Cross-channel Communication Networks

Jianwei Yang[1] **Zhile Ren**[1] **Chuang Gan**[3] **Hongyuan Zhu**[4] **Devi Parikh**[1,2]
[1]Georgia Institute of Technology, [2]Facebook AI Research,
[3] MIT-IBM Watson AI Lab, [4]Institute for Infocomm Research, A*Star, Singapore

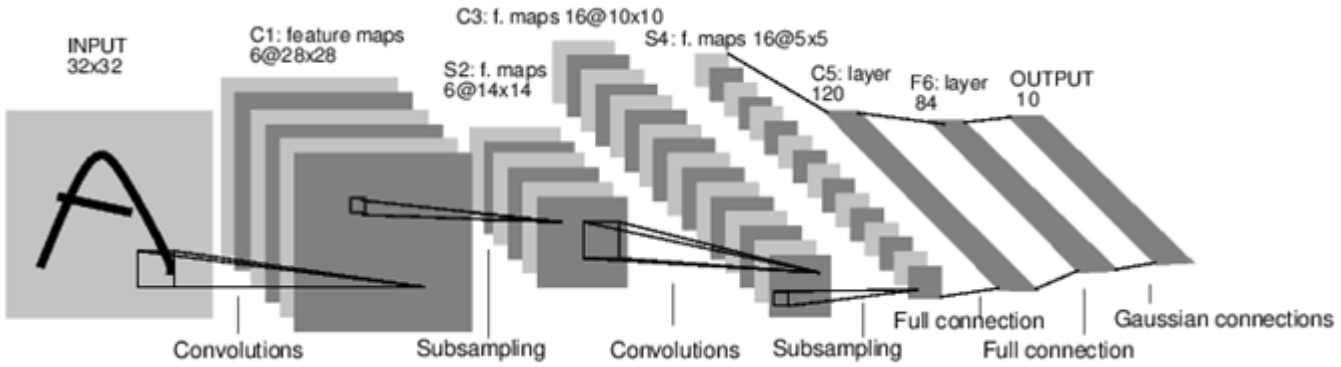
Sanghyeok Chu
sanghyeok.chu@snu.ac.kr

Computer Vision LAB
Department of Electrical and Computer Engineering

서 울 대 학 교
SEOUL NATIONAL UNIVERSITY

Graph Interpretation of Convolution operation

- **We can interpret an operation of convolution filters as “aggregating features of spatial-wise adjacent features/pixels”**
  - Most of CNNs mainly focus on spatial-wise feature aggregation



Channel-wise Aggregation

- **CNNs process input data by feed-forwarding responses to subsequent layers.**
- **But still,**
  - Filters at each layer typically respond to the input response independently.
  - Redundant information could be accumulated across different channels in the same convolutional layer.
  - **Need to encourage information exchange across different channels at the same convolutional layer!**

Previous Works

- **Squeeze-and-Excitation Networks**
  - ILSVRC 2017 winner
  - Global pooling for each channel + 2 fully-connected layers
  - Adaptively recalibrates channel-wise feature responses by explicitly modelling interdependencies between channels.

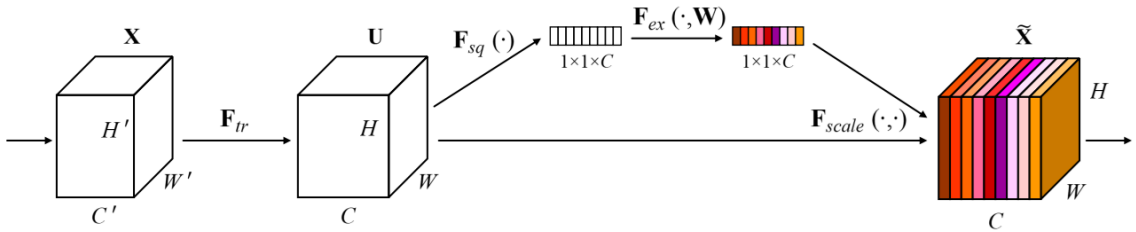
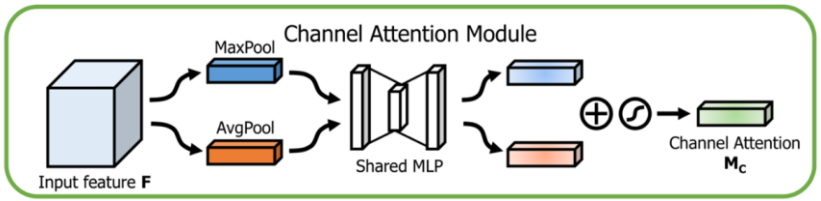


Figure 1: A Squeeze-and-Excitation block.

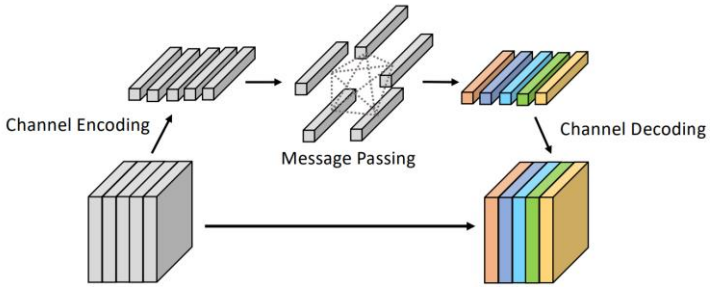
Previous Works

- **CBAM: Convolutional Block Attention Module**
  - ECCV 2018
  - MaxPool + AvgPool + 2 fully-connected layers

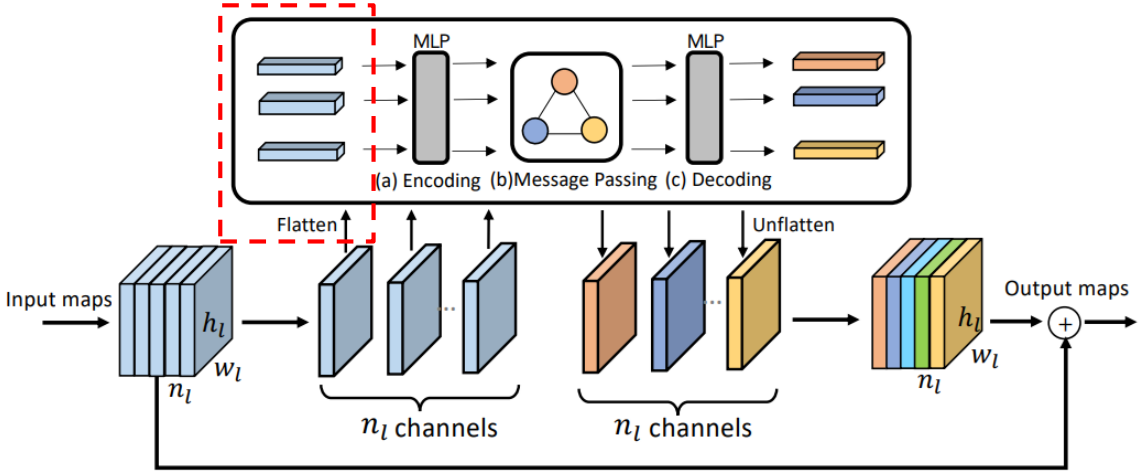


C3B: Channel-Attention Block

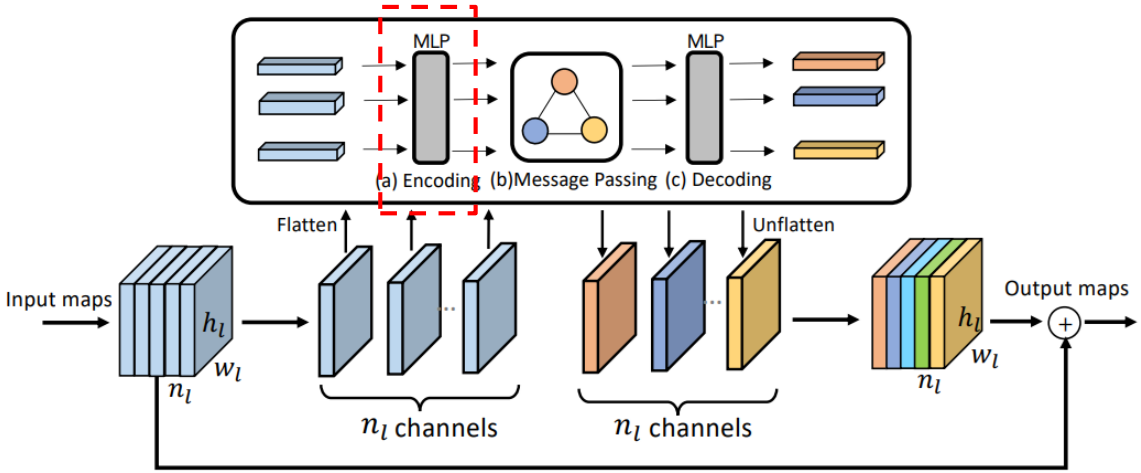
- **Graph interpretation of Channel-Attention**
  - Use similarity matrix as weight of the graph.
  - Explicitly model the channel-wise interaction



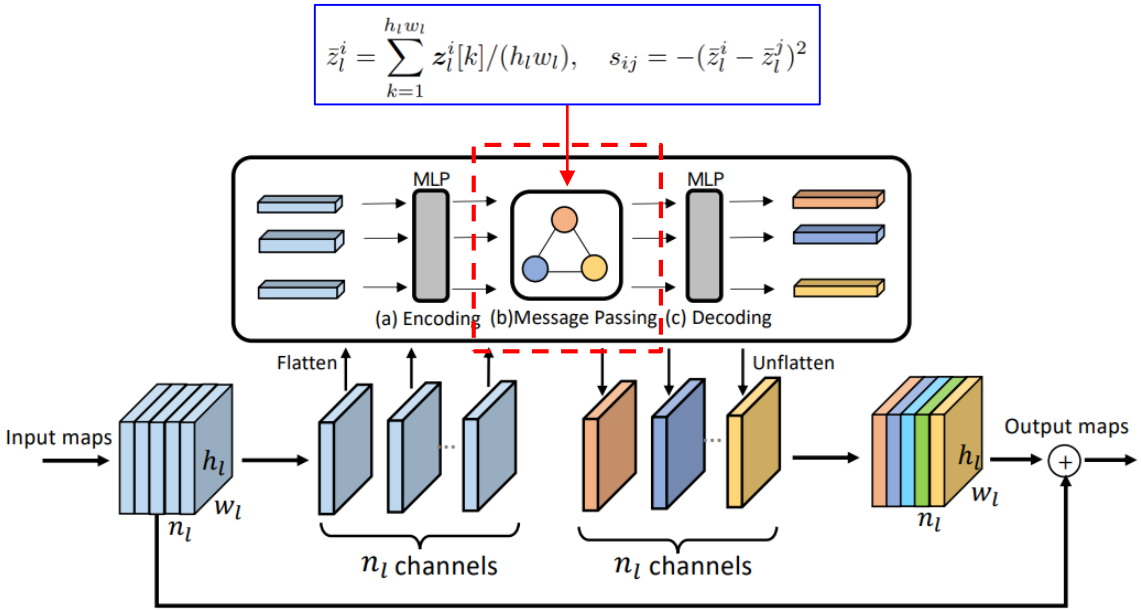
C3B: Channel-Attention Block



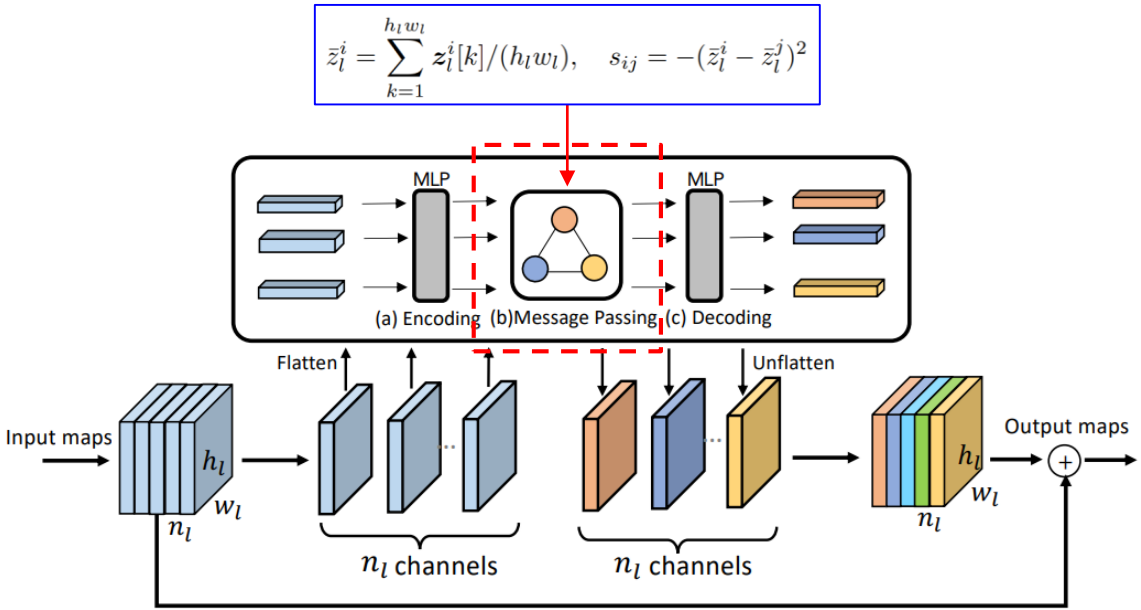
C3B: Channel-Attention Block



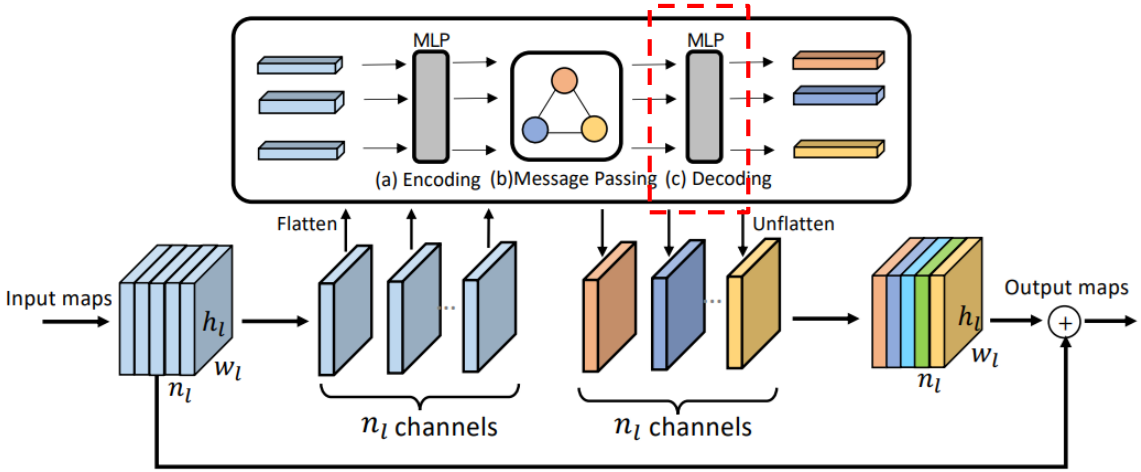
C3B: Channel-Attention Block



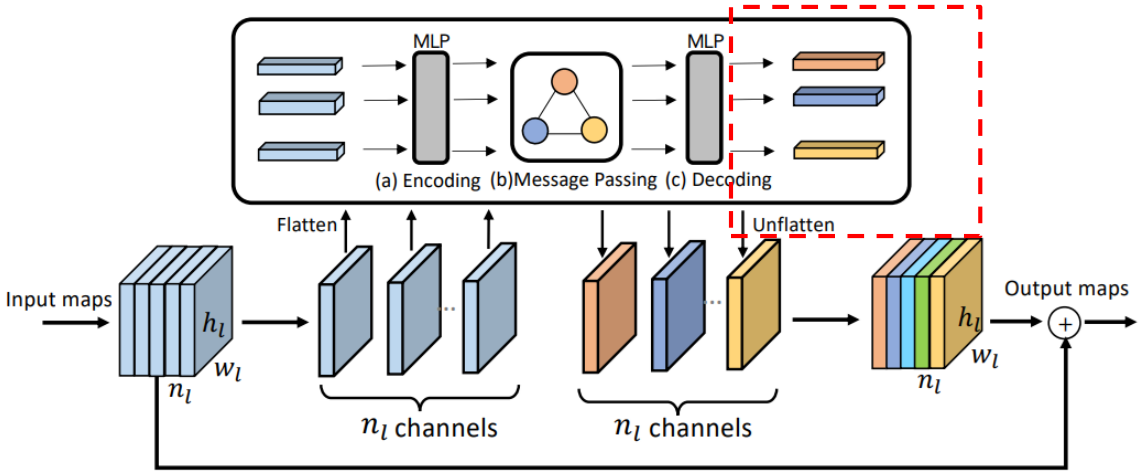
C3B: Channel-Attention Block



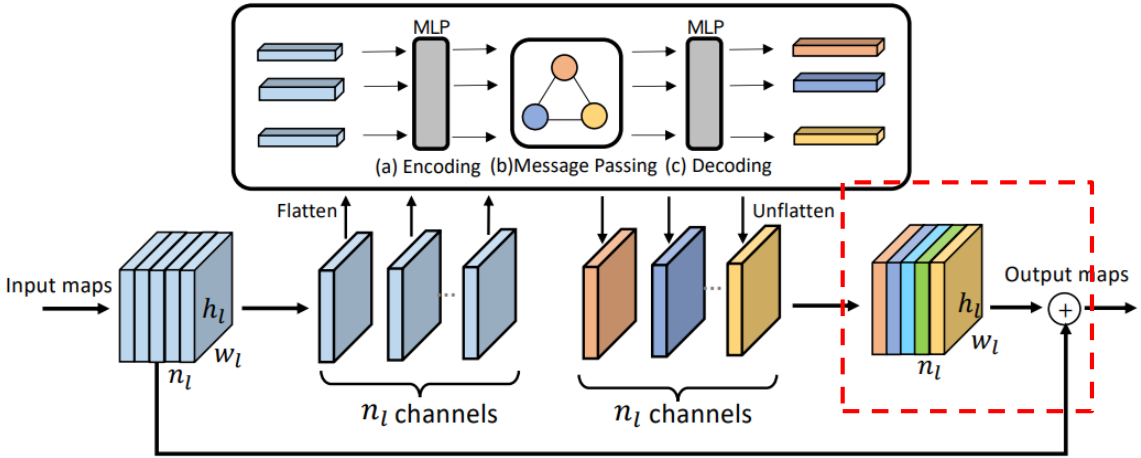
C3B: Channel-Attention Block



C3B: Channel-Attention Block



C3B: Channel-Attention Block



Experimental Results

	ResNet-20			ResNet-56			ResNet-110			Wide-ResNet		
	Size	FLOPs	Acc.	Size	FLOPs	Acc.	Size	FLOPs	Acc.	Size	FLOPs	Acc.
Baseline	0.28	41.7	67.73	0.86	128.2	71.05	1.74	257.9	72.01	26.86	3.84G	77.96
Baseline + SE	0.28	41.8	68.57	0.87	128.5	72.00	1.76	258.5	72.47	27.26	3.84G	**78.57**
Baseline + C3	0.35	46.0	**69.34**	0.93	132.5	**72.27**	1.81	262.2	**73.36**	26.93	3.87G	78.34

Table 1: Classification accuracies (%) on CIFAR-100 [16] with different models.

Segmentation	Mean IOU	Mean Acc.
Deeplabv2 [2]	75.2	85.3
Deeplabv2 + SE	**75.6**	85.6
Deeplabv2 + C3	**75.7**	**86.0**

Detection	Pascal VOC	COCO
Faster R-CNN [20]	74.6	33.9
Faster R-CNN + SE	74.8	34.3
Faster R-CNN + C3	**75.6**	**34.8**

Table 3: Performance on semantic segmentation on PASCAL-VOC-2012 (left) and object detection on PASCAL-VOC-2007 and COCO (right) with/without cross-channel communication). Bold indicates best results. For detection, mAP@(IOU=0.5) is reported for PASCAL-VOC-2007 and mAP@(IOU=0.5:0.95) is reported for COCO.

Experimental Results

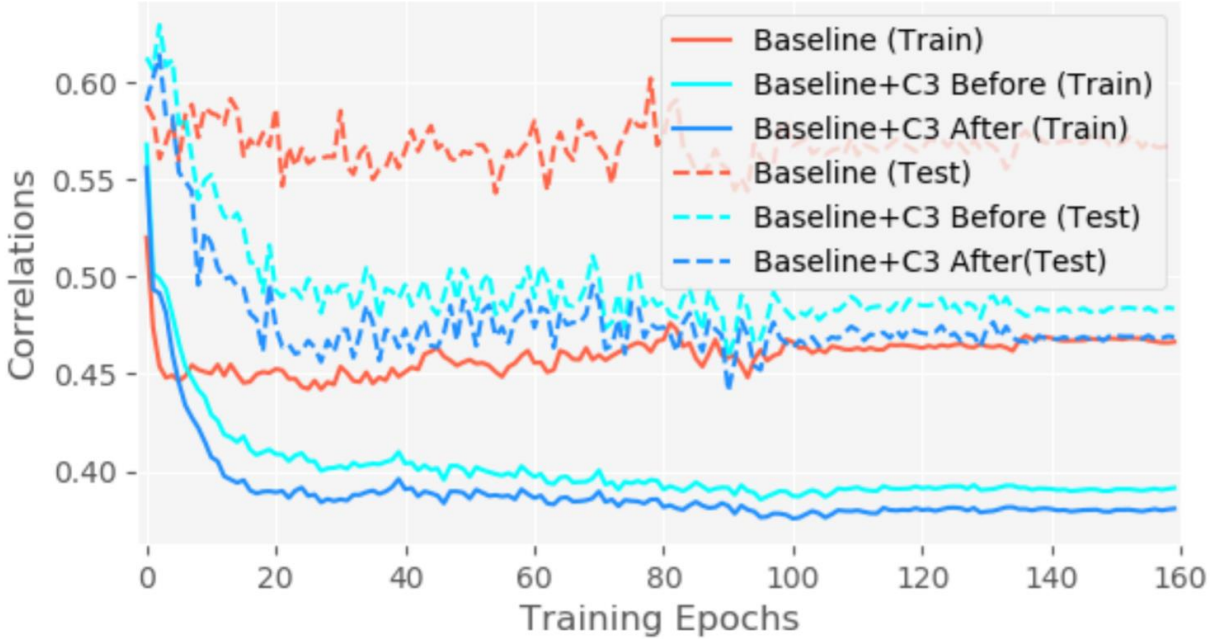


Figure 4: Correlations for models at different training stages.

Experimental Results

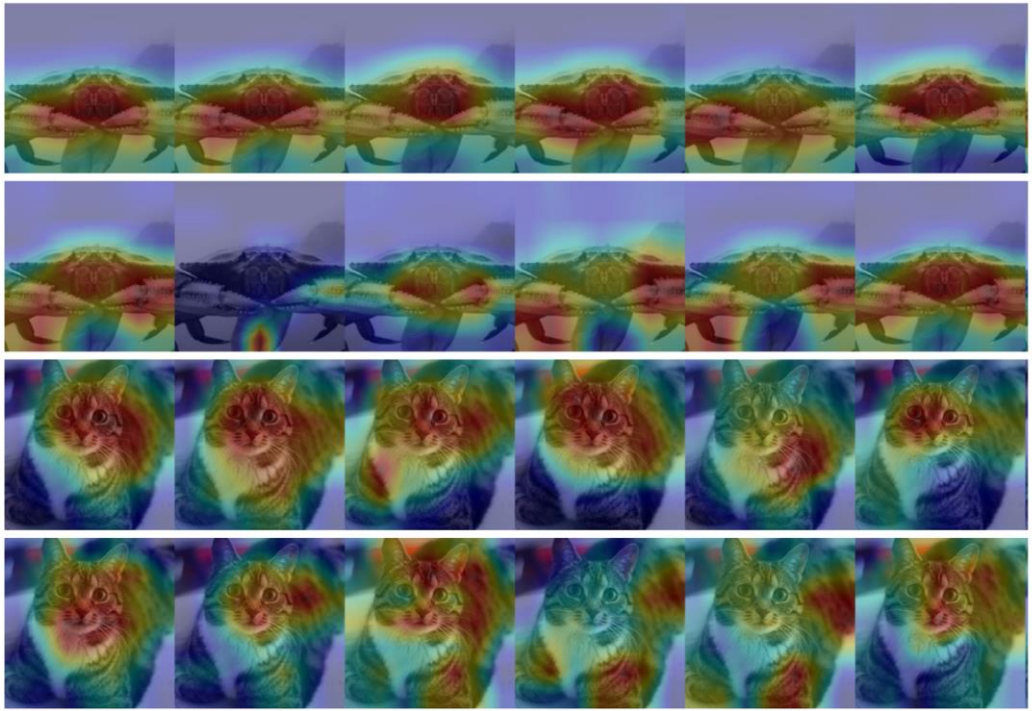


Figure 6: We visualize the top-6 mostly activated channels at the last layer. Odd rows are the class activation map from baseline ResNet-101; Even rows are from our model.

Code analysis

Cross-channel Communication Networks: Code analysis

Jianwei Yang, Zhile Ren, Chuang Gan, Hongyuan Zhu, Devi Parikh, NeurIPS 2019.

Quick review

- **Cross-channel Communication**: Filters at each layer independently generate responses given the input and do not communicate with each other
- Propose a network unit called Cross-channel Communication (C3) block

> Simple yet effective module to encourage the neuron communication within the same layer.
> With C3 block, each neuron accounts for the channel-wise responses from other neurons at the same layer and learns more discriminative and complementary representations

```
1 from IPython.display import Image, display
2 display(Image(filename='report/fig1.png'))
```

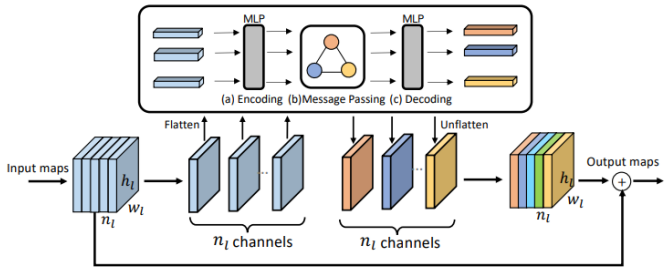


Figure 2: An overview of the Cross-channel Communication (C3) block. The feature responses in channels are passed to an encoder, and then the information is exchanged with other channels using a message passing mechanism. Finally, the features are decoded and added back to the input responses for the recalibration.

Code analysis

Code analysis

Git repo: https://github.com/jwyang/C3Net.pytorch

Implementation of C3B (cross_neuron_distributed.py)

```
'''Code explanation'''
class _CrossNeuronBlock(nn.Module):
    def __init__(self, in_channels, in_height, in_width, nblocks_channel=8, spatial_height=32, spatial_width=32, reduction=4, size_is_consistant=True):
        super(_CrossNeuronBlock, self).__init__()
        self.nblocks_channel = 1 if in_channels <= 512 else in_channels // 512 # nblock_channel: number of block along channel axis
        factor = in_height // spatial_height
        self.spatial_height = in_height // factor
        self.spatial_width = in_width // factor
        self.spatial_area = self.spatial_height * self.spatial_width
        '''feature encoder/decoder: MLP  in the figure is implemented by torch.Conv1d'''
        self.fc_in = nn.Sequential(
            nn.Conv1d(self.spatial_area, self.spatial_area // reduction, 1, 1, 0, bias=True),
            nn.ReLU(True),
            nn.Conv1d(self.spatial_area // reduction, self.spatial_area, 1, 1, 0, bias=True),
        )
        self.fc_out = nn.Sequential(
            nn.Conv1d(self.spatial_area, self.spatial_area // reduction, 1, 1, 0, bias=True),
            nn.ReLU(True),
            nn.Conv1d(self.spatial_area // reduction, self.spatial_area, 1, 1, 0, bias=True),
        )
```

Code analysis

```
    def forward(self, x):
        bt, c, h, w = x.shape
        residual = x
        '''flatten feature matrices into vectors'''
        x_stacked = x.view(bt * self.nblocks_channel, c // self.nblocks_channel, self.spatial_height * self.spatial_width) # (b) x c x (h * w)
        x_v = x_stacked.permute(0, 2, 1).contiguous() # (b) x (h * w) x c

        '''feature encoding'''
        x_v = self.fc_in(x_v) # (b) x (h * w) x c

        '''average the output by x_v.mean(1) to increase the robustness in message passing period (x_m was denoted as z_i in the paper)'''
        x_m = x_v.mean(1).view(-1, 1, c // self.nblocks_channel) # b x 1 x c

        '''calculate negative square distances'''
        score = -(x_m - x_m.permute(0, 2, 1).contiguous())**2 # b x c x c

        '''feed softmax to make the normalized attention scores'''
        attn = F.softmax(score, dim=1) # (b * h * w) x c x c

        '''[b x (h*w) x c] * [b x c x c] = [b x (h*w) x c]'''
        z = torch.bmm(x_v, attn)

        '''feature decoding'''
        out = self.fc_out(z)

        '''reshape the output vectors into feature map matrices'''
        out = out.permute(0, 2, 1).contiguous().view(bt, c, self.spatial_height, self.spatial_width) # b x c x h x w

        '''added back to input response '''
        out = F.relu(residual + out)

        return out
```

Code analysis

Implementation of C3B wrapping function to original networks

```python
'''C3B can be added to any network by this wrapping function'''
class CrossNeuronWrapper(nn.Module):
    def __init__(self, block, in_channels, in_height, in_width, spatial_height, spatial_width, reduction=8):
        super(CrossNeuronWrapper, self).__init__()
        self.block = block
        self.cn = CrossNeuronlBlock2D(in_channels, in_height, in_width, spatial_height, spatial_width, reduction=reduction)

    def forward(self, x):
        x = self.cn(x)
        x = self.block(x)
        return x
```

Git Repo

- https://github.com/sanghyeokchu/Channl-communication-Network
  - sh run_ccn.sh
    - argument: --c3b
      - 0(w/o) C3B
      - 1(with) C3B
  - Code analysis
    - Cross-channel Communication Networks_2019-23655 추상혁.ipynb
    - How to insert C3B into the network is notified at the end of the ipynb file.
- References
  - C3B Block
    - https://github.com/jwyang/C3Net.pytorch
  - ResNet20 CIFAR10 training code
    - https://github.com/akamaster/pytorch_resnet_cifar10

Requirements

- Requirements:
  - pytorch
  - torchvision
  - scipy

Experimental Result

- Trained on CIFAR-10 dataset
  - Top1 accuracy
  - w/o C3B
    - ResNet20: **88.83** @ epoch 150
    - ResNet110: 90.11 @ epoch 150
  - With C3B
    - ResNet20: 88.31 (-0.52%) @ epoch 150
    - ResNet110: **91.47** (+1.36%) @ epoch 150
  - More powerful when it is attached on large model.

Summary

- By designing a C3B to make channels communicate each other in explicit way,
  - Channels at the same layer can capture global information and calibrate with other channels
  - Therefore models can learns more diverse and complementary representations.
  - It can be added to any network regardless of its architecture