Efficient rgb-d semantic segmentation for indoor scene analysis

ICRA, 2021

Daniel Seichter, Mona Kohler, Benjamin Lewandowski, Tim Wengefeld and Horst-Michael Gross

2021 IEEE International Conference on Robotics and Automation (ICRA)

- **Problem/Objective**
  - Desire to excel in scene analysis on mobile robots.

- **Contribution/Key Idea**
  - Propose ESANet* for RGB-D semantic segmentation.

*ESANet = Efficient Scene Analysis Network.

Efficient rgb-d semantic segmentation for indoor scene analysis

- **Background.**

- Robots need to perform several tasks in parallel.

(Person perception, free space detection, mapping, navigation.)

- Too many constraints exist.

(Real time performance, limited computing and battery capabilities.)

- Efficient and shared initial processing step can facilitate tasks.

- Semantic segmentation is well suited for such as initial step.

- Propose ESANet* for RGB-D semantic segmentation.

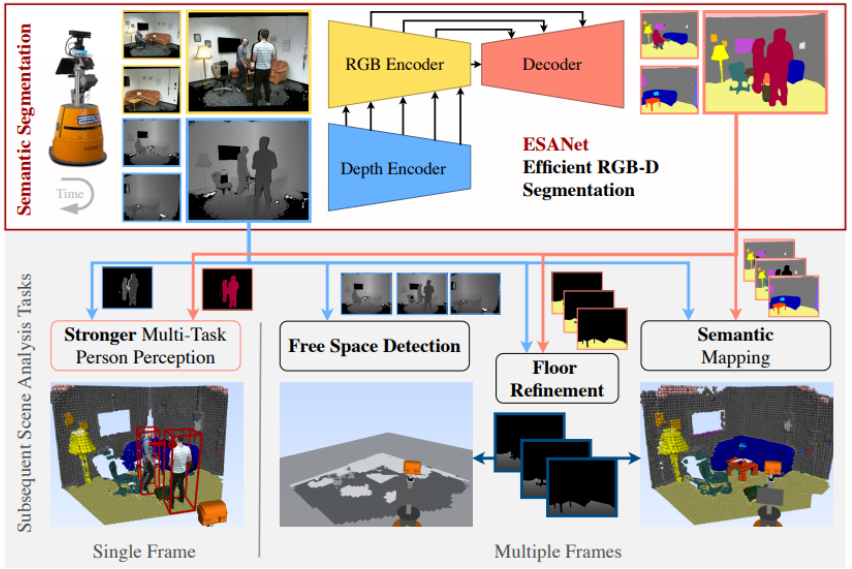
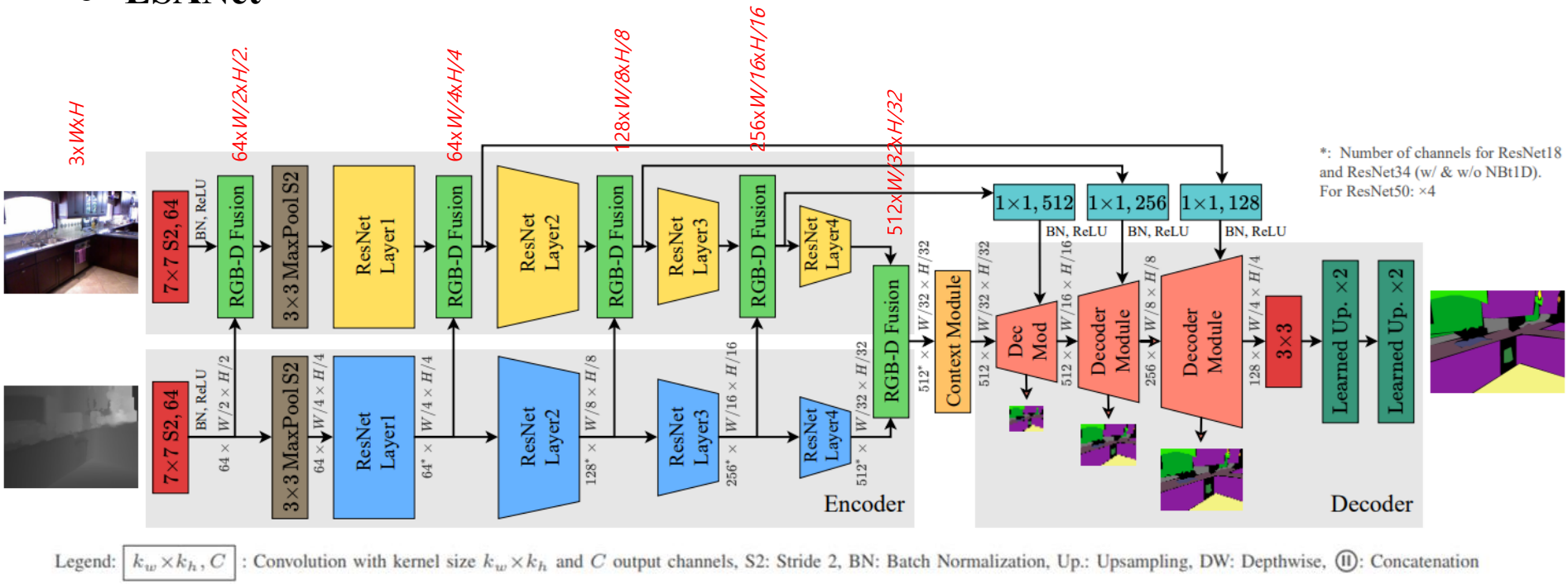


Fig. 1: Our proposed efficient RGB-D segmentation approach can serve as a common preprocessing step for subsequent tasks such as person perception, free space detection to avoid even small obstacles below the laser, or semantic mapping.

Efficient rgb-d semantic segmentation for indoor scene analysis

- **ESANet**

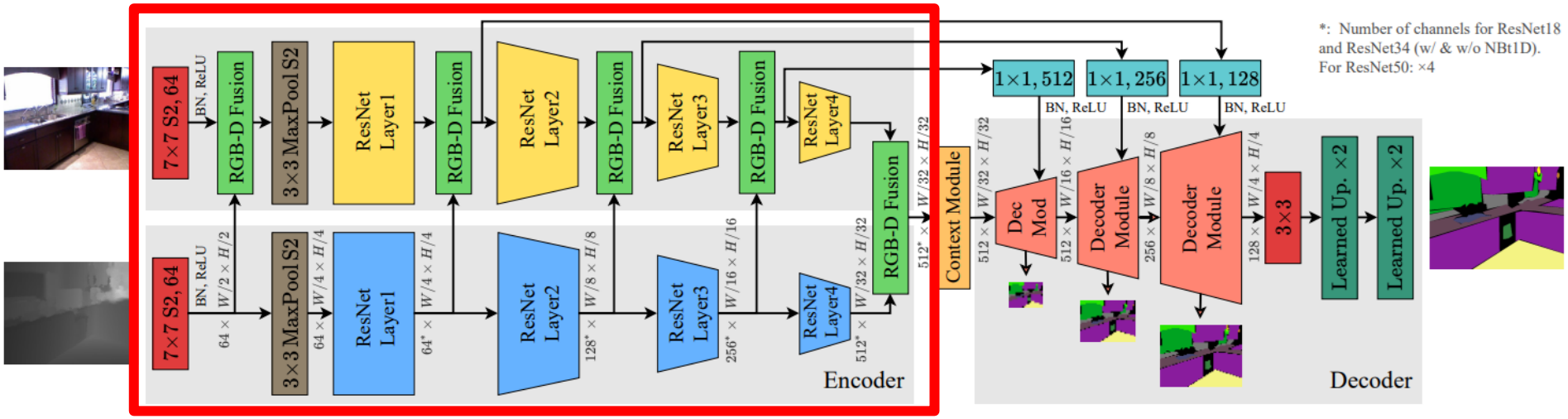


- Encoder
- Context module
- Decoder

*ESANet = Efficient Scene Analysis Network.

Efficient rgb-d semantic segmentation for indoor scene analysis

- **Encoder**



1. RGB-D segmentation.
2. RGB-D fusion.
3. NBt1D*.

*NBt1D = Non-Bottleneck-1D-Block.

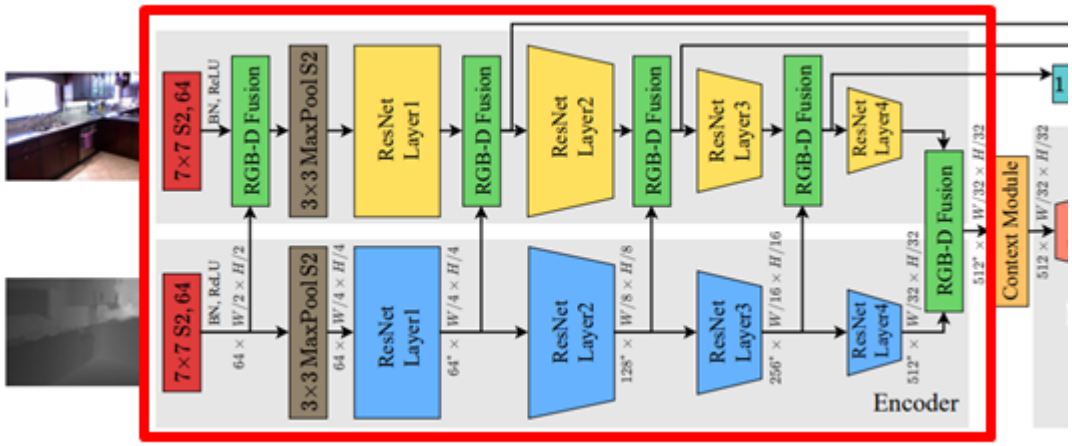
Efficient rgb-d semantic segmentation for indoor scene analysis

1. RGB-D segmentation.

- Inspired by SwiftNet's RGB segmentation.
- SwiftNet : RGB segmentation

↓ Add Depth encoder.

- ESANet : RGB-D segmentation.
- Depth encoder extracts complementary geometric info that is <u>fused in RGB encoder at five stages using an channel attention mechanism.</u>



2. RGB-D fusion.

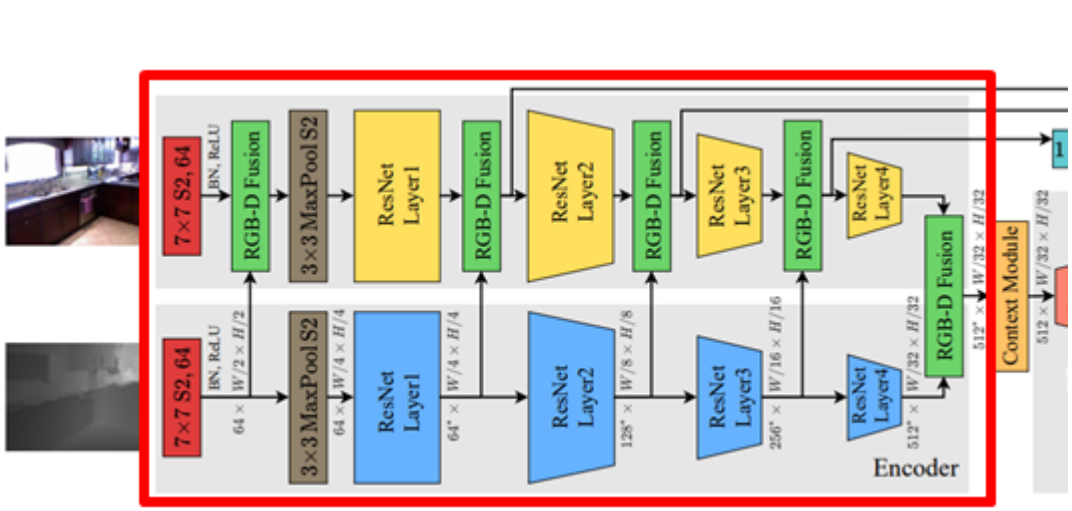
Depth features are fused into the RGB encoder using ‘channel attention mechanism’.

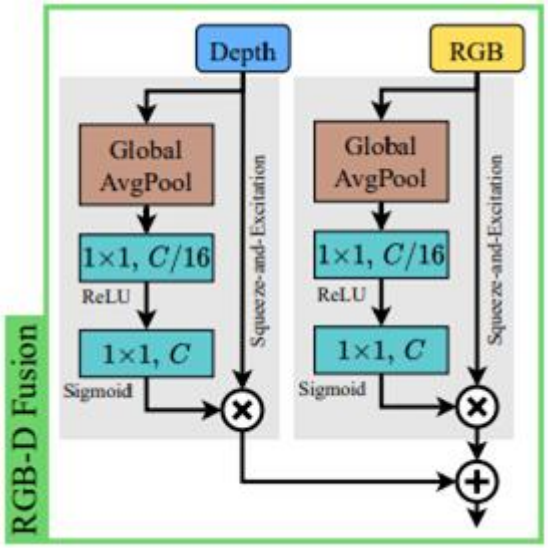
- ‘Channel attention mechanism’

Features from both modalities are reweighted with SE* module
and then summed element-wisely.

- Advantage of using ‘Channel attention mechanism’.

Model can learn which features of which modality to focus on and which to suppress.





*SE module = Squeeze and Excitation module.

Efficient rgb-d semantic segmentation for indoor scene analysis

- SE module.

Stack of SE blocks.

- SE block

Recalibrates channel-wise feature responses in two steps, *squeeze* and *excitation* by modelling channel interdependencies.

Squeeze : produce a channel descriptor by aggregating the features maps across HxW dimensions.

Excitation : sample-specific activations govern the excitation of each channel.

- Advantage of SE module.

Be able to increase sensitivity to informative features and suppress less useful ones.

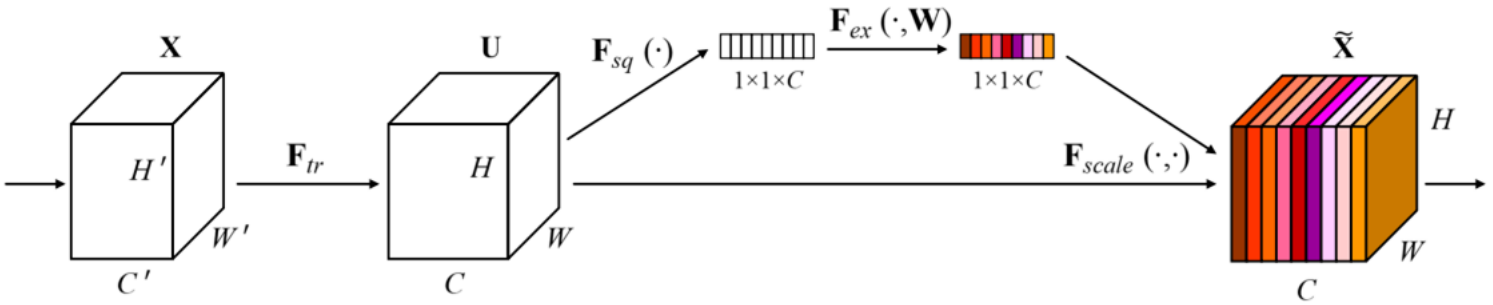
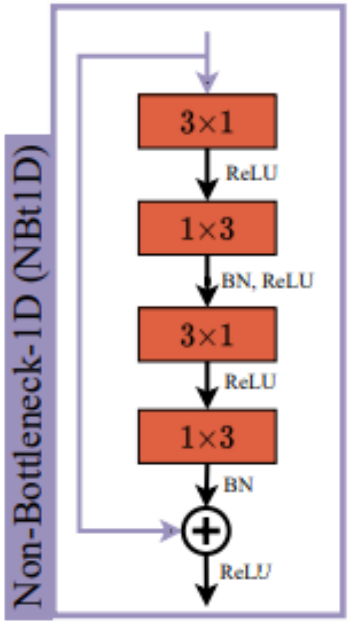


Figure 1: A Squeeze-and-Excitation block.

*SE module = Squeeze and Excitation module.

3. NBt1D*.

Each 3x3 convolution is replaced by a 3x1 and a 1x3 convolution with a ReLU in-between.



*NBt1D = Non-Bottleneck-1D-Block.

- NBt1D : initially proposed in ERFNet for real-time & accurate semantic segmentation.

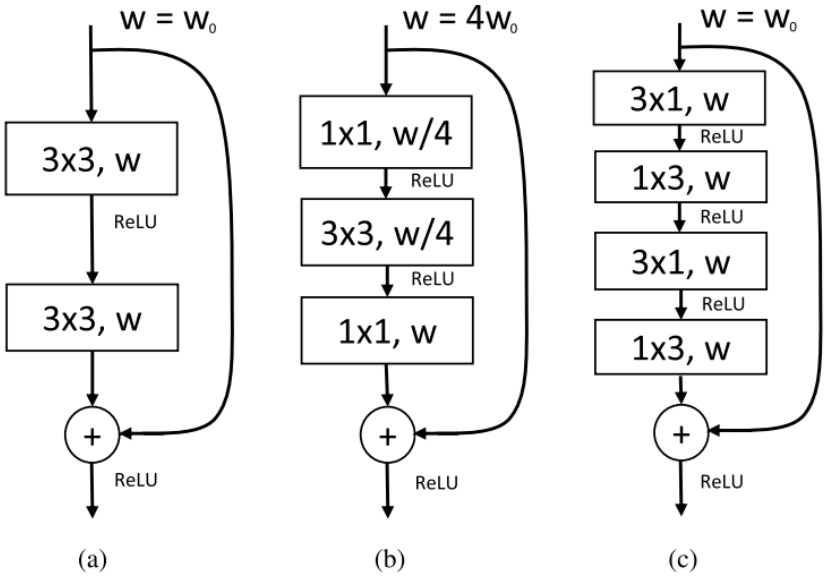


Fig. 2. Depiction of the two residual layers originally proposed in [6] (Non-bottleneck and Bottleneck), and our proposed design (Non-bottleneck-1D). w represents the number of feature maps input to the layer, internally reduced by 4 in the bottleneck design. In the convolutional blocks, "$d_1 \times d_2$, f" indicate their kernel sizes (d_1, d_2) and number of output feature maps (f). (a) Non-bottleneck. (b) Bottleneck. (c) Non-bottleneck-1D.

(a) Non-bottleneck.
Advantage : High accuracy.
Disadvantage : High computational cost.

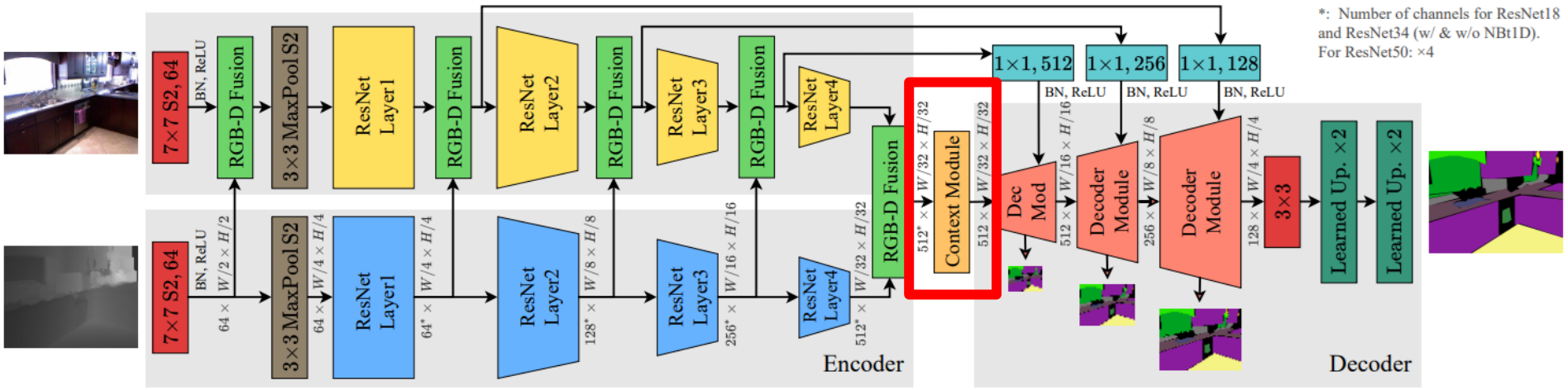
(a) Bottleneck.
Advantage : Low computational cost.
Disadvantage : Low accuracy.

(a) NBt1D
High accuracy + Low computational cost.

*NBt1D = Non-Bottleneck-1D-Block.

Efficient rgb-d semantic segmentation for indoor scene analysis

- **Context Module**



1. Use several branches.
2. Pooling sizes are always a factor of the input resolution of the context module.

Efficient rgb-d semantic segmentation for indoor scene analysis

1. Use several branches.

- Objective.

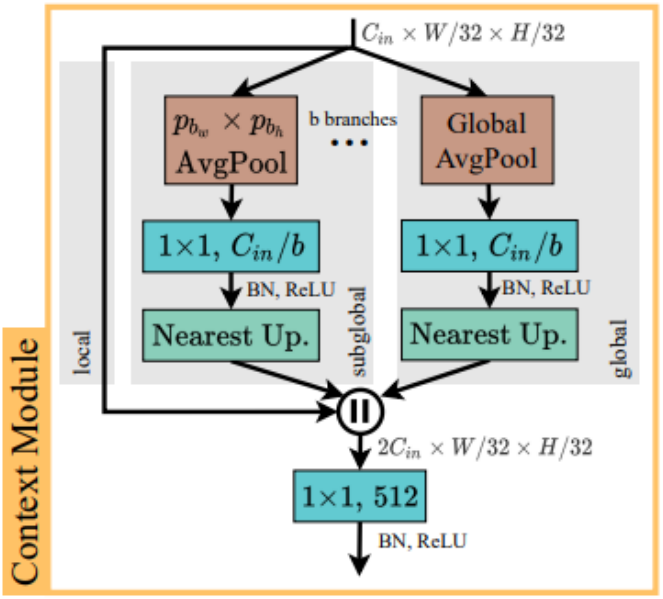
Incorporate context information by aggregating features at different scales.

- Problem.

Limited receptive field of ResNet.

- Solution.

Using several branches in a context module similar to the Pyramid Pooling Module in PSPNet*.



*PSPNet = Pyramid Scene Parsing network.

Efficient rgb-d semantic segmentation for indoor scene analysis

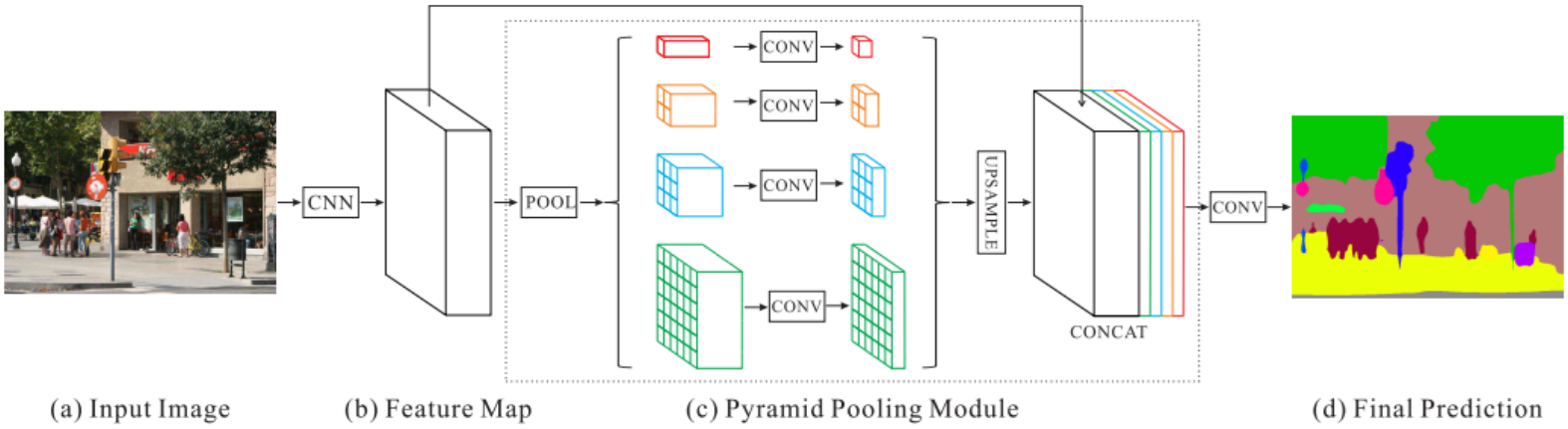


Figure 3. Overview of our proposed PSPNet. Given an input image (a), we first use CNN to get the feature map of the last convolutional layer (b), then a pyramid parsing module is applied to harvest different sub-region representations, followed by upsampling and concatenation layers to form the final feature representation, which carries both local and global context information in (c). Finally, the representation is fed into a convolution layer to get the final per-pixel prediction (d).

2. Design pooling sizes to be always a factor of the input resolution of the context module.

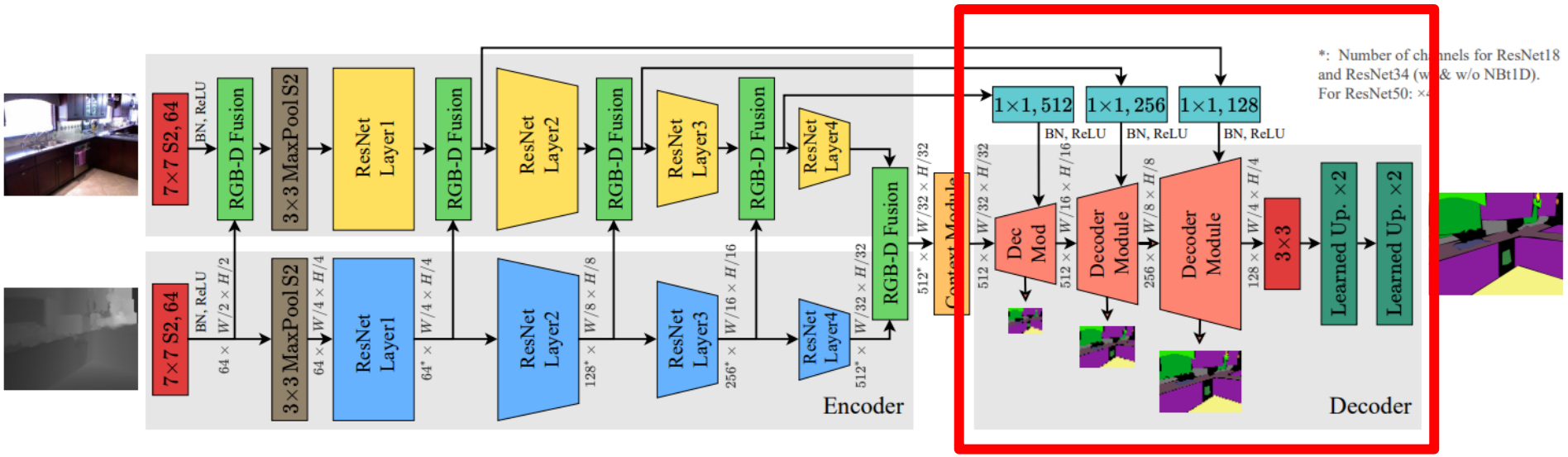
- Reason

NVIDIA TensorRT only supports pooling with fixed sizes.

*PSPNet = Pyramid Scene Parsing network.

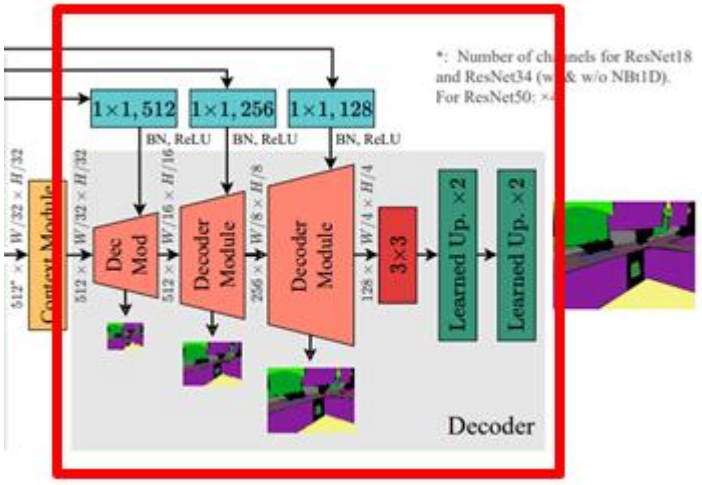
Efficient rgb-d semantic segmentation for indoor scene analysis

- **Decoder**



1. 512 channels in the first decoder module.
2. Skip connection.
3. 3 additional NBt1D.
4. Light-weight learned upsampling method.
5. Add supervision to each decoder module.

Efficient rgb-d semantic segmentation for indoor scene analysis



1. 512 channels in the first decoder module.

   - SwiftNet.

   Use a fixed number of 128 channels.

   ↓

   - ESANet.

   Use 512 channels at first decoder module and decrease the number of channels as the resolution increases.

2. Skip connection.

   - Problem.

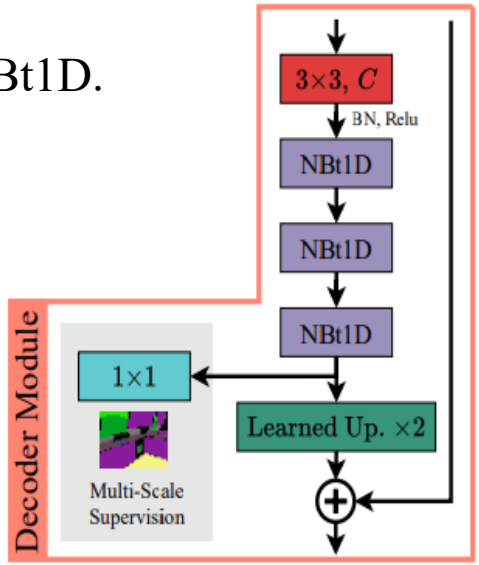
   Too many details were lost during downsampling in the encoders.

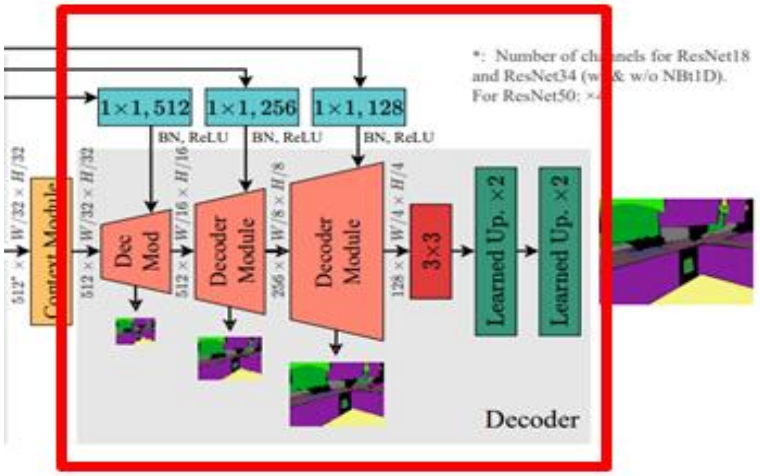
   - Solution.

   Design skip connections from encoder to decoder stages of the same resolution.

Efficient rgb-d semantic segmentation for indoor scene analysis

3. 3 additional NBt1D.





4. Light-weight learned upsampling method.

- Problem.

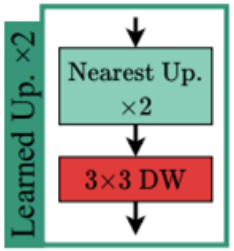
Transposed convolutions for upsampling : expensive, undesired artifacts.

- Solution.

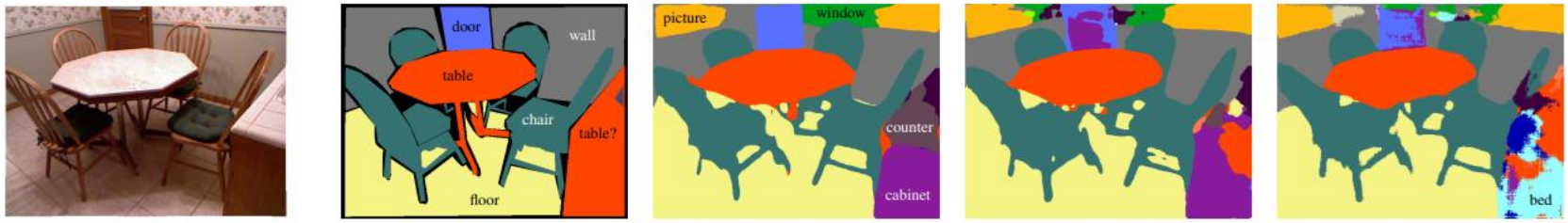
Use light-weight learned upsampling method.

- Method.

1. Use nearest neighbor upsampling to enlarge the resolution.
2. 3x3 depthwise convolution is applied to combine adjacent features.



*NBt1D = Non-Bottleneck-1D-Block.



RGB image | Ground Truth | **Learned Up. (ours)** | Bilinear Up. | Transp. Conv. (ACNet [11])

Fig. 3: Qualitative comparison of upsampling methods on NYUv2 test set (same colors as in Fig. 1 and Fig. 6).

5. Add supervision to each decoder module.

At each scale, a 1x1 convolution computes a segmentation of a smaller scale, which is supervised by the down-scaled ground truth segmentation.

- **Experiments**

Train networks using PyTorch.
500 epochs.
Batch size = 8.
Best models were chosen based on the mIoU.

- Indoor dataset.

NYUv2 : 795 training images, 654 testing images, 40 classes.
SUNRGB-D : 5285 training images, 5050 testing images, 37 classes.

- Outdoor dataset.

Cityscapes : 2975 training images, 500 validation images, 1525 testing images, 19 classes.

- **Results**

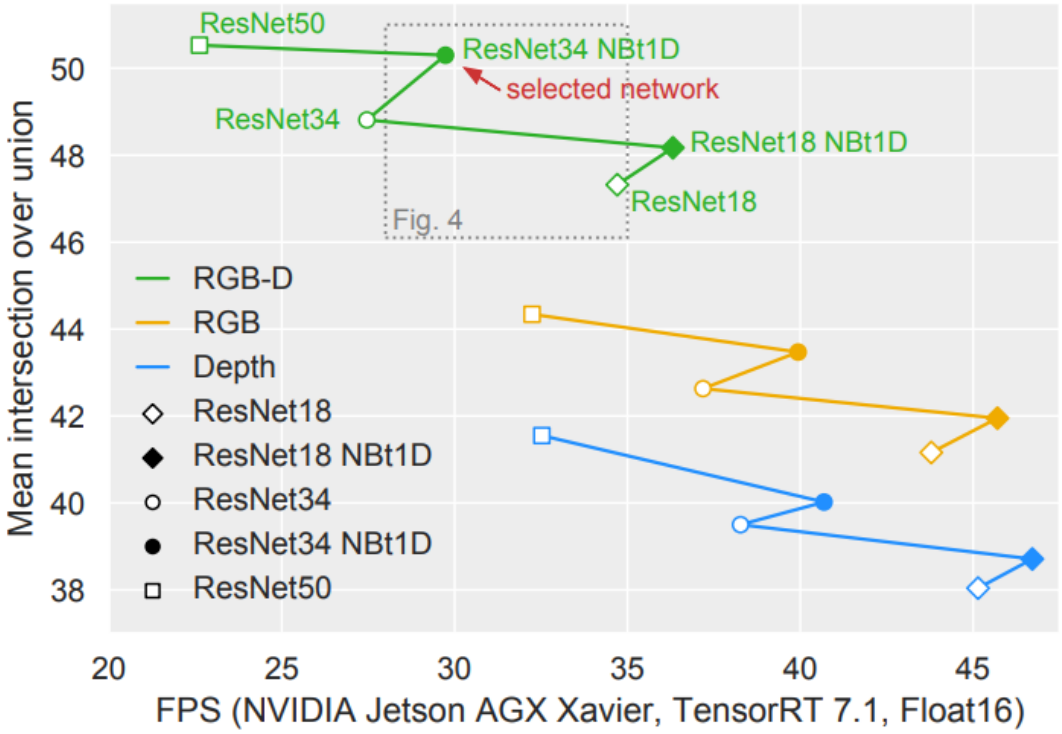


Fig. 4: Comparison of RGB-D to RGB and depth networks (single encoder) and different backbones on NYUv2 test set.

- **Results**

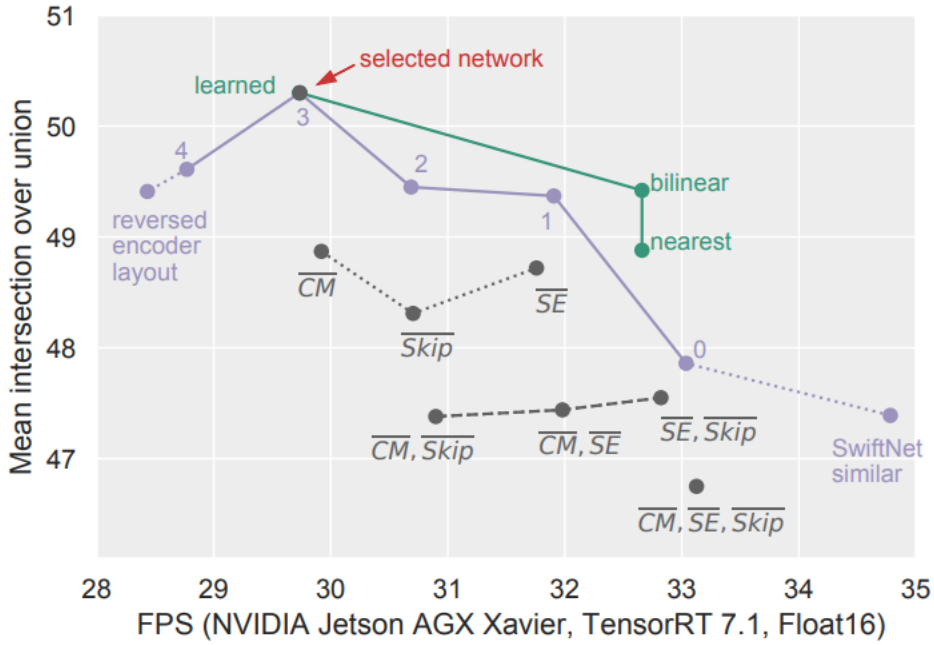
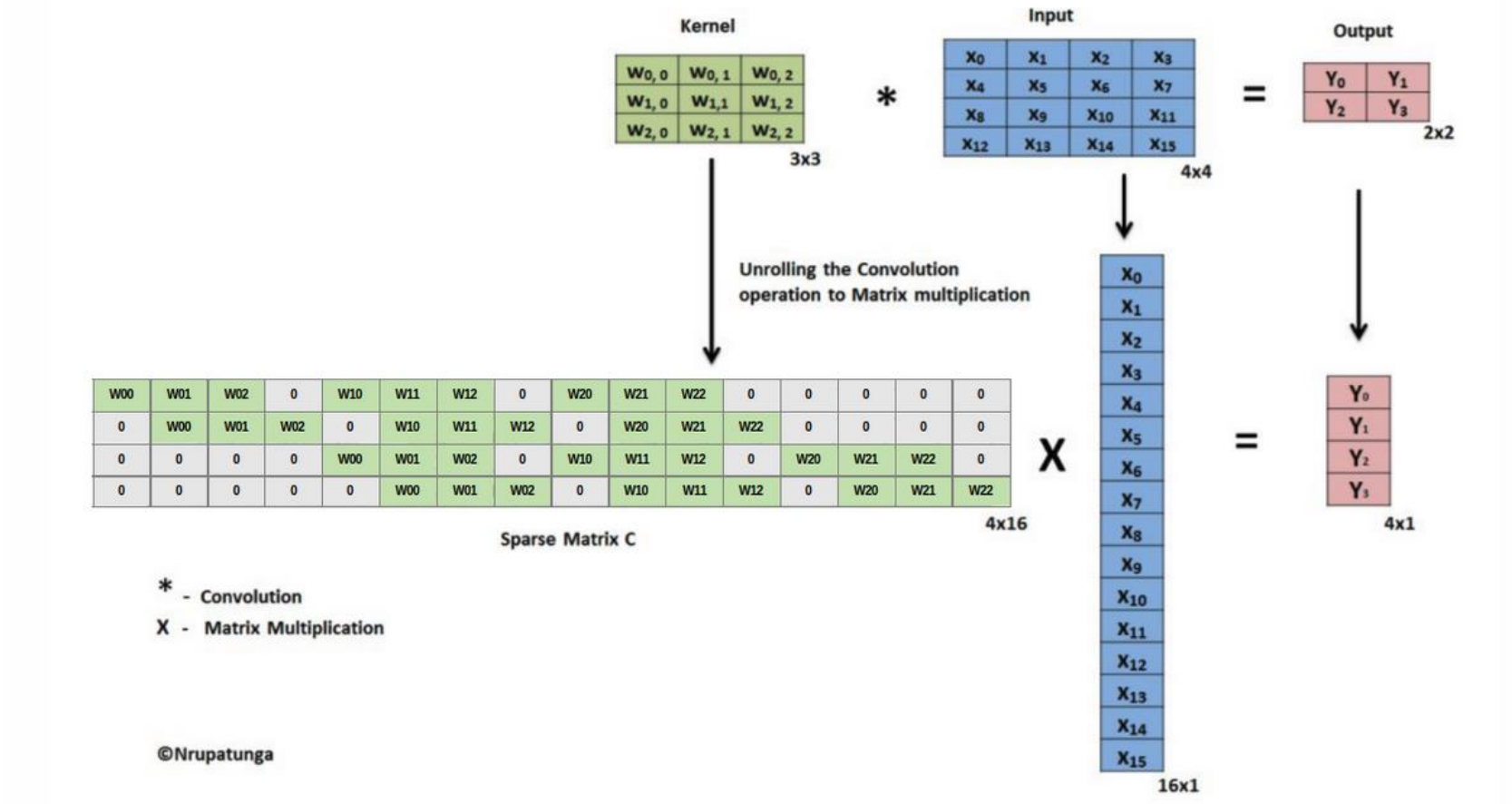


Fig. 5: Ablation Study on NYUv2 test. Each color indicates modifying one aspect: purple: number of NBt1D blocks in decoder module, dark green: upsampling method, and gray: usage of specific network parts with $\overline{CM}$: no context module, $\overline{Skip}$: no encoder-decoder skip connections, and $\overline{SE}$: no Squeeze-and-Excitation before fusing RGB and depth.

추가 공부

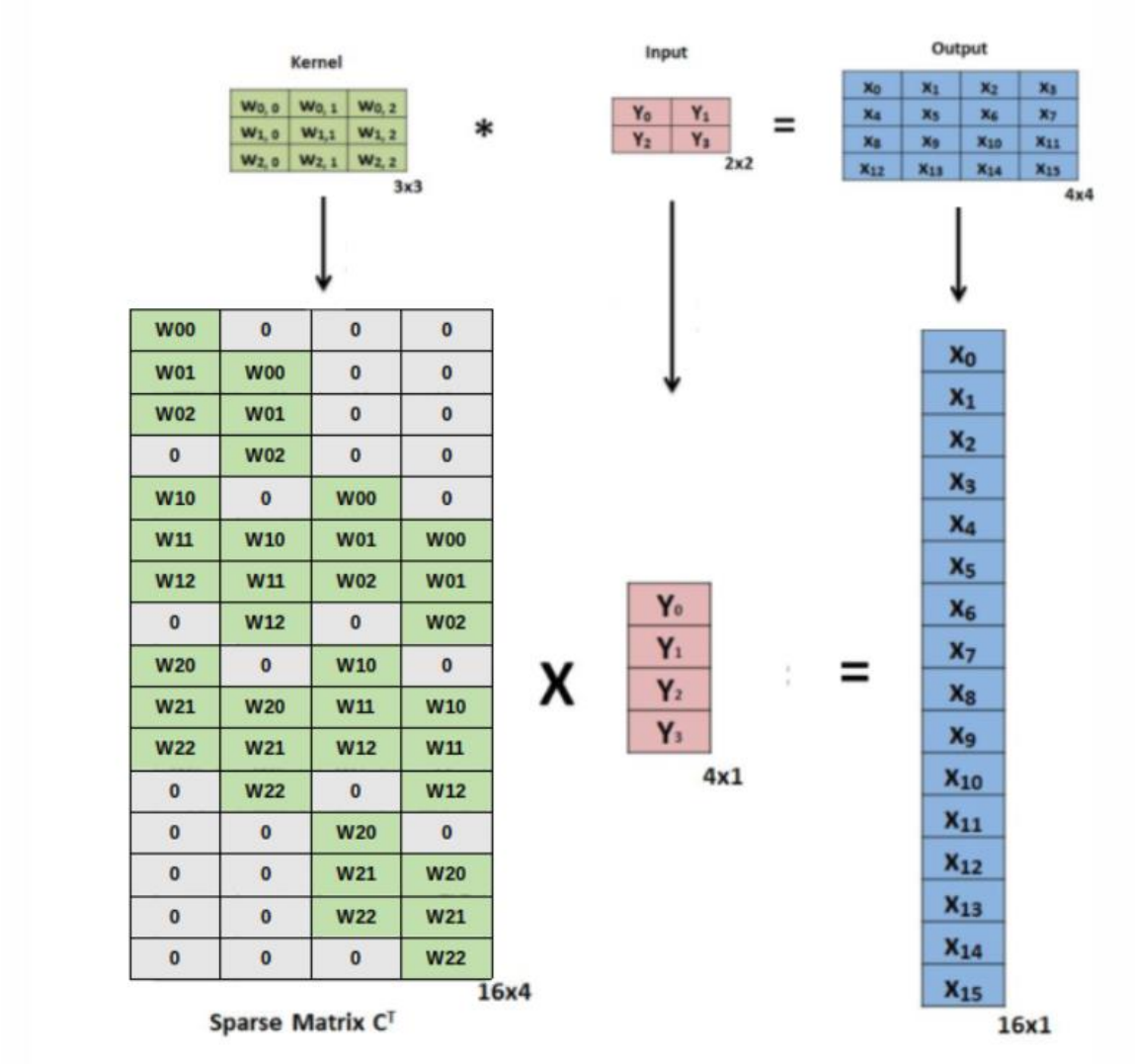
https://eehoeskrap.tistory.com/431

2D Convolution.



https://eehoeskrap.tistory.com/431

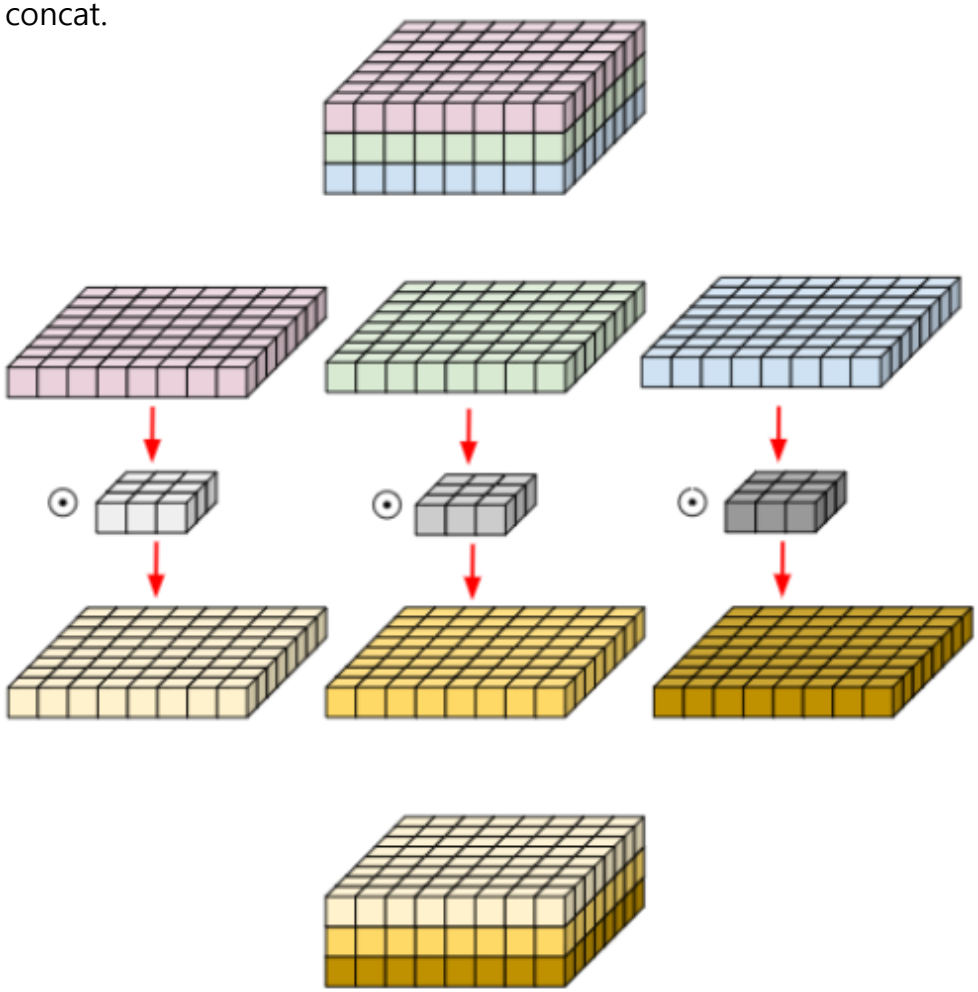
2D Transposed Convolution



https://eehoeskrap.tistory.com/431

Depthwise Convolution

channelwise convolution 후 concat.



https://openaccess.thecvf.com/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf

ResNet

RESNet 등장 배경.

깊은 망의 문제점 (1. Vanishing/Exploding Gradient 문제. 2. training 어려움.) 해결 목적.

기존.

목표 : 최적의 H(x) 찾기.

ResNet.

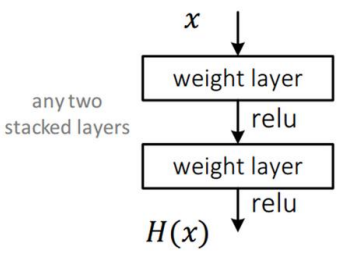
목표 : 최적의 F(x) 찾기.

F(x)=H(x)-x.

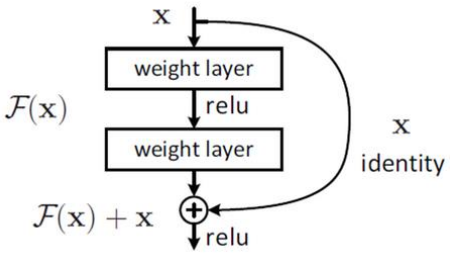
Residual learning의 기본 블락 : 입력에서 출력으로 직행 shortcut 연결 생성 : 연산량 관점에 오직 덧셈만 추가될 뿐.

최적의 경우 F(x) = 0. : 학습 방향 미리 결정되어있음. : pre-conditioning 구실 함.

F(x) = 0 되는 방향으로 학습을 하게 되면, 입력의 작은 움직임(fluctuation)을 쉽게 검출 가능. : F(x)가 작은 움직임을 학습한다. : F(x)가 나머지 (residual)를 학습한다. : residual learning.



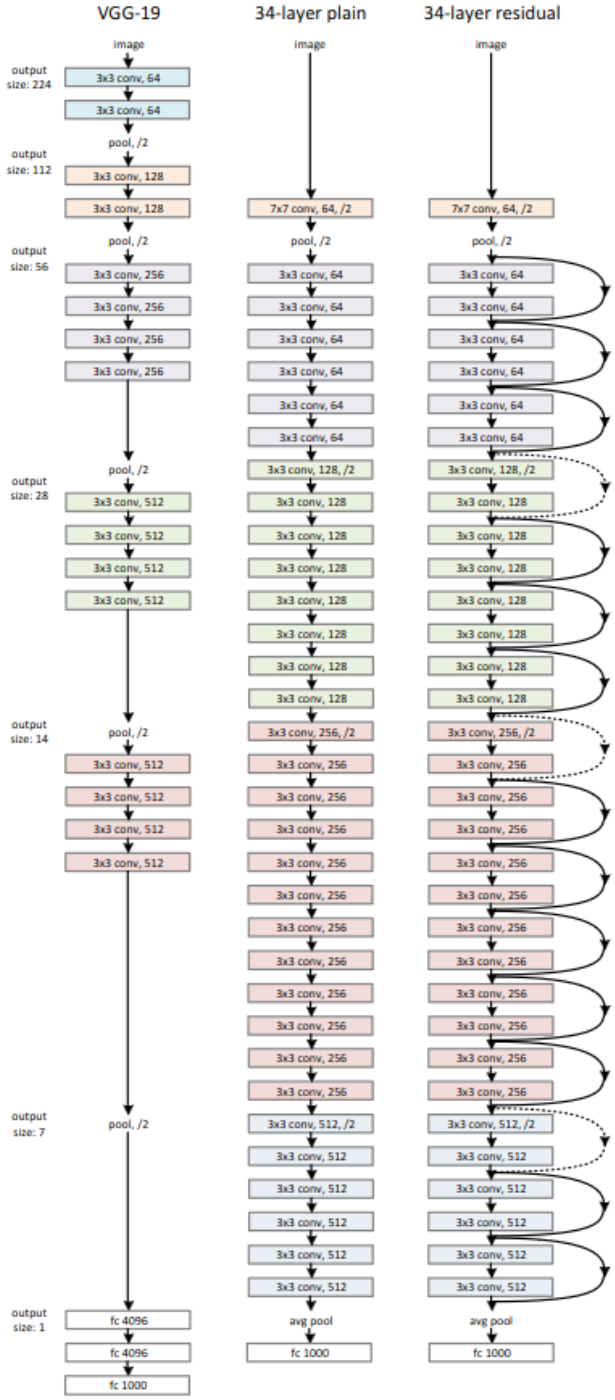
기존.



ResNet block.

https://openaccess. al_Learning_CVPR_2016_paper.pdf

RESNet-34.

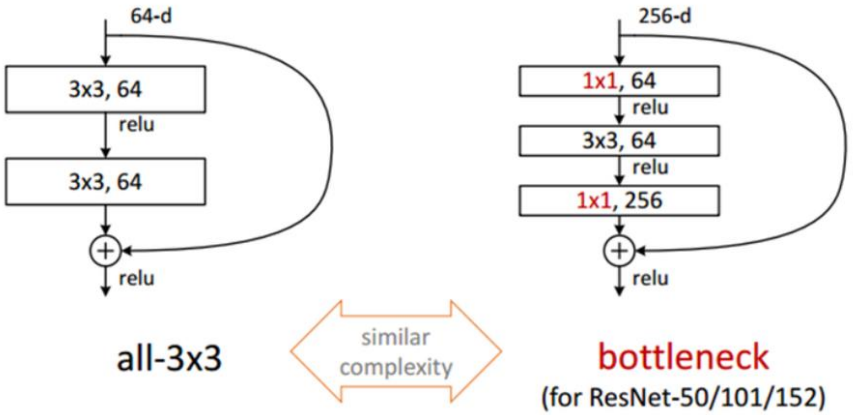


layer name	output size	18-layer	34-layer
conv1	112×112		
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$
	1×1		aver
FLOPs		1.8×10^9	3.6×10^9

Table 1. Architectures for ImageNet. Building blocks are shown in b
sampling is performed by conv3_1, conv4_1, and conv5_1 with a strid

https://openaccess.thecvf.com/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf

Bottleneck building block for ResNet-50/101/152.



Bottleneck 구조 명칭 이유 : 차원 감소 후 차원 증가 모습이 병목처럼 보여서.

처음 1x1 conv 효과 : dimension 감소.

마지막 1x1 conv 효과 : dimension 증가.

Bottleneck 구조 사용 이유 : 연산량 절감하여 연산시간 감소.

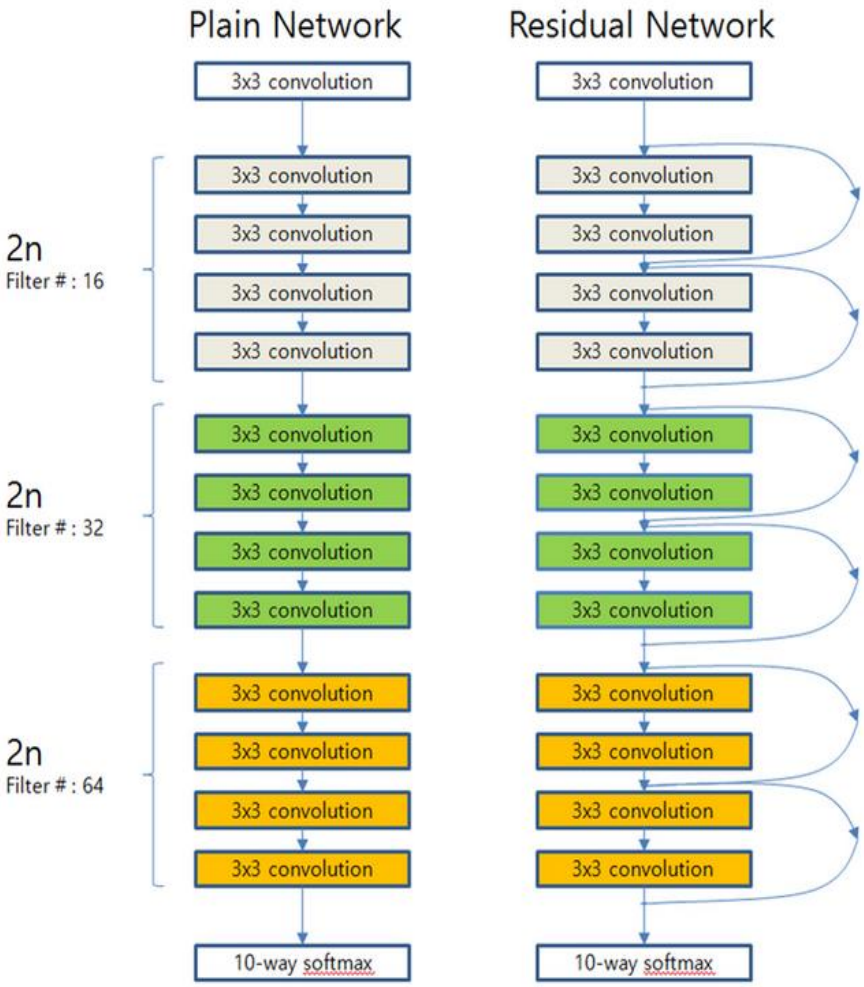
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Table 1. Architectures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Down-sampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

https://openaccess.thecvf.com/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf

CIFAR-10 dataset에 대한 실험.

CIFAR-10 dataset : 32x32 images. training image 5만개, test image 1만개. 총 10개 class. 즉, class당 6000장 존재함.



-맨 처음 conv layer에 3x3 conv 사용.
이유 :
ImageNet dataset 은 224x224 image로 크기 커서 맨처음 conv layer 에 7x7 conv 사용했지만, CIFAR-10 dataset 은 32x32 image로 크기가 작아서.

-6n개의 3x3 conv layer 그룹 존재.
각 2n에 대해, feature map 크기 {32,16,8}.
filter 개수 {16,32,64}.

-전체 layer 수는 '6n+2' 개.
이유 :
맨 마지막에 global average pooling 과 10-way softmax를 배치해서.