Paper Presentation: Involution

Group 4

1. Harsh Chalikwar - 2021A3PS2878G
2. Aditya Kurande - 2021AAPS2485G
3. Atharv Mane - 2020A7TS0153G
4. Simran Srivastava - 2021AAPS2931G
5. Milind Kumar Prasad - 2020A7PS0130G

Involution: Inverting the Inherence of Convolution for Visual Recognition

Duo Li[1] Jie Hu[2] Changhu Wang[2] Xiangtai Li3 Qi She[2] Lei Zhu[3] Tong Zhang[1] Qifeng Chen[1] The Hong Kong University of Science and Technology[1] ByteDance AI Lab[2] Peking University[3]

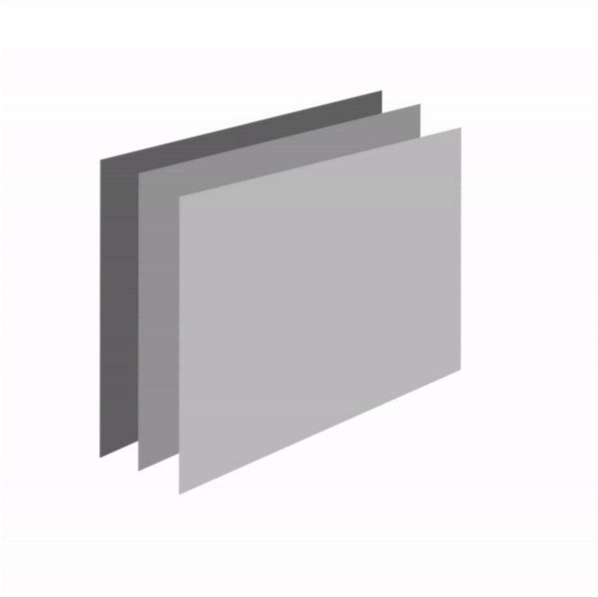
CVPR 2021

Motivation:

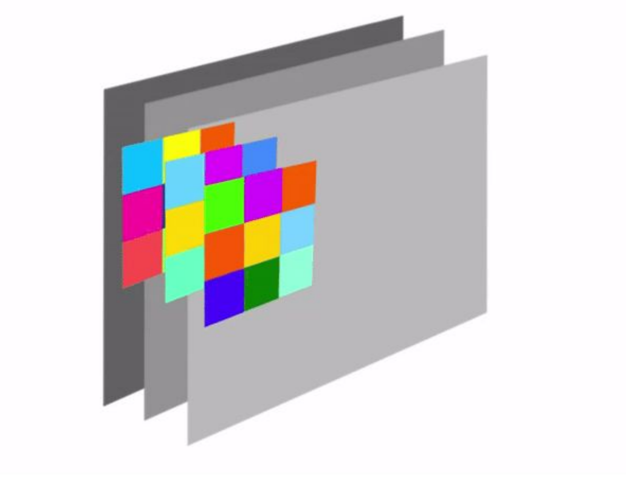
Challenging the fundamental design of convolution

Two key properties of Convolution

Channel-Specific



Spatial-Agnostic



Channel-Specific

Collect diverse information encoded in different channels

Inter-channel redundancy

Spatial-Agnostic

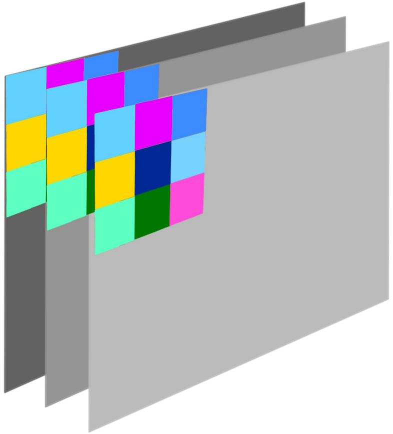
Translation equivalence

Can't adapt to diverse visual patterns with respect to different spatial positions.

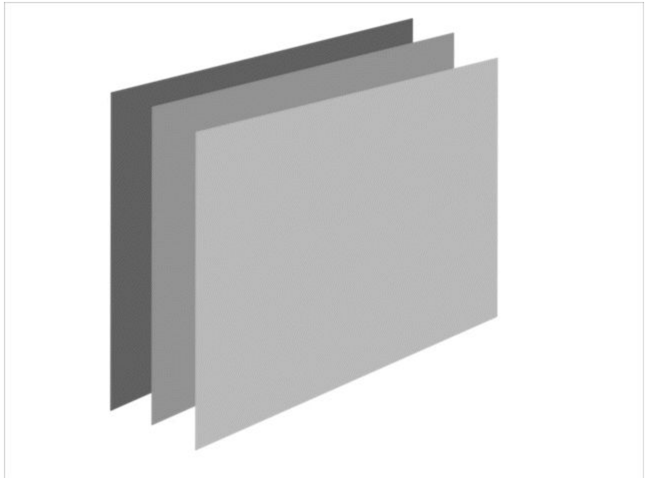
Can't capture long-range spatial interactions in a single shot

Two key properties of Involution

Channel-~~Specific~~ Agnostic



Spatial-~~Agnostic~~ Specific



Two- Fold Privilege

1. **Summarize the context** in a wider spatial arrangement:

   Overcome the difficulty of modeling long-range interactions

2. **Adaptively allocate the weights** over different positions:

   Prioritize the most informative visual elements in the spatial domain

Primary Contributions

1. **Rethink the inherent properties of convolution:** Channel Agnostic and Spatial Specific kernel over the usual channel-specific and spatial-agnostic

2. **Unify the view of self-attention and convolution** through the lens of involution: bridge the emerging philosophy of incorporating self-attention into the learning procedure of visual representation.

3. **Wide range of implementations:** Involution-powered architectures work universally well across a wide array of vision tasks, including image classification, object detection, instance and semantic segmentation, offering significantly better performance than the convolution-based counterparts.

Design of Involution

- Involution Operation

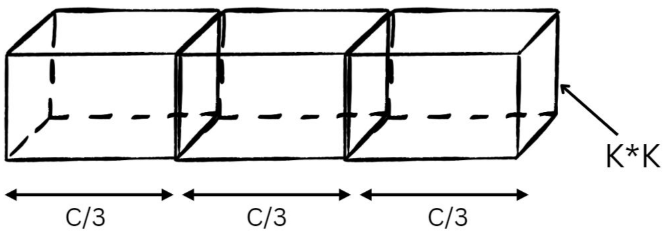
$$\mathbf{Y}_{i,j,k} = \sum_{(u,v)\in\Delta_K} \mathcal{H}_{i,j,u+\lfloor K/2\rfloor,v+\lfloor K/2\rfloor,\lceil kG/C\rceil}\mathbf{X}_{i+u,j+v,k}.$$

- kernel generation function

$$\mathcal{H}_{i,j} = \phi(\mathbf{X}_{i,j}) = \mathbf{W}_1\sigma(\mathbf{W}_0\mathbf{X}_{i,j}).$$

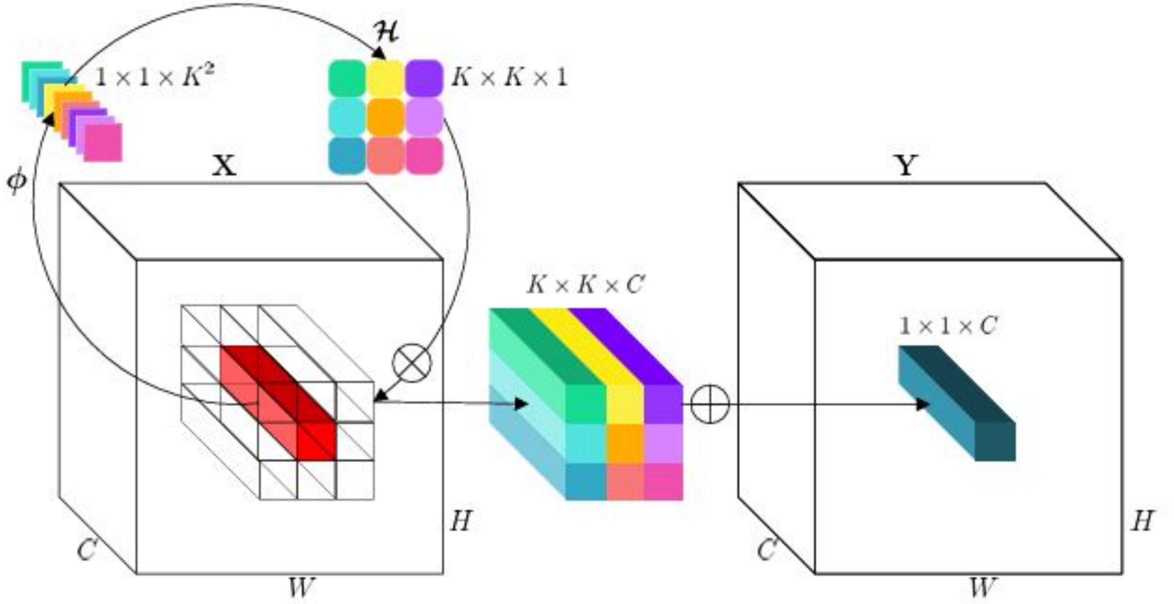
- RedNet modifies ResNet architecture, replacing 3x3 convolutions with involution in bottleneck positions for enhanced efficiency.
- Retain all the 1x1 convolution for channel projection and fusion.

Input volume with G = 3



involution kernels $\mathcal{H} \in \mathbb{R}^{H \times W \times K \times K \times G}$

$\mathbf{X} \in \mathbb{R}^{H \times W \times C_i}$



Algorithm 1 Pseudo code of involution in a PyTorch-like style.

```
# B: batch size, H: height, W: width
# C: channel number, G: group number
# K: kernel size, s: stride, r: reduction ratio

#################### initialization ####################
o = nn.AvgPool2d(s, s) if s > 1 else nn.Identity()
reduce = nn.Conv2d(C, C//r, 1)
span = nn.Conv2d(C//r, K*K*G, 1)
unfold = nn.Unfold(K, dilation, padding, s)
##################### forward pass #####################
x_unfolded = unfold(x) # B,CxKxK,HxW
x_unfolded = x_unfolded.view(B, G, C//G, K*K, H, W)
# kernel generation, Eqn.(6)
kernel = span(reduce(o(x))) # B,KxKxG,H,W
kernel = kernel.view(B, G, K*K, H, W).unsqueeze(2)
# Multiply-Add operation, Eqn.(4)
out = mul(kernel, x_unfolded).sum(dim=3) # B,G,C/G,H,W
out = out.view(B, C, H, W)
return out
```

Involution Design Justification

How and Why?

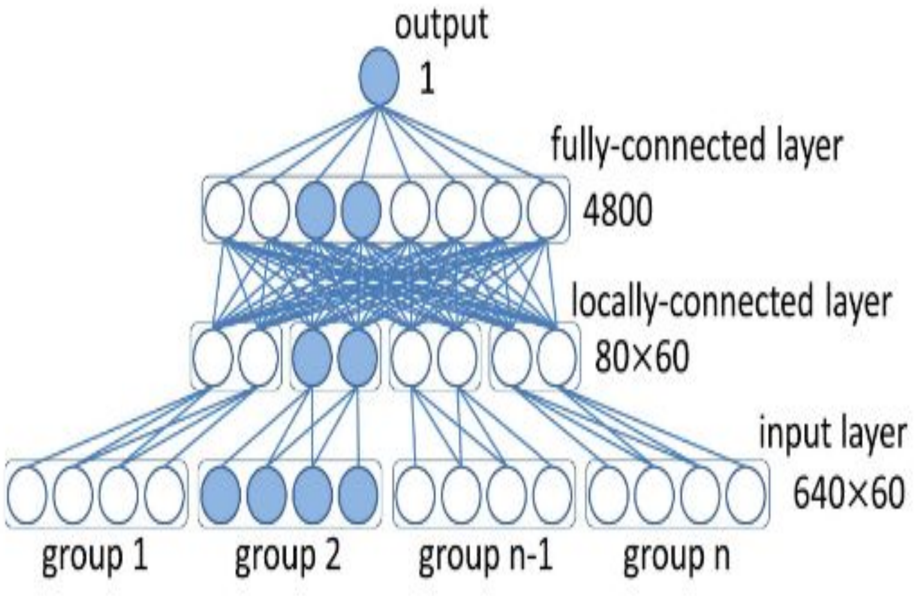
Convolution vs. Involution - Spatial Agnosticism vs. Specificity

- Different parts of image may require different weights spatially.

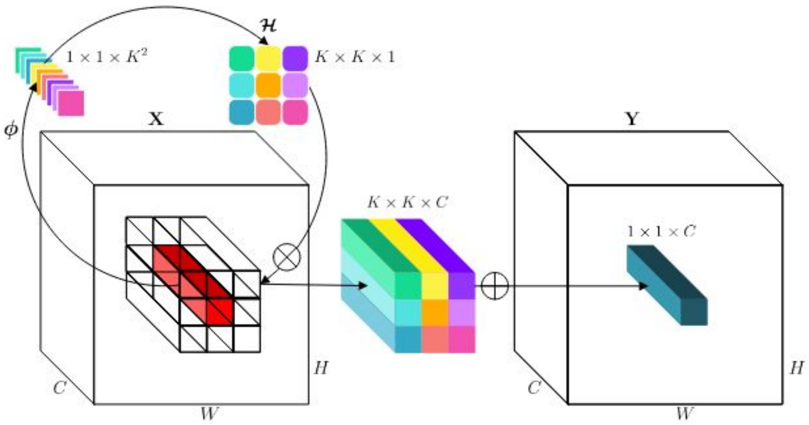
- E.g. Facial Recognition (e.g. DeepID, DeepFace)
  - Failed because of hardbound spatial non-interaction.

- Involutions share the “generation function” spatially.

Hard vs. Soft (Dynamic) Spatial Diversity - DeepID vs. Involution



Deep ID



Involution

Why Dynamically Prepared Kernels?

- Significant amount of papers have shown that Dynamically prepared kernels perform better than static Kernels.
- Like having specially tailored CNNs for each data-point.
- Neural Networks like “Hypernetworks” directly produce convolution kernels. Thus they retain both properties of convolution that we have discussed.

Why shared channels?

- Channel-wise redundancy!! Save Parameters!!
- What does this mean?
  - The features extracted by CNNs (especially at shallower layers) are channel-wise redundant. i.e they pick up similar details.

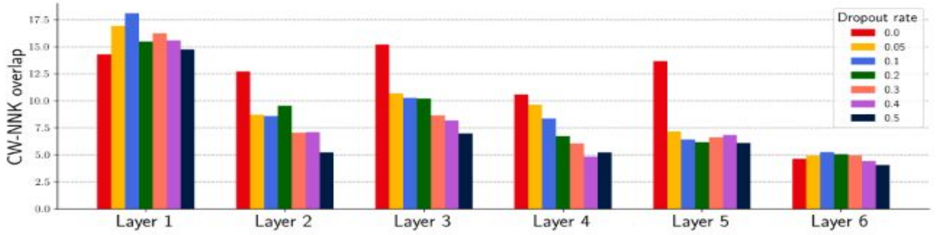


Fig. 1: Number of pairwise channel NNK intersections to average number of NNK neighbors per channel ratio (CW-NNK overlap) in each layer for different dropout rates.

Bonet, David, et al. "Channel redundancy and overlap in convolutional neural networks with channel-wise nnk graphs." Explains this in greater detail.

Involution and Self-Attention

No Wonder it works

Involution and Self Attention

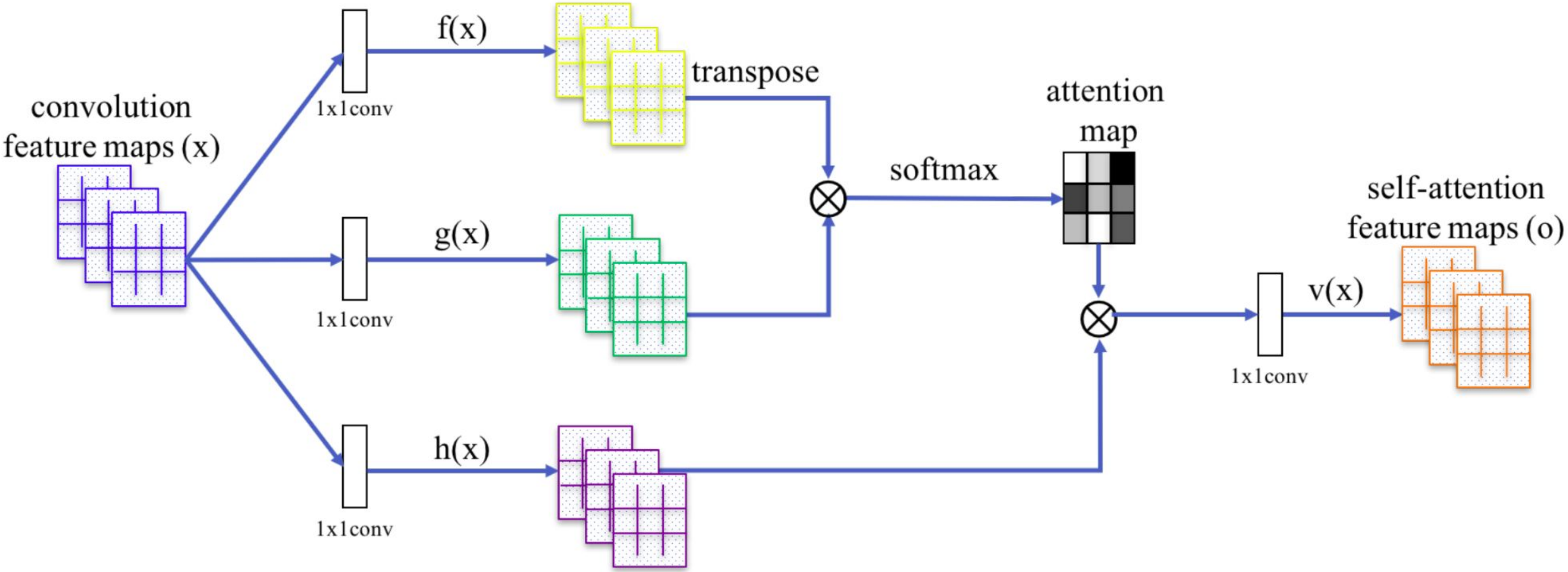
$$Y_{i,j,k} = \sum_{p,q \in \omega} Func(QK^T)_{i,j,p,q,\lceil \frac{kH}{C} \rceil} V_{p,q,k}$$

Self-Attention Mechanism

$$Y_{i,j,k} = \sum_{i,j \in \Delta} H_{i,j,u+\lfloor \frac{K}{2} \rfloor, v+\lfloor \frac{K}{2} \rfloor} X_{i+u,j+v,k}$$

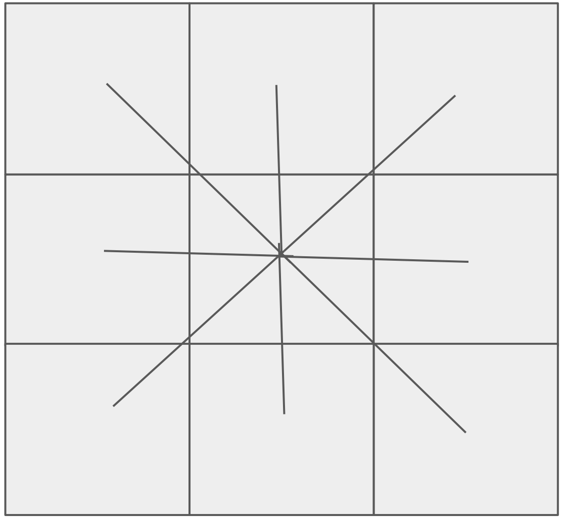
Involution Mechanism

Involution and Self Attention

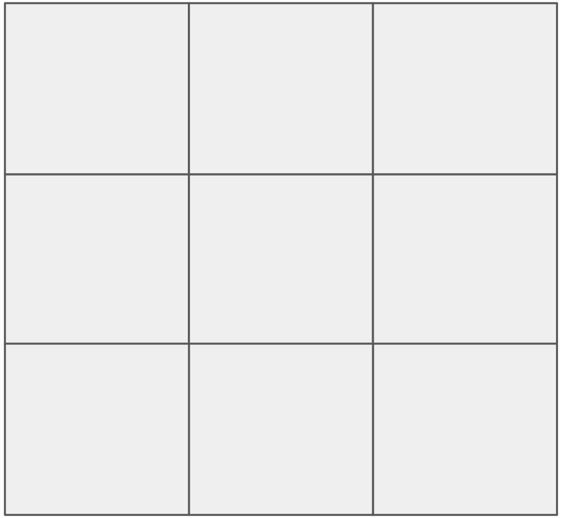


Involution and Self Attention - Difference?

Attention



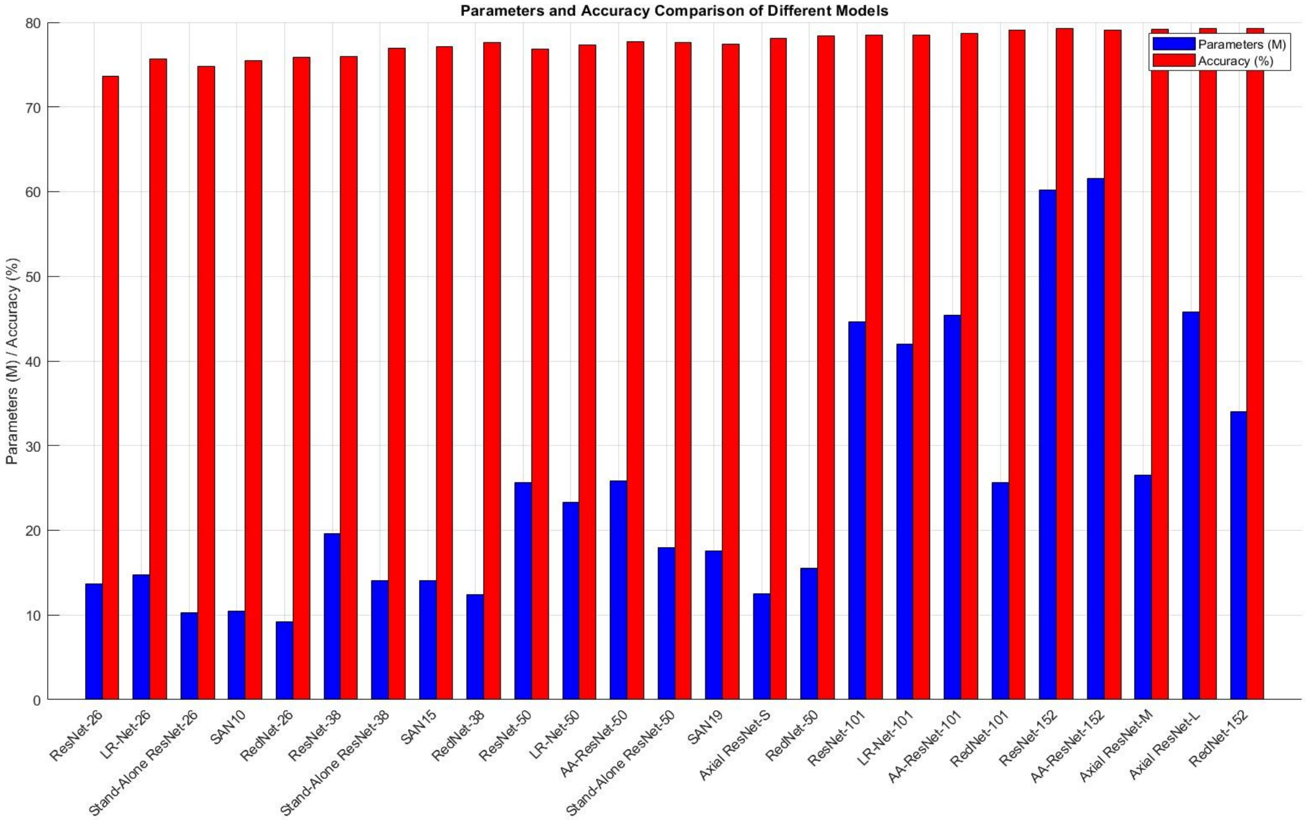
Involution

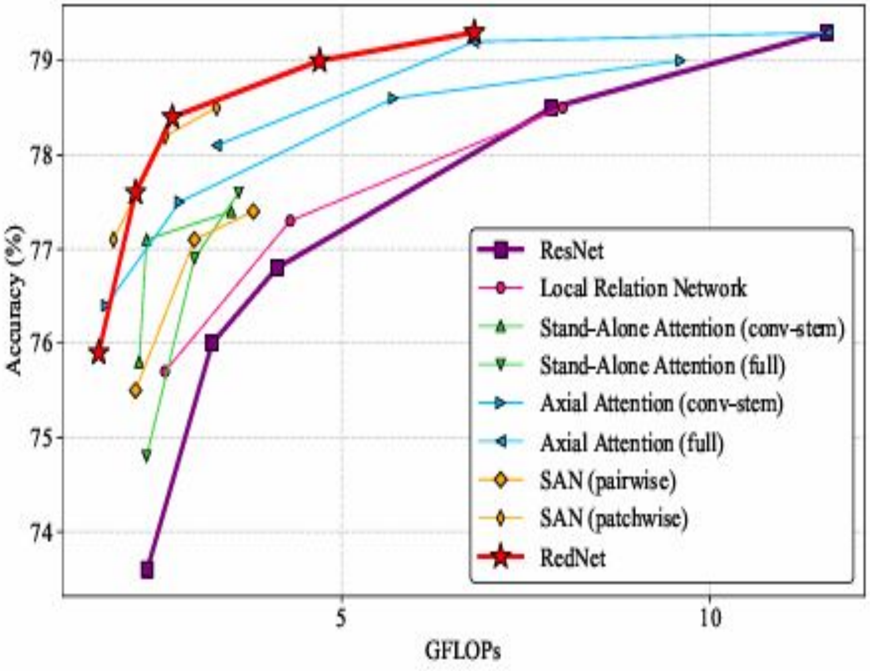


Experiments

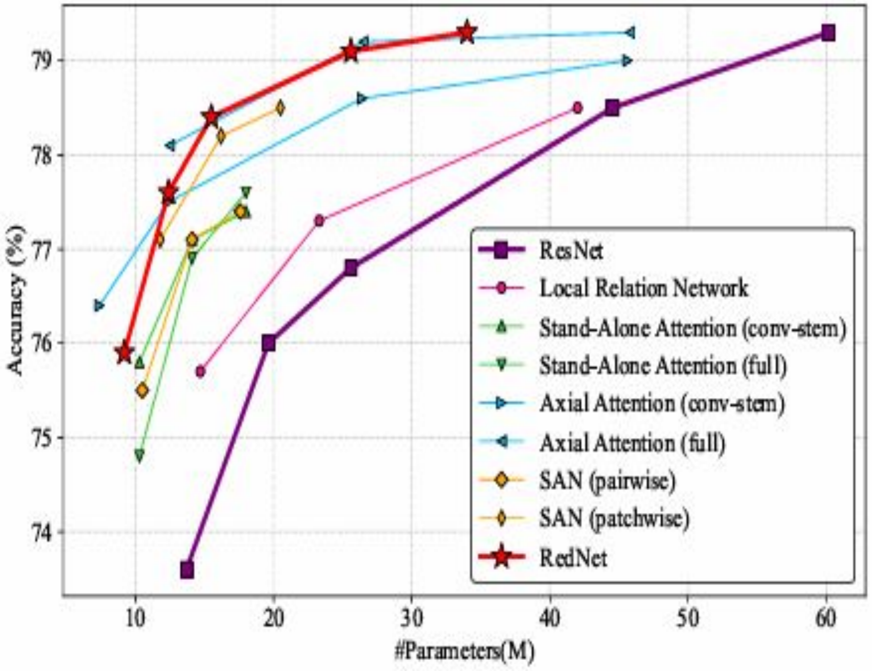
Image Classification

- Dataset - ImageNet
- Preprocessing - random size cropping and horizontal flipping





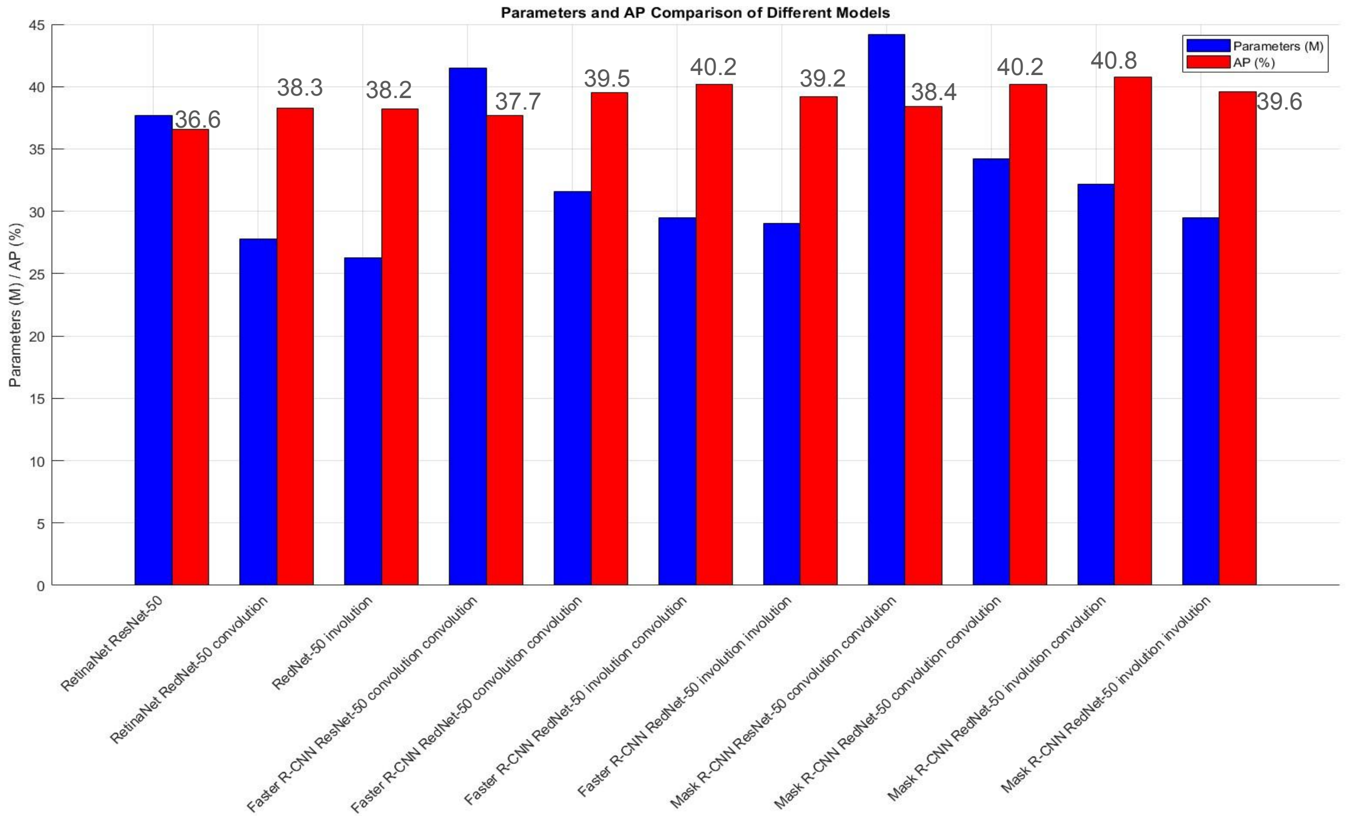
(a) The accuracy-complexity envelope on ImageNet.



(b) The accuracy-parameter envelope on ImageNet.

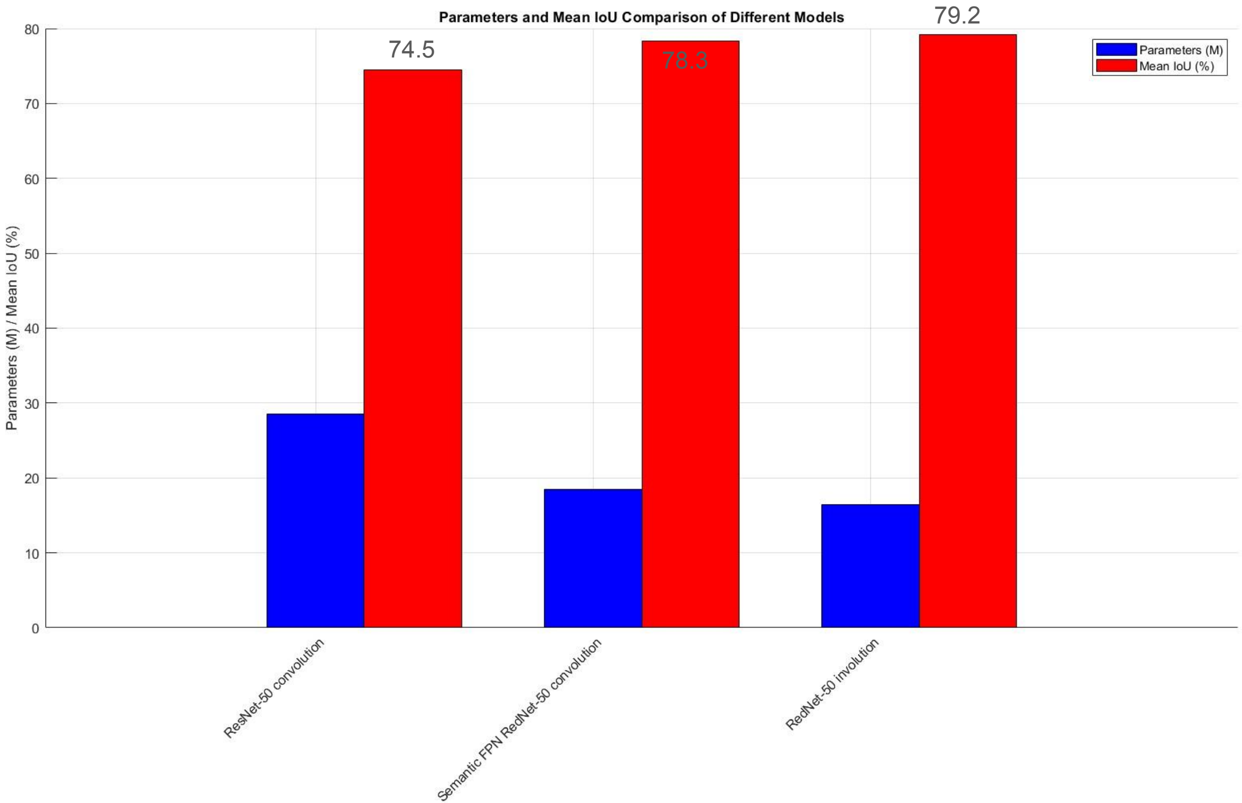
Object Detection and Instance Segmentation

Performance comparison on COCO detection and segmentation

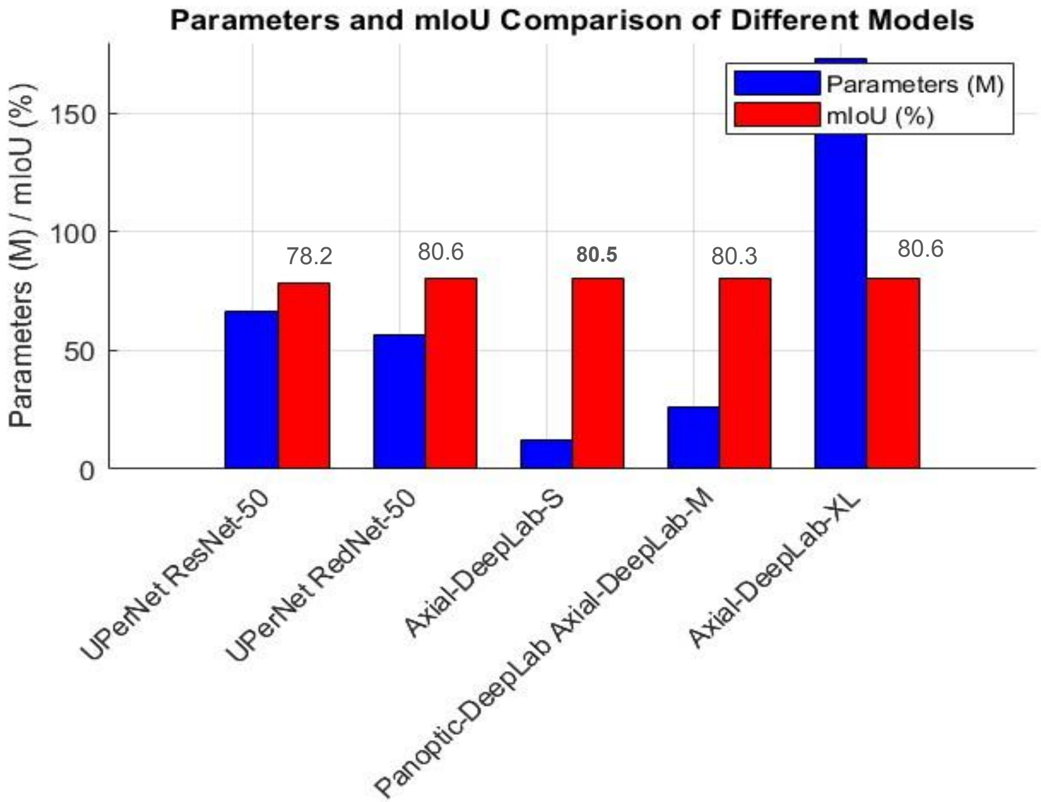


Semantic Segmentation

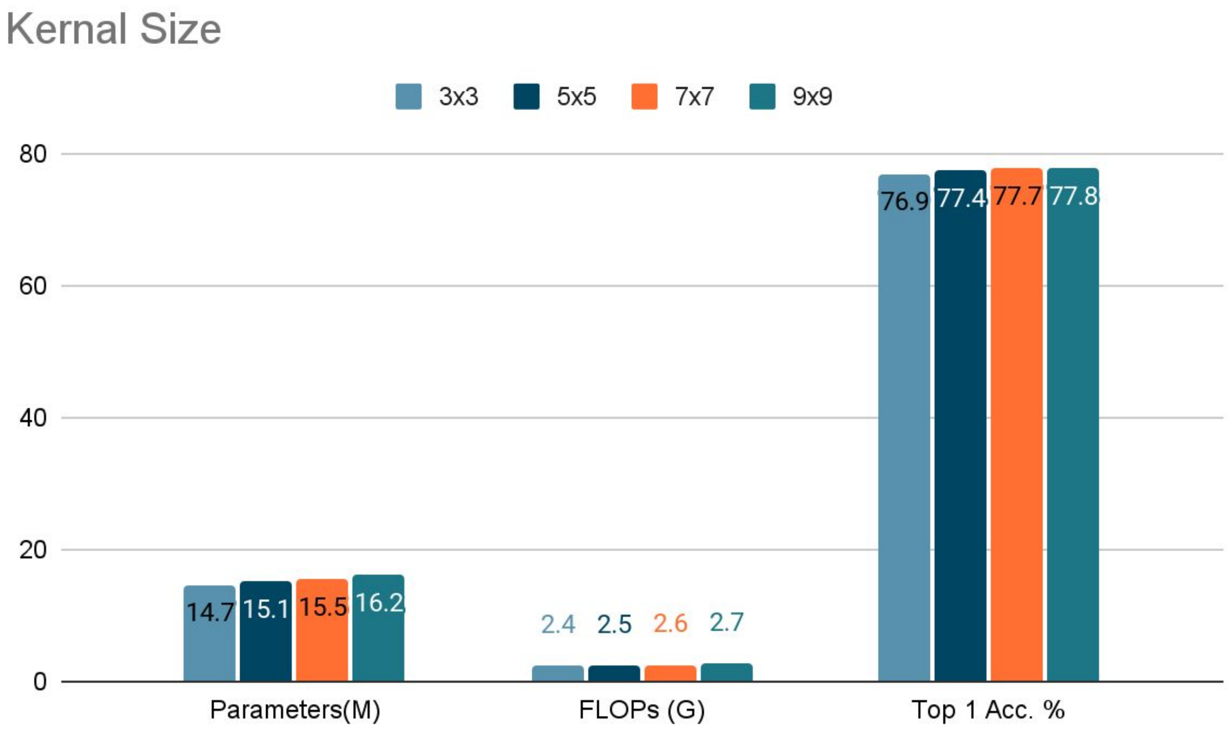
Performance comparison on Cityscapes , segmentation based on Semantic FPN

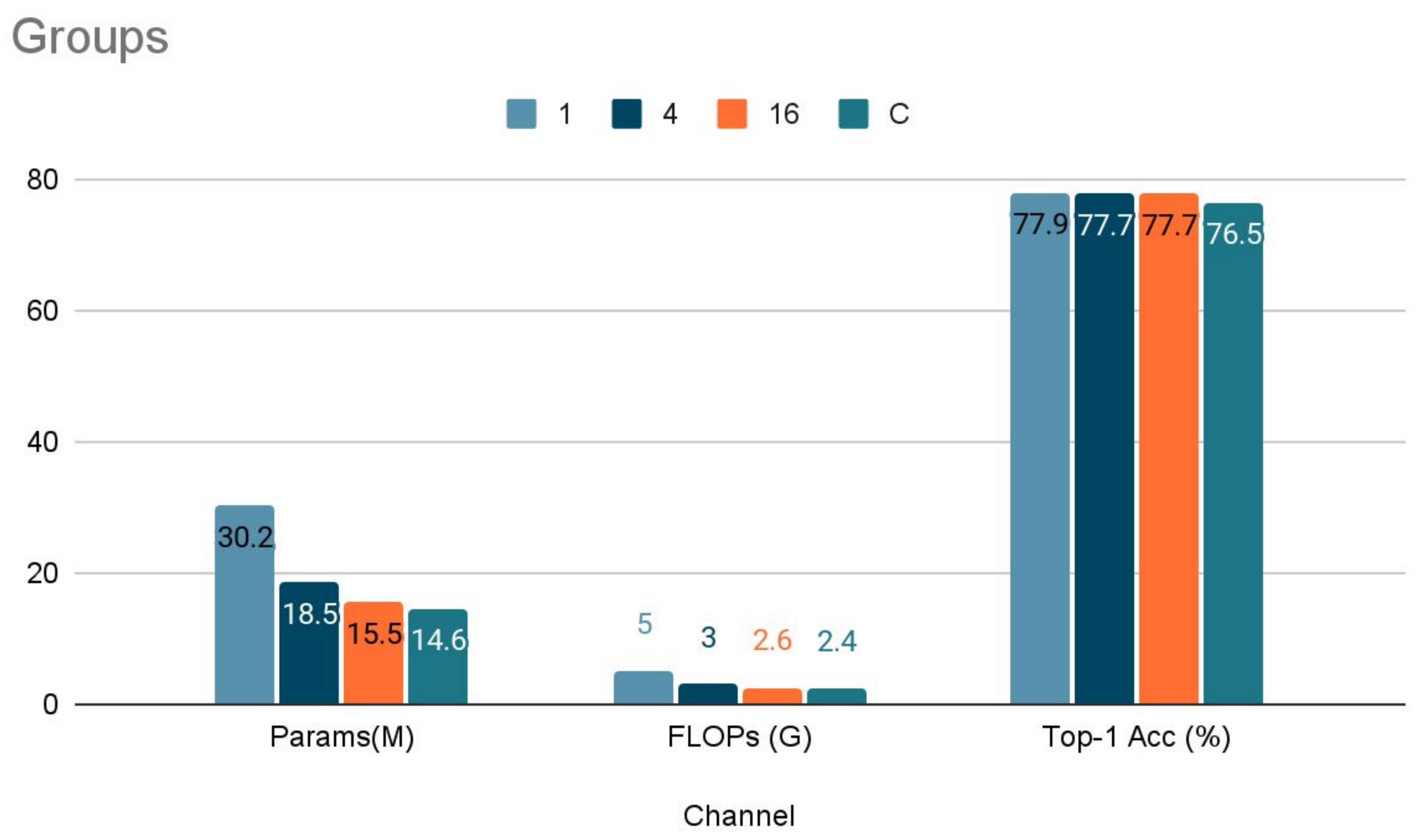


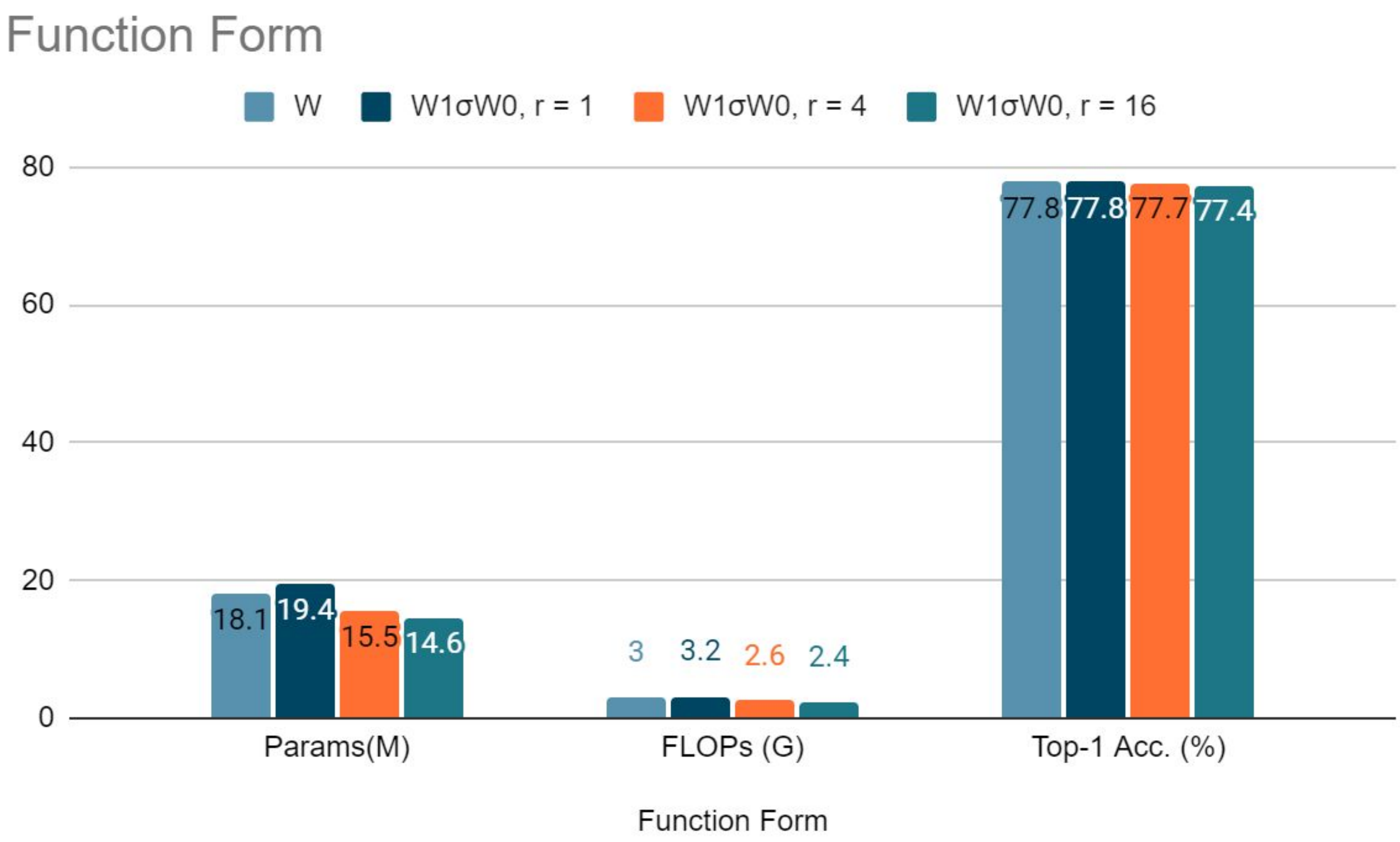
Performance comparison on Cityscapes segmentation based on UPerNet



Ablation Analysis

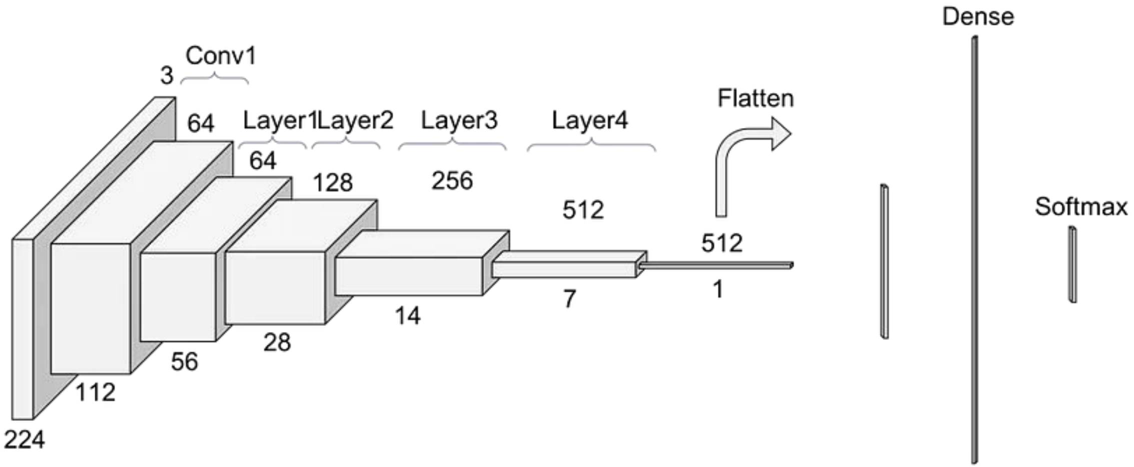



Groups
1
4
16
C
80
60
40
20
0
30.2
18.5
15.5
14.6
5
3
2.6
2.4
77.9
77.7
77.7
76.5
Params(M)
FLOPs (G)
Top-1 Acc (%)
Channel


Function Form
W
W1σW0, r = 1
W1σW0, r = 4
W1σW0, r = 16
80
60
40
20
0
18.1
19.4
15.5
14.6
3
3.2
2.6
2.4
77.8
77.8
77.7
77.4
Params(M)
FLOPs (G)
Top-1 Acc. (%)
Function Form

Stem: Placing 3 × 3 involution at bottleneck position of the stem:

Accuracy: 77.7% to 78.4%



Thank You