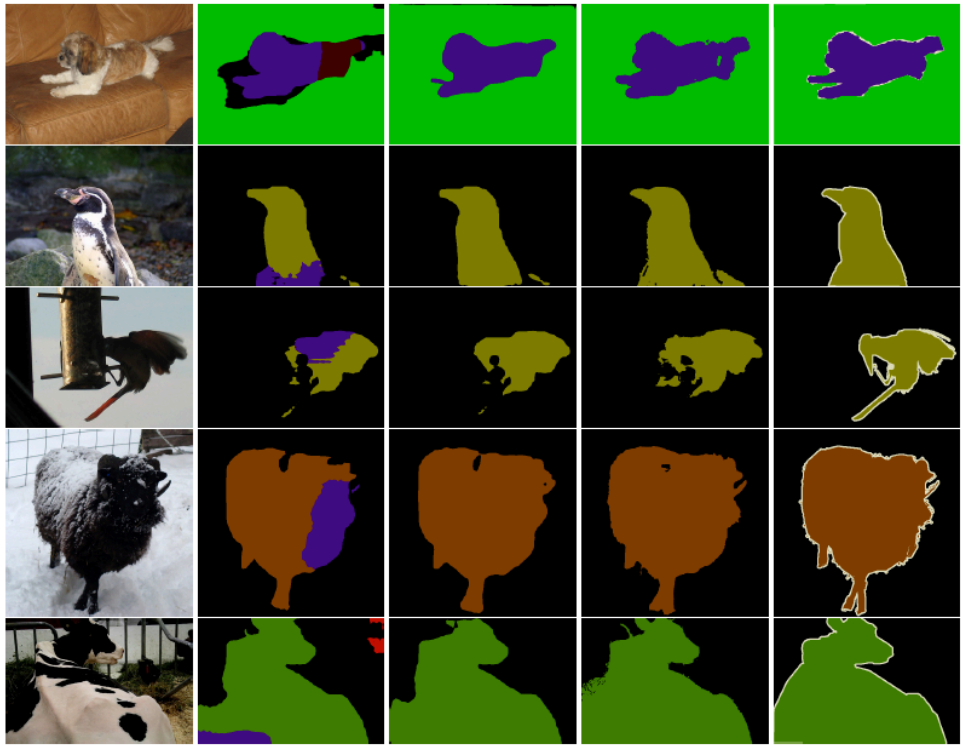
Context Module

- The context module is designed to increase the performance of dense prediction architectures by **aggregating multi-scale contextual information**.
- Each of these convolutions is followed by a **pointwise truncation max(·; 0).**
- Front End Module의 Output을 Context Module의 Input으로 넣고 실험을 진행 함

Layer	1	2	3	4	5	6	7	8
Convolution	3×3	3×3	3×3	3×3	3×3	3×3	3×3	1×1
Dilation	1	1	2	4	8	16	1	1
Truncation	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Receptive field	3×3	5×5	9×9	17×17	33×33	65×65	67×67	67×67
Output channels								
Basic	C	C	C	C	C	C	C	C
Large	$2C$	$2C$	$4C$	$8C$	$16C$	$32C$	$32C$	C

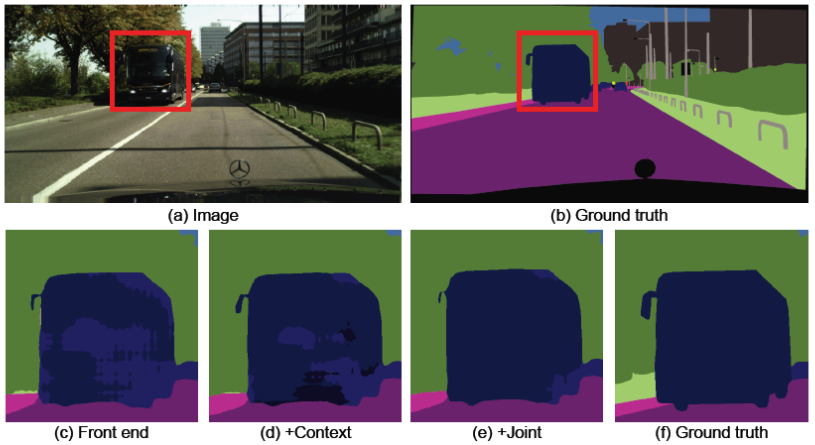
Table 1: Context network architecture. The network processes C feature maps by aggregating contextual information at progressively increasing scales without losing resolution.

다른 방법들과의 비교 – 정성적 비교 (Context Module까지 추가)



(a) Image (b) Front end (c) +Context (d) +CRF-RNN (e) Ground truth

Figure 3: Semantic segmentations produced by different models. From left to right: (a) input image, (b) prediction by the front-end module, (c) prediction by the large context network plugged into the front end, (d) prediction by the front end + context module + CRF-RNN, (e) ground truth.



(a) Image (b) Ground truth

(c) Front end (d) +Context (e) +Joint (f) Ground truth

Figure 5: Results produced by the Dilation10 model after different training stages. (a) Input image. (b) Ground truth segmentation. (c) Segmentation produced by the model after the first stage of training (front-end only). (d) Segmentation produced after the second stage, which trains the context module. (e) Segmentation produced after the third stage, in which both modules are trained jointly.

다른 방법들과의 비교 – 정량적 비교 (Context Module까지 추가)

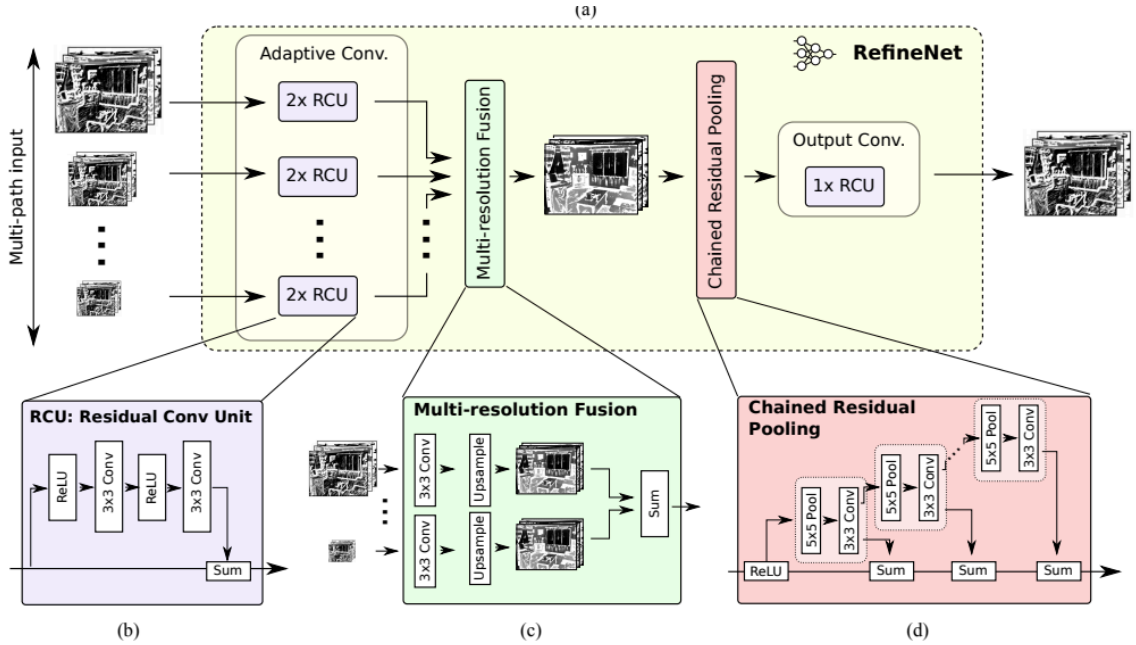
	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean IoU
Front end	86.3	38.2	76.8	**66.8**	63.2	87.3	78.7	82	33.7	76.7	53.5	73.7	76	76.6	83	**51.9**	77.8	44	79.9	**66.3**	69.8
Front + Basic	86.4	37.6	78.5	66.3	64.1	89.9	79.9	84.9	**36.1**	79.4	**55.8**	77.6	81.6	79	83.1	51.2	81.3	43.7	82.3	65.7	71.3
Front + Large	**87.3**	**39.2**	**80.3**	65.6	**66.4**	**90.2**	**82.6**	**85.8**	34.8	**81.9**	51.7	**79**	**84.1**	**80.9**	**83.2**	51.2	**83.2**	**44.7**	**83.4**	65.6	**72.1**
Front end + CRF	89.2	38.8	80	**69.8**	63.2	88.8	80	85.2	33.8	80.6	55.5	77.1	80.8	77.3	84.3	**53.1**	80.4	45	80.7	**67.9**	71.6
Front + Basic + CRF	89.1	38.7	81.4	67.4	65	91	81	86.7	**37.5**	81	**57**	79.6	83.6	79.9	**84.6**	52.7	83.3	44.3	82.6	67.2	72.7
Front + Large + CRF	**89.6**	**39.9**	**82.7**	66.7	**67.5**	**91.1**	**83.3**	**87.4**	36	**83.3**	52.5	**80.7**	**85.7**	**81.8**	84.4	52.6	**84.4**	**45.3**	**83.7**	66.7	**73.3**
Front end + RNN	88.8	38.1	80.8	**69.1**	65.6	89.9	79.6	85.7	36.3	83.6	57.3	77.9	83.2	77	**84.6**	**54.7**	82.1	**46.9**	80.9	66.7	72.5
Front + Basic + RNN	89	38.4	82.3	67.9	65.2	91.5	80.4	87.2	**38.4**	82.1	**57.7**	79.9	85	79.6	84.5	53.5	84	45	82.8	66.2	73.1
Front + Large + RNN	**89.3**	**39.2**	**83.6**	67.2	**69**	**92.1**	**83.1**	**88**	**38.4**	**84.8**	55.3	**81.2**	**86.7**	**81.3**	84.3	53.6	**84.4**	45.8	**83.8**	**67**	**73.9**

Table 3: Controlled evaluation of the effect of the context module on the accuracy of three different architectures for semantic segmentation. Experiments performed on the VOC-2012 validation set. Validation images were not used for training. Top: adding the context module to a semantic segmentation front end with no structured prediction (Long et al., 2015). The basic context module increases accuracy, the large module increases it by a larger margin. Middle: the context module increases accuracy when plugged into a front-end + dense CRF configuration (Chen et al., 2015a). Bottom: the context module increases accuracy when plugged into a front-end + CRF-RNN configuration (Zheng et al., 2015).

논문 리뷰 - Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation.

- Lin, Guosheng, et al. "Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation." arXiv preprint arXiv:1611.06612 (2016).

- **핵심 아이디어** : Multi-Path Fusion을 수행하는 RefineNet 구조를 제안해서 더욱 좋은 성능의 Sematic Image Segmentation 결과를 보여줌



기존 방법과 RefineNet 비교

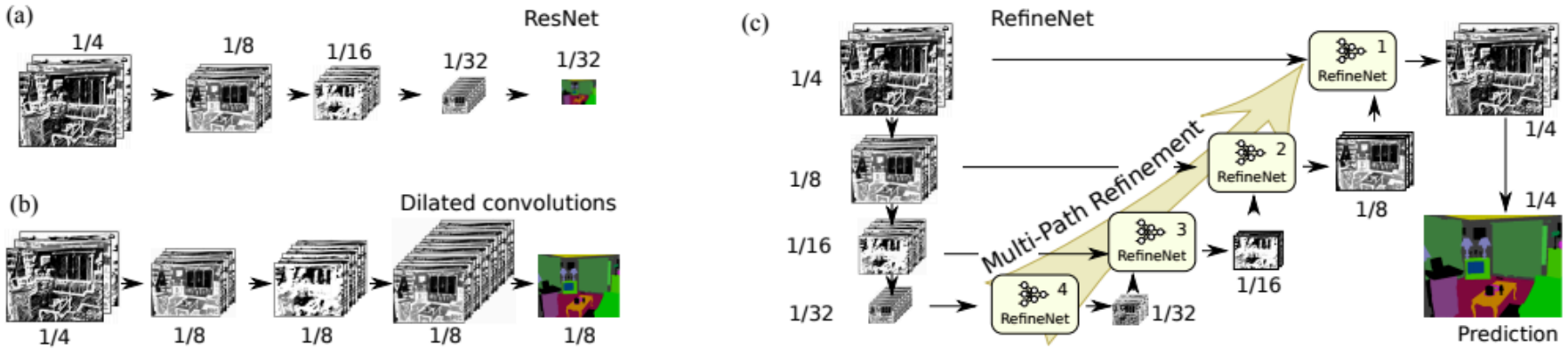


Figure 2. Comparison of fully convolutional approaches for dense classification. Standard multi-layer CNNs, such as ResNet *(a)* suffer from downscaling of the feature maps, thereby losing fine structures along the way. Dilated convolutions *(b)* remedy this shortcoming by introducing atrous filters, but are computationally expensive to train and quickly reach memory limits even on modern GPUs. Our proposed architecture that we call RefineNet *(c)* exploits various levels of detail at different stages of convolutions and fuses them to obtain a high-resolution prediction without the need to maintain large intermediate feature maps. The details of the RefineNet block are outlined in Sec. 3 and illustrated in Fig 3.

RefineNet Architecture

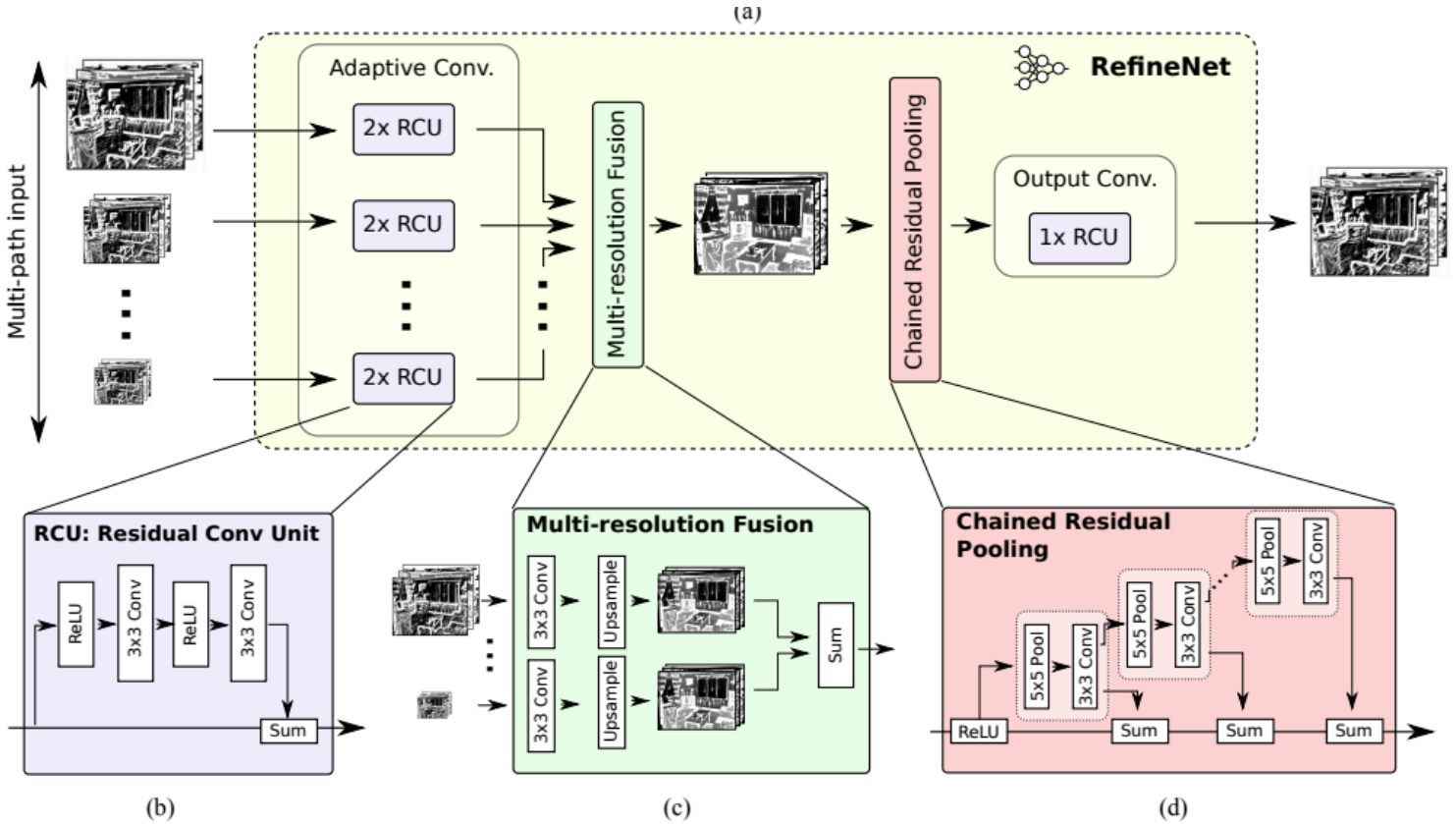


Figure 3. The individual components of our multi-path refinement network architecture RefineNet. Components in RefineNet employ residual connections with identity mappings. In this way, gradients can be directly propagated within RefineNet via local residual connections, and also directly propagate to the input paths via long-range residual connections, and thus we achieve effective end-to-end training of the whole system.

다양한 형태로 구현한 RefineNet

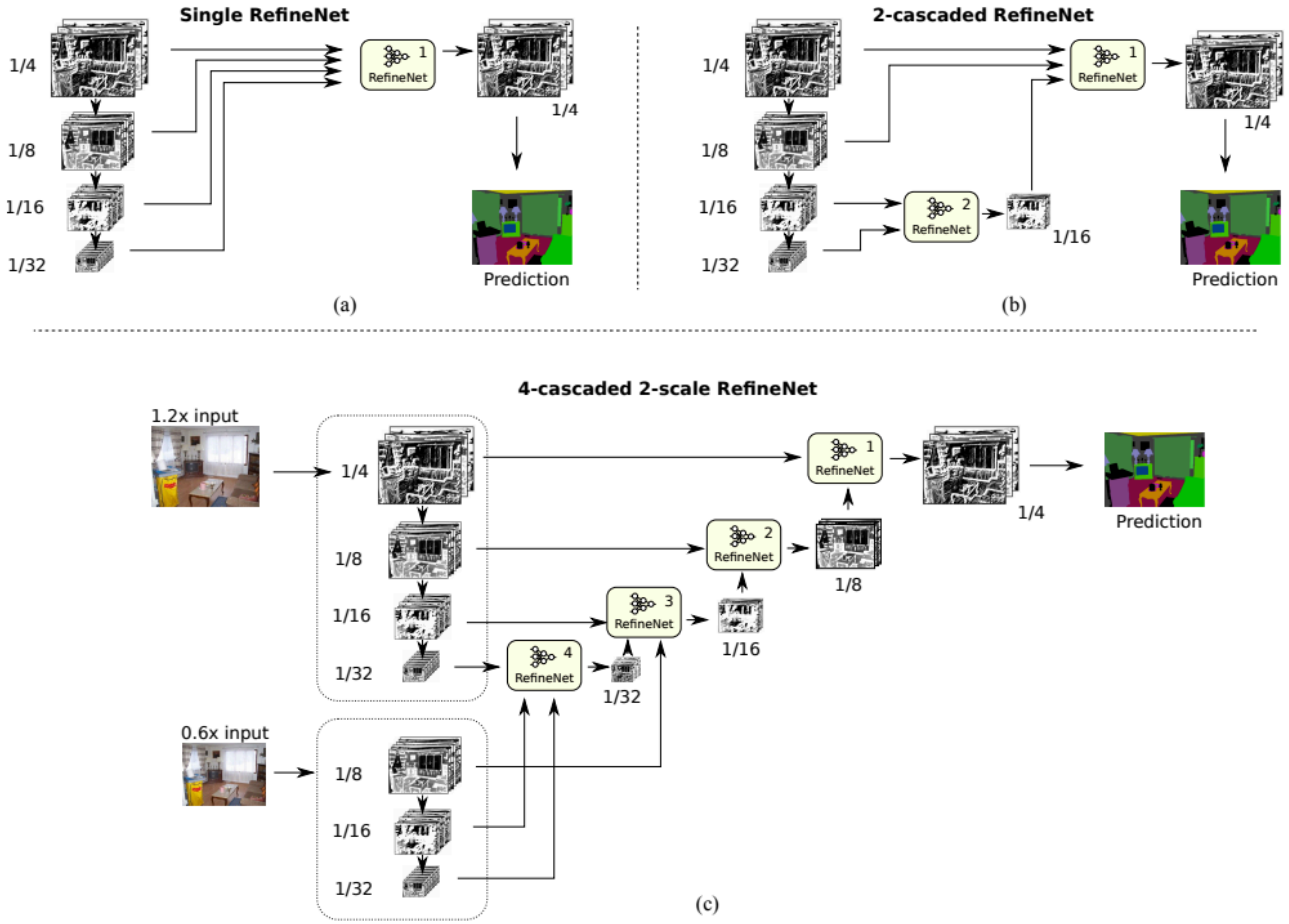
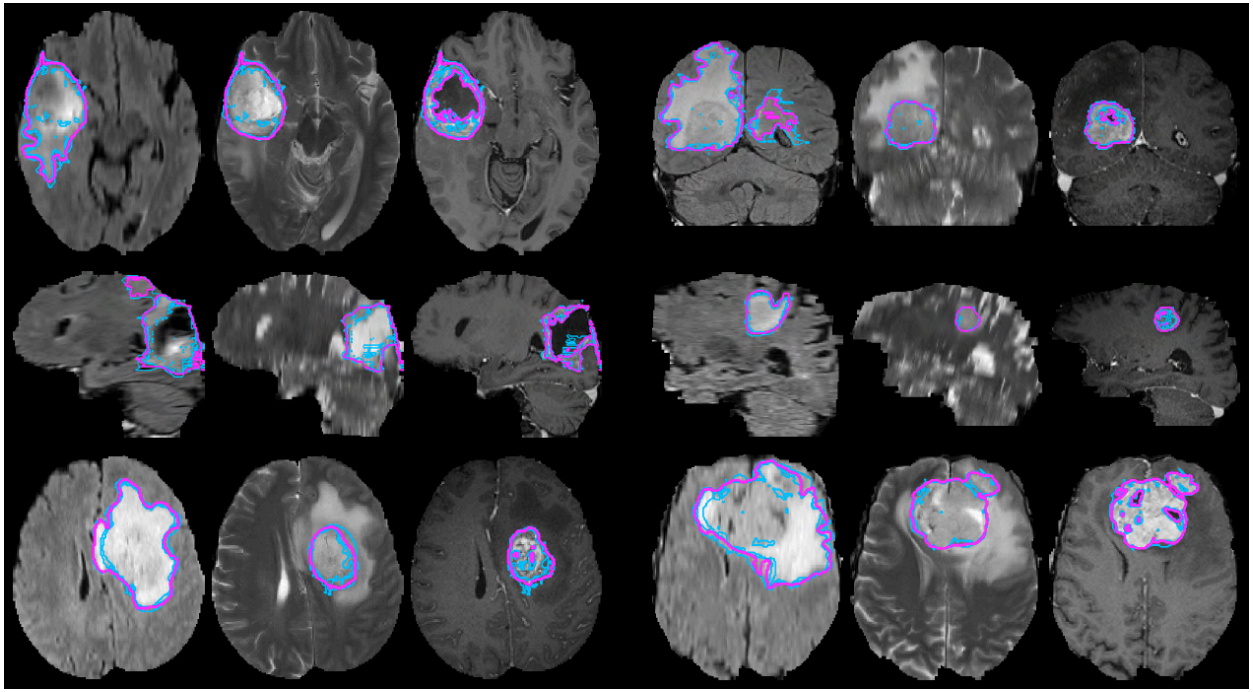


Figure 7. Illustration of 3 variants of our network architecture: (a) single RefineNet, (b) 2-cascaded RefineNet and (c) 4-cascaded RefineNet with 2-scale ResNet. Note that our proposed RefineNet block can seamlessly handle different numbers of inputs of arbitrary resolutions and dimensions without any modification.

Outline

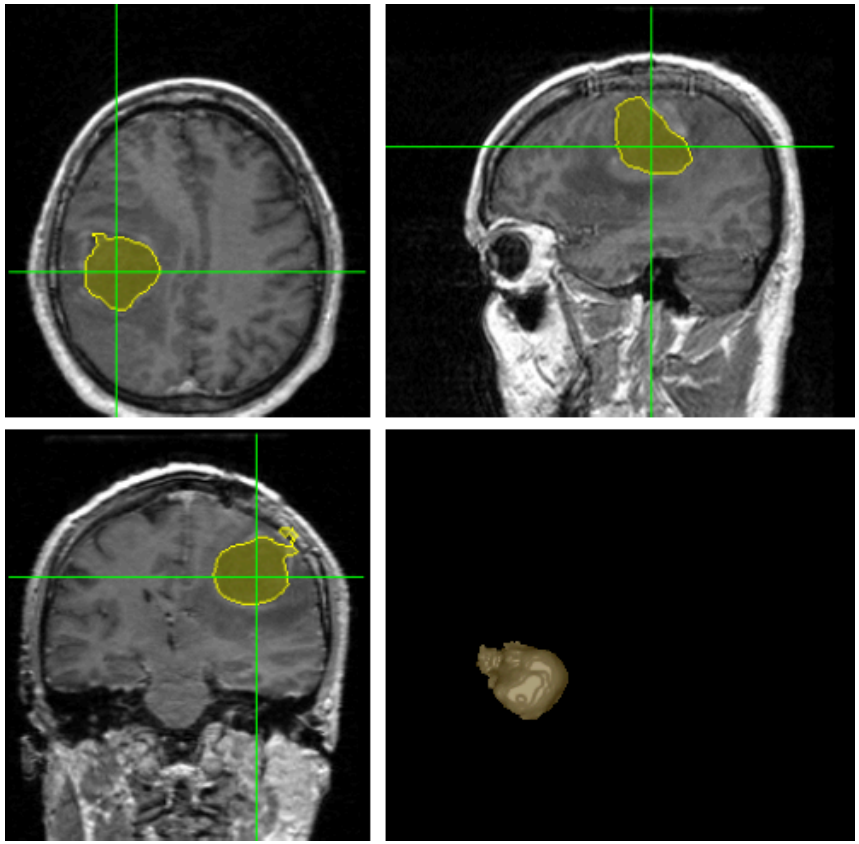
- BraTS Database 소개
- 논문 리뷰 - Brain Tumor Segmentation with Deep Neural Networks
- Fully Convolutional Networks for Semantic Segmentation(FCN)을 이용한 Brain Tumor Segmentation

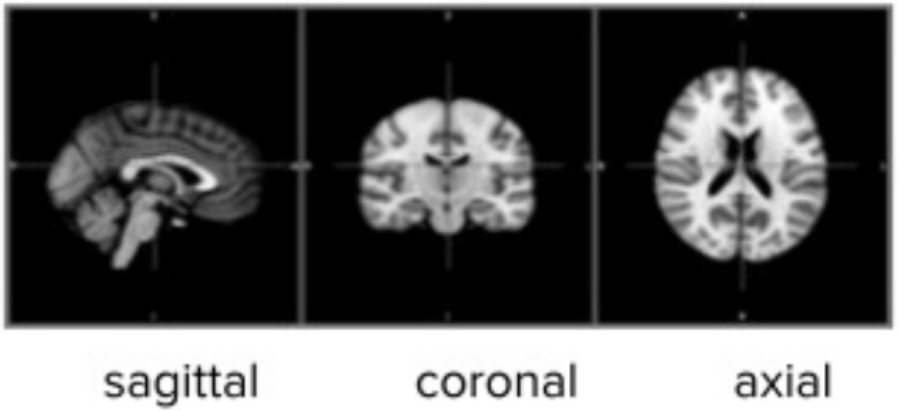
Multimodal Brain Tumor Segmentation Challenge(BraTS)



Multimodal Image Data

- Brain Tumor segmentation을 위한 BraTS 데이터는 3차원 형태로 주어진다.





BraTS 데이터 구성

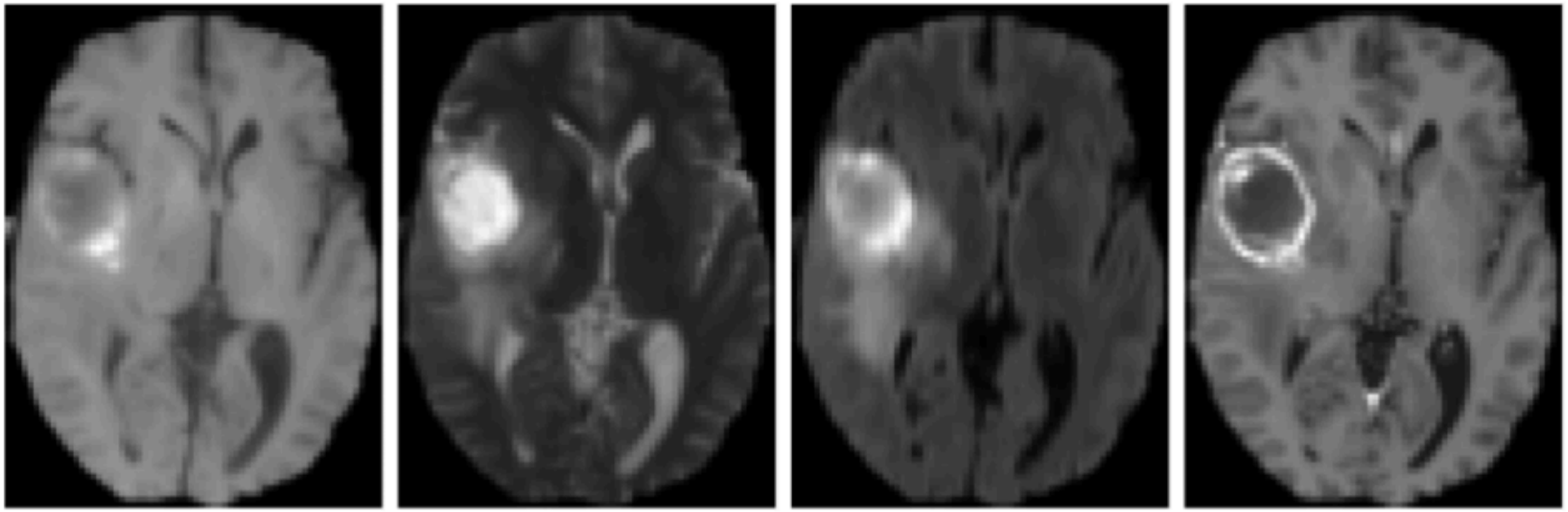
- 데이터 형태 : .mha 파일 형식의 3차원 Multimodal MRI 영상. HGG(High-Grade Glioma-상대적으로 더 심각한 종양-), LGG(Low-Grade Glioma-상대적으로 덜 심각한 종양-) 두가지 타입의 영상이 존재

- Ground Truth : 아래와 같은 5개의 Label로 구성되어 있다.

```
1 for necrosis
2 for edema
3 for non-enhancing tumor
4 for enhancing tumor
0 for everything else
```

T1, T2, T1c, FLAIR

- T1, T2, T1c, FLAIR 4가지 기법으로 촬영한 MRI 이미지가 제공된다.



(a) T1 (b) T2 (c) FLAIR (d) T1c

논문 리뷰 - Brain Tumor Segmentation with Deep Neural Networks

- Havaei, Mohammad, et al. "Brain tumor segmentation with deep neural networks." *Medical image analysis* 35 (2017): 18-31.

- **핵심 아이디어** : Brain Tumor Segmentation을 위한 새로운 Deep Neural Networks(DNN) 구조를 제안(Two-path way CNN, Cascaded architectures)

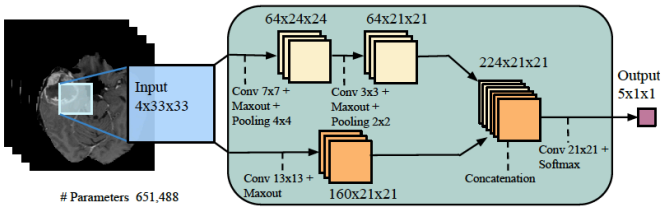
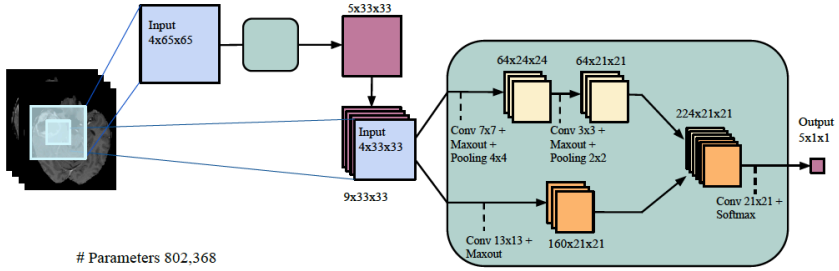


Figure 2: Two-pathway CNN architecture (TwoPathCNN). The figure shows the input patch going through two paths of convolutional operations. The feature-maps in the local and global paths are shown in yellow and orange respectively. The convolutional layers used to produce these feature-maps are indicated by dashed lines in the figure. The green box embodies the whole model which in later architectures will be used to indicate the TwoPathCNN.



(a) Cascaded architecture, using input concatenation (InputCascadeCNN).

Convolutional Neural Networks(CNNs)

- Maxout activtion을 사용한 CNNs을 구조를 이용

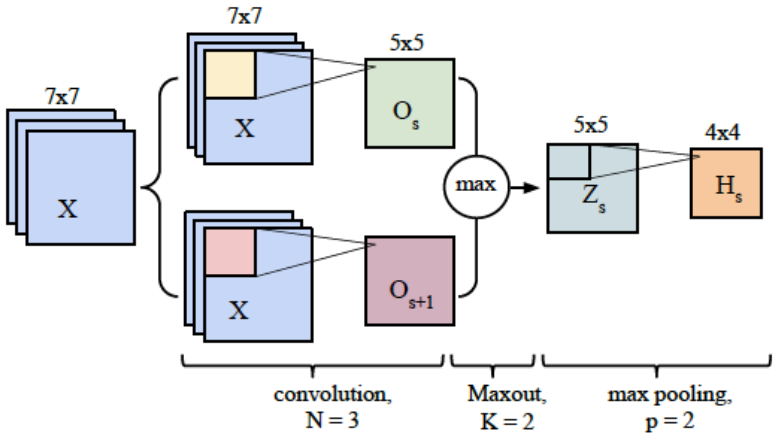


Figure 1: A single convolution layer block showing computations for a single feature map. The input patch (here 7×7), is convolved with series of kernels (here 3×3) followed by Max-out and max-pooling.

$$Z_{s,i,j} = \max\left\{O_{s,i,j}, O_{s+1,i,j}, \ldots, O_{s+K-1,i,j}\right\}$$

Maxout Networks

- Goodfellow, Ian J., et al. "Maxout networks." *arXiv preprint arXiv:1302.4389* (2013).
- Sigmoid, ReLU 같은 activation function을 대체한 activation 방법.
- K개의 output을 중에서 max값을 activation으로 취한다.

$$h_i(x) = \max_{j \in [1,k]} (z_{ij})$$
$$z_{ij} = x^T W_{\ldots ij} + b_{ij}$$

- is a **piecewise linear** approximation to an arbitrary convex function

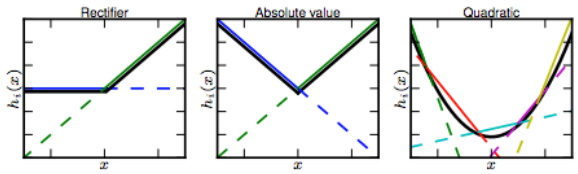
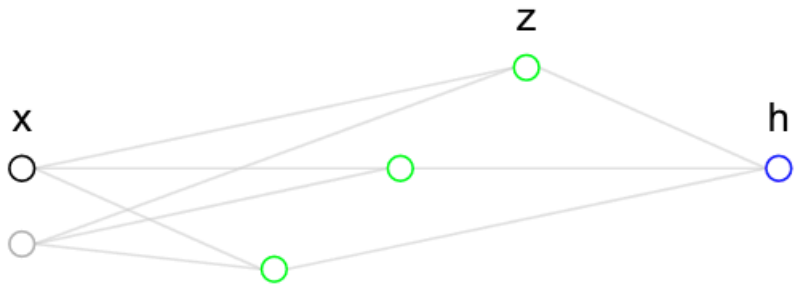


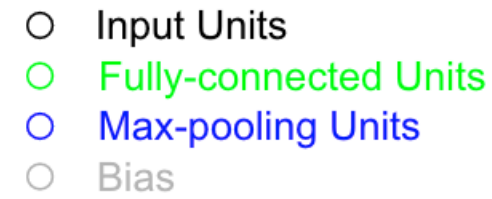
Figure 1. Graphical depiction of how the maxout activation function can implement the rectified linear, absolute value rectifier, and approximate the quadratic activation function. This diagram is 2D and only shows how maxout behaves with a 1D input, but in multiple dimensions a maxout unit can approximate arbitrary convex functions.

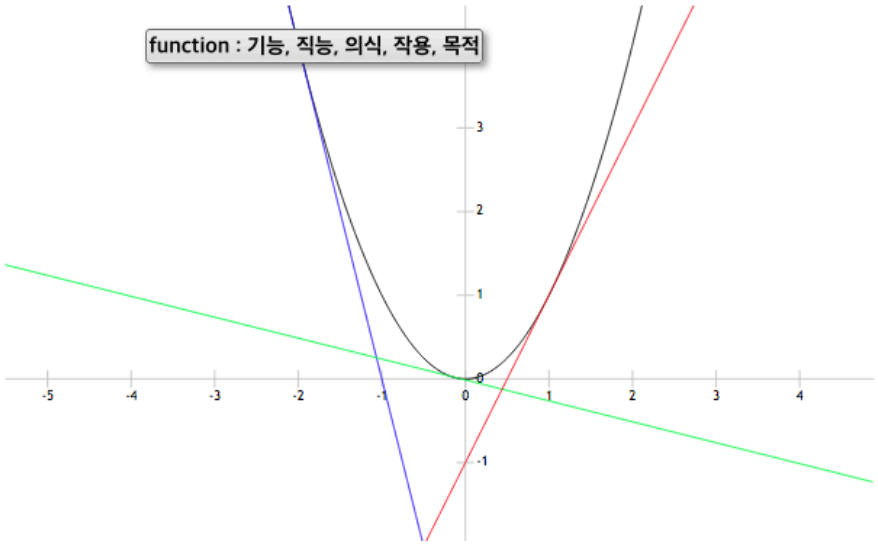
- **Maxout의 장점** : Dropout과 결합했을때의 성능이 ReLU 같은 non-linear activation보다 좋다.

Maxout Networks – Simple Example

- 3개의 linear function(hidden unit)으로 f(x) = x^2을 approximation하는 경우를 가정해보면 아래 그림과 같다.







Slide credit: (https://goo.gl/YPpZy7)

Two-pathway CNN

- 7x7 receptive fields and 13x13 receptive fields. We refer to these streams as the **local pathway** and the **global pathway**, respectively.

- The motivation for this architectural choice is that we would like the prediction of the label of a pixel to be influenced by two aspects: **the visual details of the region around that pixel** and **its larger "context"**

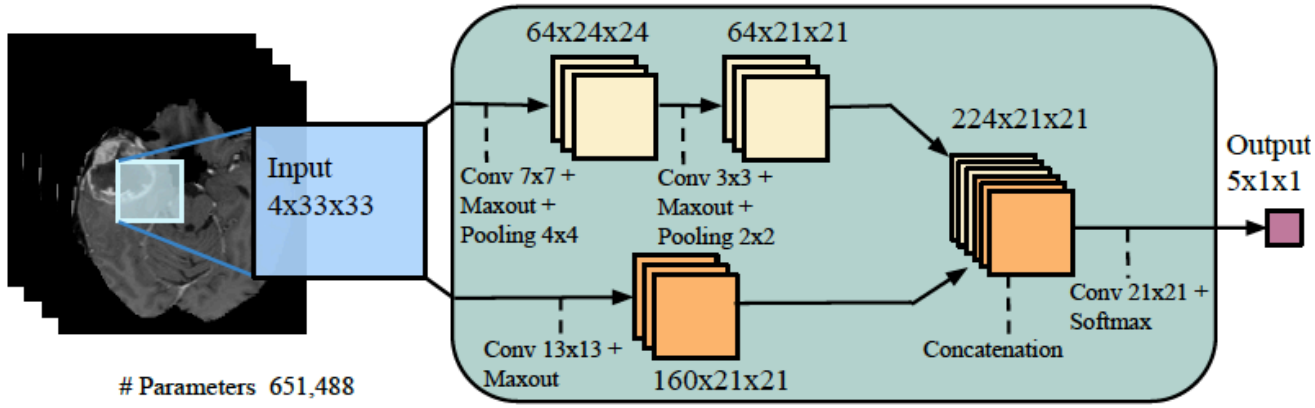


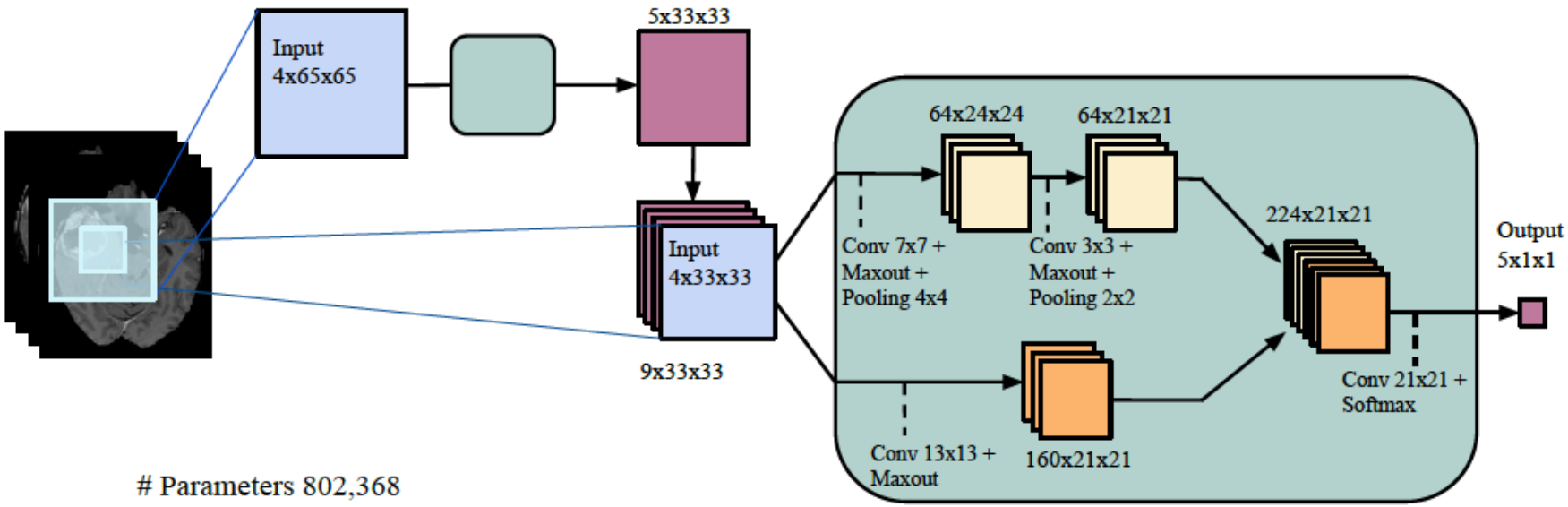
Figure 2: Two-pathway CNN architecture (TwoPathCNN). The figure shows the input patch going through two paths of convolutional operations. The feature-maps in the local and global paths are shown in yellow and orange respectively. The convolutional layers used to produce these feature-maps are indicated by dashed lines in the figure. The green box embodies the whole model which in later architectures will be used to indicate the TwoPathCNN.

Cascaded architectures

- 2개의 CNN을 구성하고 첫번째 CNNs의 Output(Segmentation Result)을 2번째 CNNs의 Input에 추가한다.
- Moreover, we use the same two-pathway structure for both CNNs.
- **InputCascadeCNN**, **LocalCascadeCNN**, **MFCascadeCNN** 세가지 cascade architecture를 제안
- **Cascaded architectures의 장점** : Segementation을 진행할때 이웃한 위치의 Label이 무엇인지를 함께 고려하면서 Segmentation이 가능함. (CRF를 사용할때의 장점을 CNN architecture에 포함시킴)

InputCascadeCNN

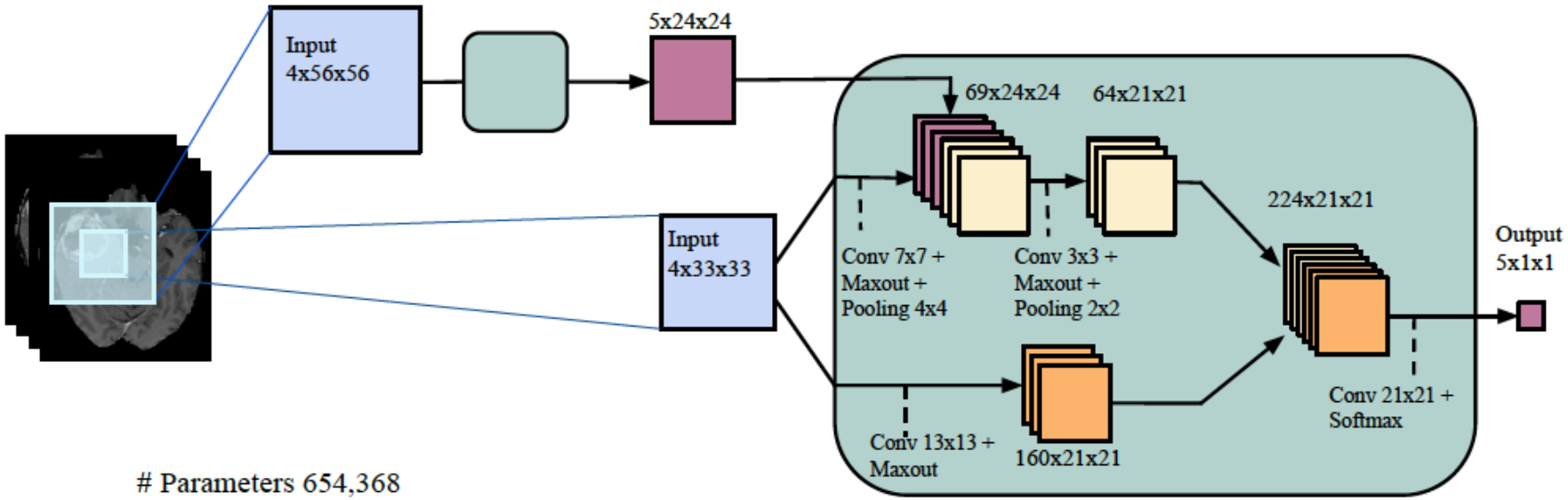
- **InputCascadeCNN** : In this architecture, we provide the first CNN's output directly as input to the second CNN. They are thus **simply treated as additional image channels** of the input patch.



(a) Cascaded architecture, using input concatenation (INPUTCASCADECNN).

LocalCascadeCNN

- **LocalCascadeCNN** : In this architecture, we move up one layer in the **local pathway** and perform concatenation to its first hidden layer, in the second CNN.

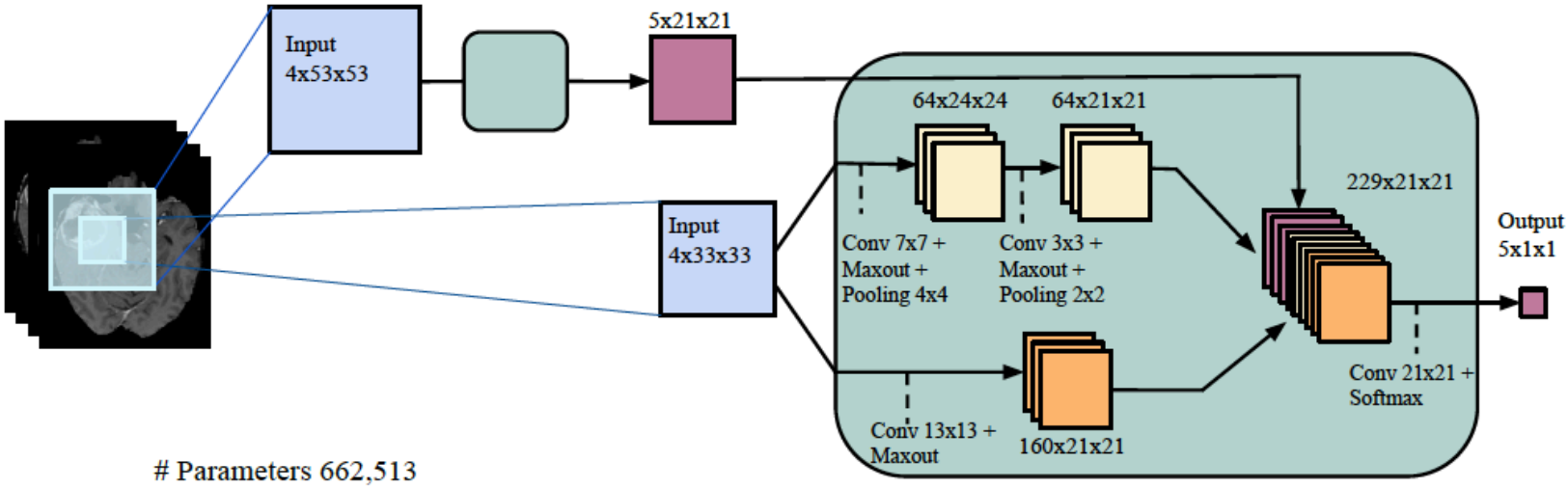


(b) Cascaded architecture, using local pathway concatenation (LocalCascadeCNN).

MFCascadeCNN

- **MFCascadeCNN** : This architecture is interesting, as it is similar to the computations made by one pass of mean-field inference [46] in a CRF whose pairwise potential functions are the weights in the output kernels.

- From this view, the output of the first CNN is the first iteration of mean-field, while the output of the second CNN would be the second iteration.



(c) Cascaded architecture, using pre-output concatenation, which is an architecture with properties similar to that of learning using a limited number of mean-field inference iterations in a CRF (MFCascadeCNN).

Experimental Setting

- BraTS 2013 데이터셋으로 실험을 진행

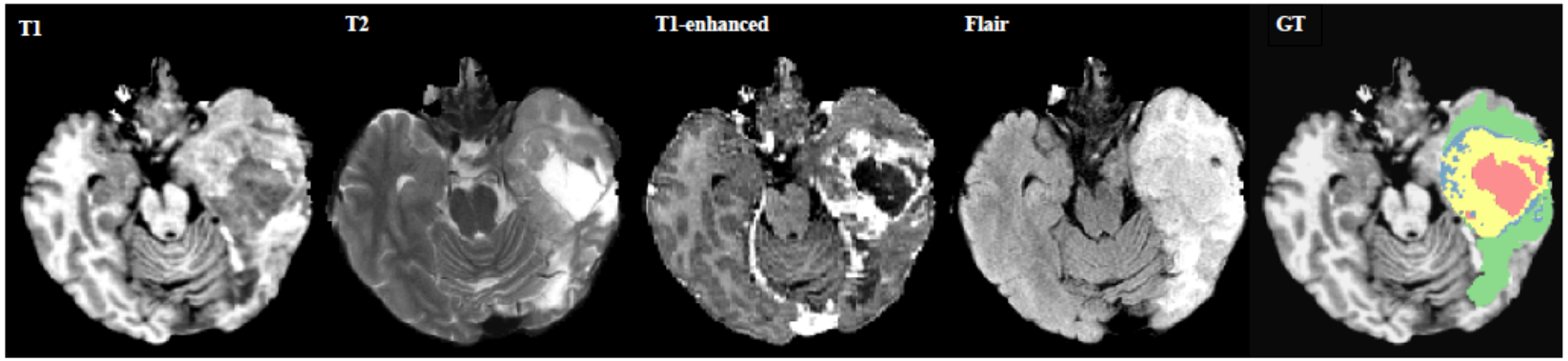


Figure 4: The first four images from left to right show the MRI modalities used as input channels to various CNN models and the fifth image shows the ground truth labels where edema, enhanced tumor, necrosis, non-enhanced tumor.

Two-phase training

- **Two-phase training** : Brain Tumor Data의 Label간의 **Imbalance**를 보정하기 위한 기법

- Brain tumor segmentation is a highly data imbalanced problem where the **healthy voxels (i.e. label 0) comprise 98% of total voxels**. From the remaining 2% pathological voxels, 0.18% belongs to necrosis (label 1), 1.1% to edema (label 2), 0.12% to non-enhanced (label 3) and 0.38% to enhanced tumor (label 4).

- 따라서 First Phase로 전체 데이터셋으로 Neural Networks를 학습한다.

- 그 다음에, Second Phase로 마지막 Layer를 제외한 Networks를 Freezing하고 **마지막 Layer만 Label간의 Imbalance가 보정된 데이터셋으로 re-training**한다.

- This way we get the best of both worlds: most of the capacity **(the lower layers)** is used in a balanced way to account for **the diversity in all of the classes**, while the **output probabilities are calibrated correctly** (thanks to the **re-training of the output layer** with the natural frequencies of classes in the data).

Experimental Result

- 다양한 Two-pathway CNN 구조로 실험을 진행 함.
- AverageCNN – LocalPathCNN과 GlobalPath CNN을 독립적으로 학습하고 (joint하게 학습하지 않고) 평균을 취함
- The second training phase is noted by appending '*' to the architecture name.

Table 1: Performance of the TwoPathCNN model and variations. The second phase training is noted by appending '*' to the architecture name. The 'Rank' column represents the ranking of each method in the online score board at the time of submission.

Rank	Method	Dice			Specificity			Sensitivity		
		Complete	Core	Enhancing	Complete	Core	Enhancing	Complete	Core	Enhancing
4	TwoPathCNN*	0.85	0.78	0.73	0.93	0.80	0.72	0.80	0.76	0.75
9	LocalPathCNN*	0.85	0.74	0.71	0.91	0.75	0.71	0.80	0.77	0.73
10	AveragCNN*	0.84	0.75	0.70	0.95	0.83	0.73	0.77	0.74	0.73
14	GlobalPathCNN*	0.82	0.73	0.68	0.93	0.81	0.70	0.75	0.65	0.70
14	TwoPathCNN	0.78	0.63	0.68	0.67	0.50	0.59	0.96	0.89	0.82
15	LocalPathCNN	0.77	0.64	0.68	0.65	0.52	0.60	0.96	0.87	0.80

Experimental Result

- 다양한 Cascaded architectures 구조로 실험을 진행 함.
- InputCascadeCNN이 가장 좋은 성능을 보여 줌

Table 2: Performance of the cascaded architectures. The reported results are from the second phase training. The 'Rank' column shows the ranking of each method in the online score board at the time of submission.

Rank	Method	Dice			Specificity			Sensitivity		
		Complete	Core	Enhancing	Complete	Core	Enhancing	Complete	Core	Enhancing
2	InputCascadeCNN*	0.88	0.79	0.73	0.89	0.79	0.68	0.87	0.79	0.80
4-a	MFCascadeCNN*	0.86	0.77	0.73	0.92	0.80	0.71	0.81	0.76	0.76
4-c	LocalCascadeCNN*	0.88	0.76	0.72	0.91	0.76	0.70	0.84	0.80	0.75

Experimental Result

- As seen from this figure, although the segmentation is performed on Axial view but the output is consistent in Coronal and Sagittal views.

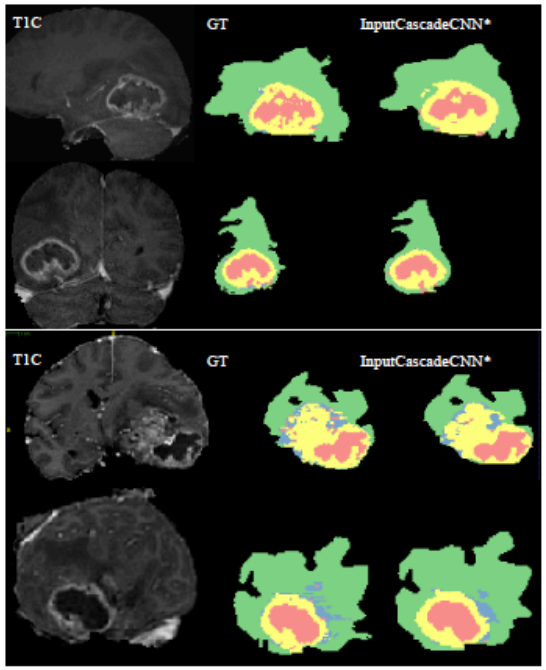


Figure 8: Visual results from our top performing model, InputCascadeCNN* on Coronal and Sagittal views. The subjects are the same as in Figure 7. In every sub-figure, the top row represents the Sagital view and the bottom row represents the Coronal view. The color codes are as follows: edema, enhanced tumor, necrosis, non-enhanced tumor.

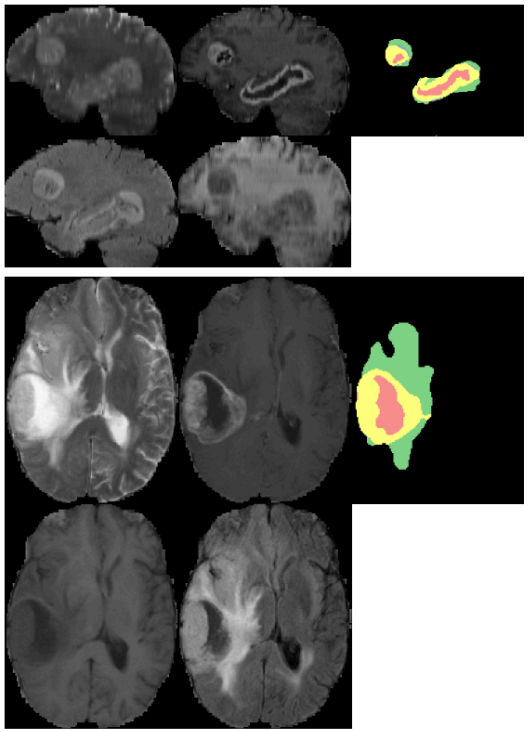


Figure 9: Visual segmentation results from our top performing model, InputCascadeCNN*, on examples of the BRATS2013 test dataset in Saggital (top) and Axial (bottom) views. The color codes are as follows: edema, enhanced tumor, necrosis, non-enhanced tumor.

Experimental Result

- Epoch별 결과

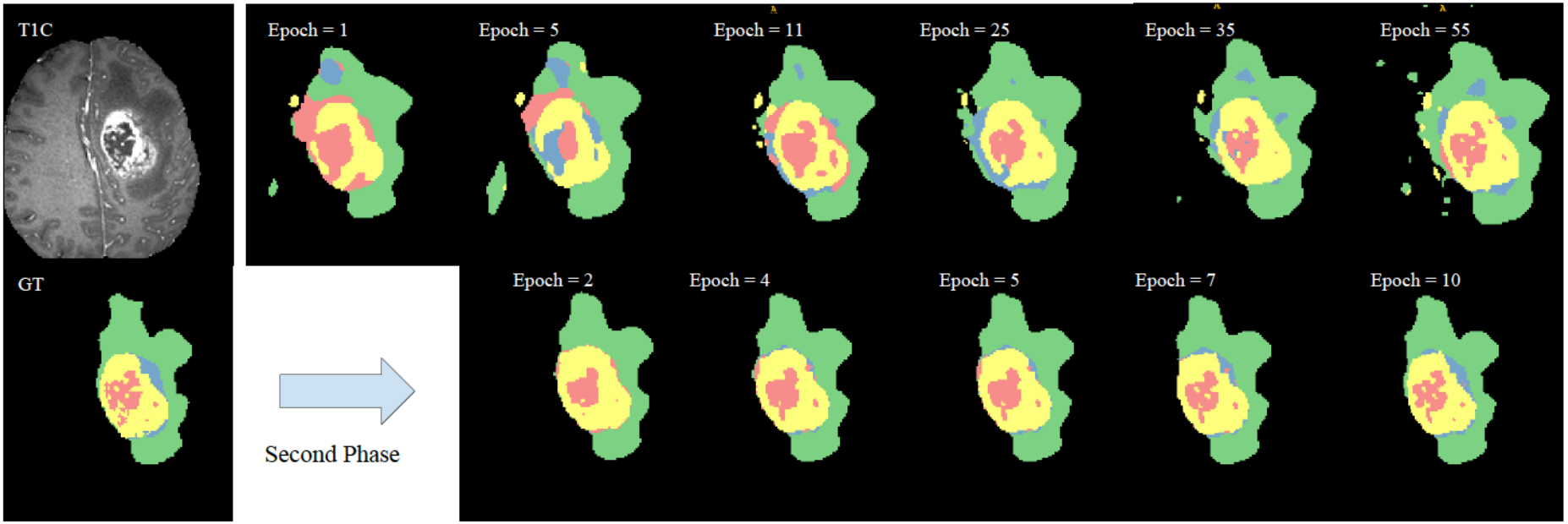


Figure 6: Progression of learning in InputCascadeCNN*. The stream of figures on the first row from left to right show the learning process during the first phase. As the model learns better features, it can better distinguish boundaries between tumor sub-classes. This is made possible due to uniform label distribution of patches during the first phase training which makes the model believe all classes are equiprobable and causes some false positives. This drawback is alleviated by training a second phase (shown in second row from left to right) on a distribution closer to the true distribution of labels. The color codes are as follows: edema, enhanced tumor, necrosis, non-enhanced tumor.

Local Path, Global Path 시각화

- As seen from this figure, features in the **local path include more edge detectors** while the ones in the **global path are more localized features**.

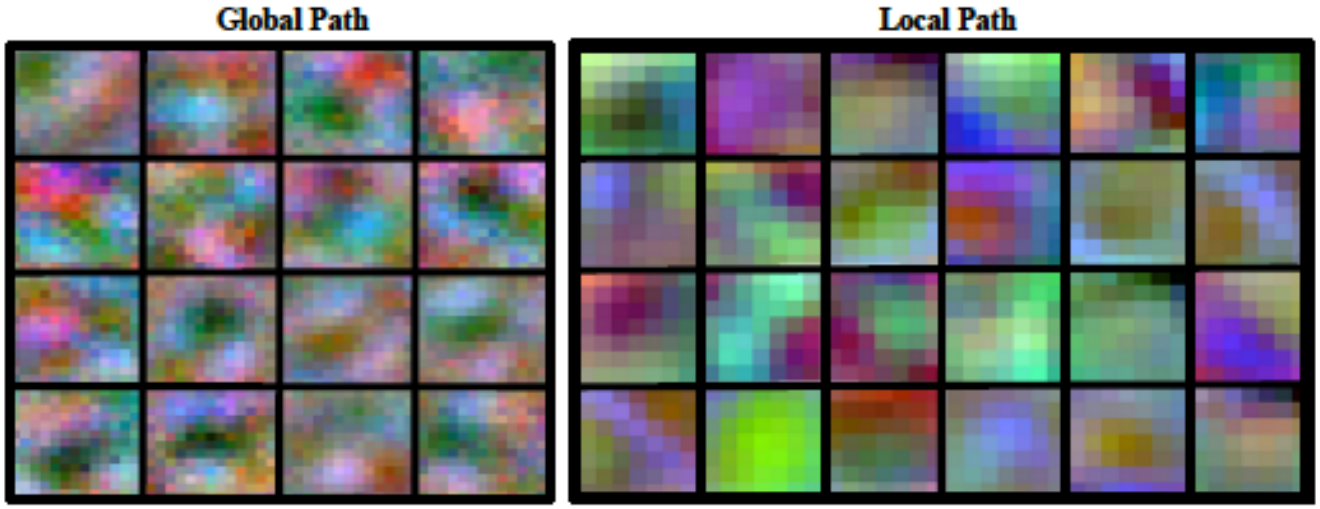


Figure 5: Randomly selected filters from the first layer of the model. From left to right the figure shows visualization of features from the first layer of the global and local path respectively. Features in the local path include more edge detectors while the global path contains more localized features.

Experimental Result

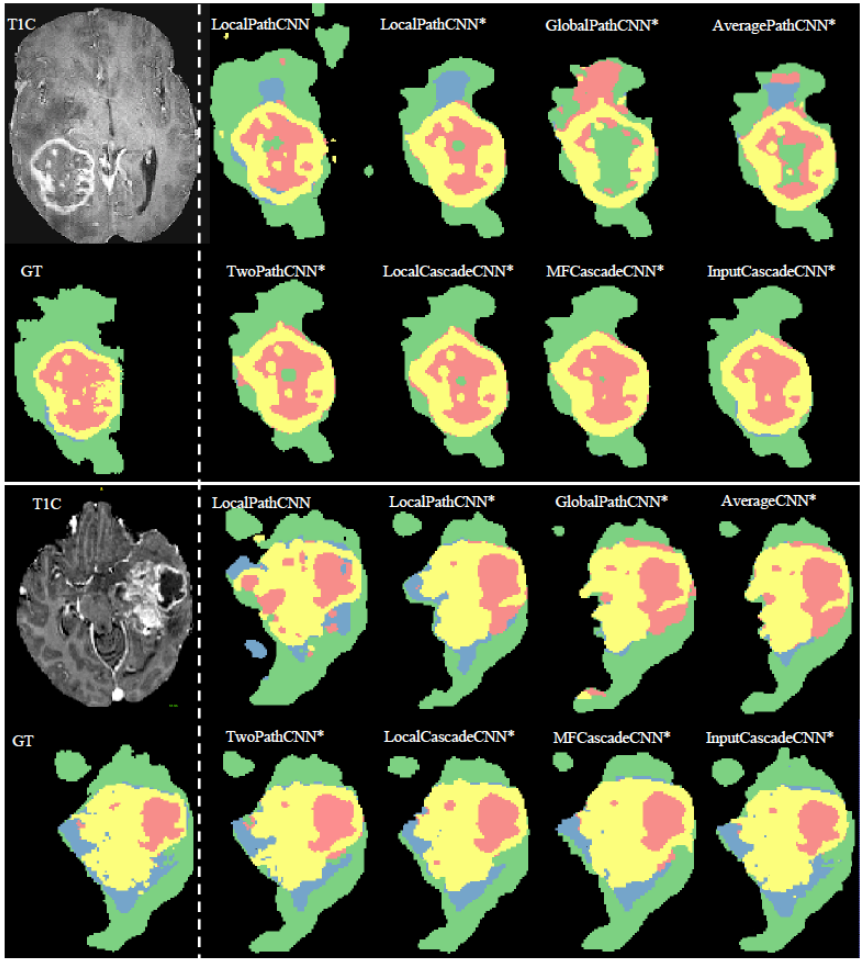


Figure 7: Visual results from our CNN architectures from the Axial view. For each sub-figure, the top row from left to right shows T1C modality, the conventional one path CNN, the Conventional CNN with two training phases, and the TwoPathCNN model. The second row from left to right shows the ground truth, LocalCascadeCNN model, the MFCascadeCNN model and the InputCascadeCNN. The color codes are as follows: edema, enhanced tumor, necrosis, non-enhanced tumor.

다른 방법들과의 비교

Table 3: Comparison of our implemented architectures with the state-of-the-art methods on the BRATS-2013 test set.

Method	Dice			Specificity			Sensitivity		
	Complete	Core	Enhancing	Complete	Core	Enhancing	Complete	Core	Enhancing
InputCascadeCNN*	0.88	0.79	0.73	0.89	0.79	0.68	0.87	0.79	0.80
Tustison	0.87	0.78	0.74	0.85	0.74	0.69	0.89	0.88	0.83
MFCascadeCNN*	0.86	0.77	0.73	0.92	0.80	0.71	0.81	0.76	0.76
TwoPathCNN*	0.85	0.78	0.73	0.93	0.80	0.72	0.80	0.76	0.75
LocalCascadeCNN*	0.88	0.76	0.72	0.91	0.76	0.70	0.84	0.80	0.75
LocalPathCNN*	0.85	0.74	0.71	0.91	0.75	0.71	0.80	0.77	0.73
Meier	0.82	0.73	0.69	0.76	0.78	0.71	0.92	0.72	0.73
Reza	0.83	0.72	0.72	0.82	0.81	0.70	0.86	0.69	0.76
Zhao	0.84	0.70	0.65	0.80	0.67	0.65	0.89	0.79	0.70
Cordier	0.84	0.68	0.65	0.88	0.63	0.68	0.81	0.82	0.66
TwoPathCNN	0.78	0.63	0.68	0.67	0.50	0.59	0.96	0.89	0.82
LocalPathCNN	0.77	0.64	0.68	0.65	0.52	0.60	0.96	0.87	0.80
Festa	0.72	0.66	0.67	0.77	0.77	0.70	0.72	0.60	0.70
Doyle	0.71	0.46	0.52	0.66	0.38	0.58	0.87	0.70	0.55

다른 방법들과의 비교

Table 4: Comparison of our top implemented architectures with the state-of-the-art methods on the BRATS-2013 leaderboard set.

Method	Dice			Specificity			Sensitivity		
	Complete	Core	Enhancing	Complete	Core	Enhancing	Complete	Core	Enhancing
InputCascadeCNN*	0.84	0.71	0.57	0.88	0.79	0.54	0.84	0.72	0.68
Tustison	0.79	0.65	0.53	0.83	0.70	0.51	0.81	0.73	0.66
Zhao	0.79	0.59	0.47	0.77	0.55	0.50	0.85	0.77	0.53
Meier	0.72	0.60	0.53	0.65	0.62	0.48	0.88	0.69	0.6
Reza	0.73	0.56	0.51	0.68	0.64	0.48	0.79	0.57	0.63
Cordier	0.75	0.61	0.46	0.79	0.61	0.43	0.78	0.72	0.52

Table 5: Comparison of our top implemented architectures with the state-of-the-art methods on the BRATS-2012 "4 label" test set as discussed in [32].

Method	Dice		
	Complete	Core	Enhancing
InputCascadeCNN*	0.81	0.72	0.58
Subbanna	0.75	0.70	0.59
Zhao	0.82	0.66	0.42
Tustison	0.75	0.55	0.52
Festa	0.62	0.50	0.61

Fully Convolutional Networks for Semantic Segmentation(FCN)을 이용한 Brain Tumor Segmentation 구현

- https://github.com/solaris33/dl_cv_tensorflow_10weeks/tree/master/week9/FCN-BRATS

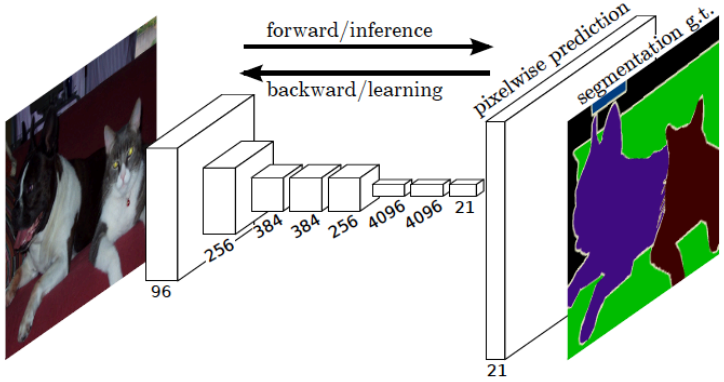
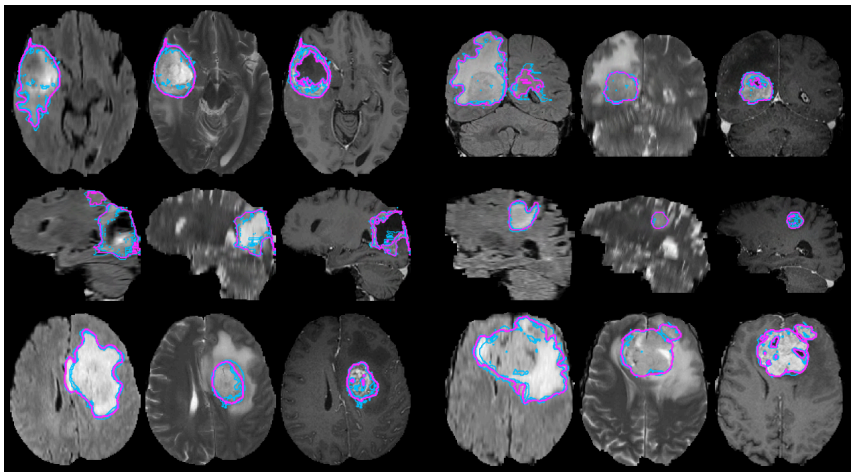


Figure 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.



실험을 위한 BraTS 데이터 세팅

- 데이터 : BraTS 2013 Training Data 중에서 HGG dataset의 뇌의 상단 이미지(T1c)만을 사용한다.
- Ground Truth : 아래와 같은 5개의 Label로 구성되어 있다.

1 for necrosis
2 for edema
3 for non-enhancing tumor
4 for enhancing tumor
0 for everything else

실험을 위한 데이터 구성 & 모델 아키텍쳐

- 데이터 구성: 60%를 Training Data, 30%를 Validation Data, 10%를 Test Data로 구성한다.

- 모델 아키텍쳐 : Fully Convolutional Networks for Semantic Segmentation(FCN) 을 이용한다.

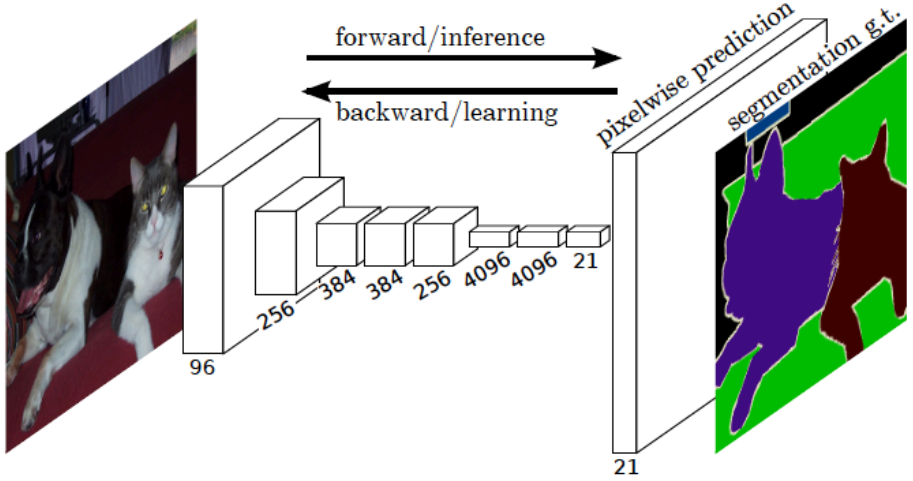


Figure 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

과제 – Custom Dataset에 대한 YOLO 모델 구현

- https://github.com/solaris33/dl_cv_tensorflow_10weeks/tree/master/week10/tensorflow-yolo 코드베이스를 토대로 Custom 데이터셋에 YOLO 모델을 구현해봅시다.

Questions & Answers

Thank You!