Cross-channel Communication Networks

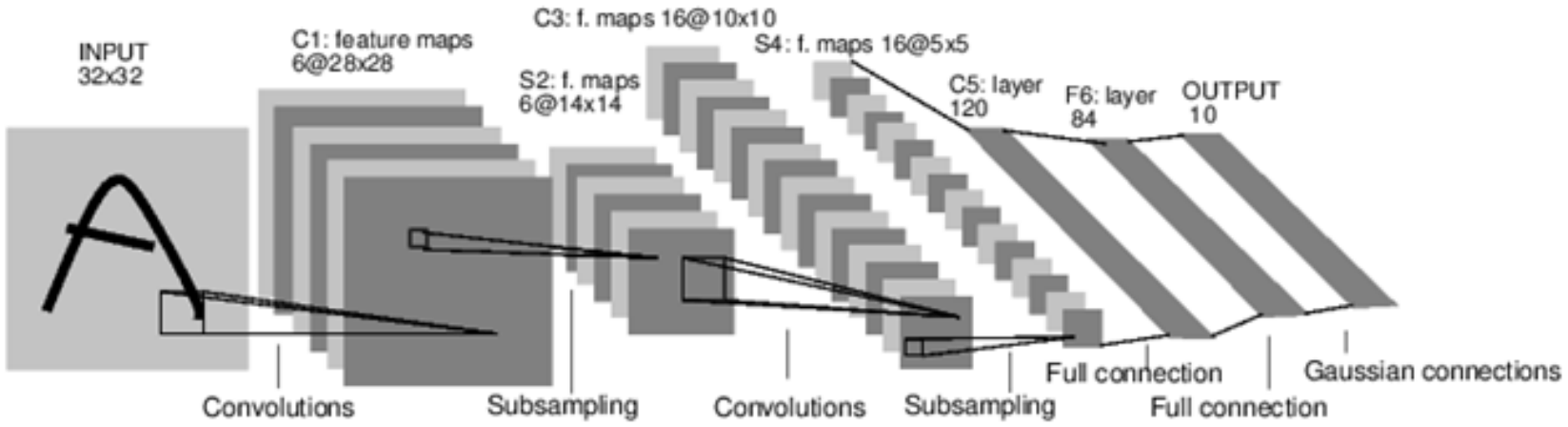
Jianwei Yang[1] **Zhile Ren**[1] **Chuang Gan**[3] **Hongyuan Zhu**[4] **Devi Parikh**[1,2]
[1]Georgia Institute of Technology, [2]Facebook AI Research,
[3] MIT-IBM Watson AI Lab, [4]Institute for Infocomm Research, A*Star, Singapore

SangHyeok Chu

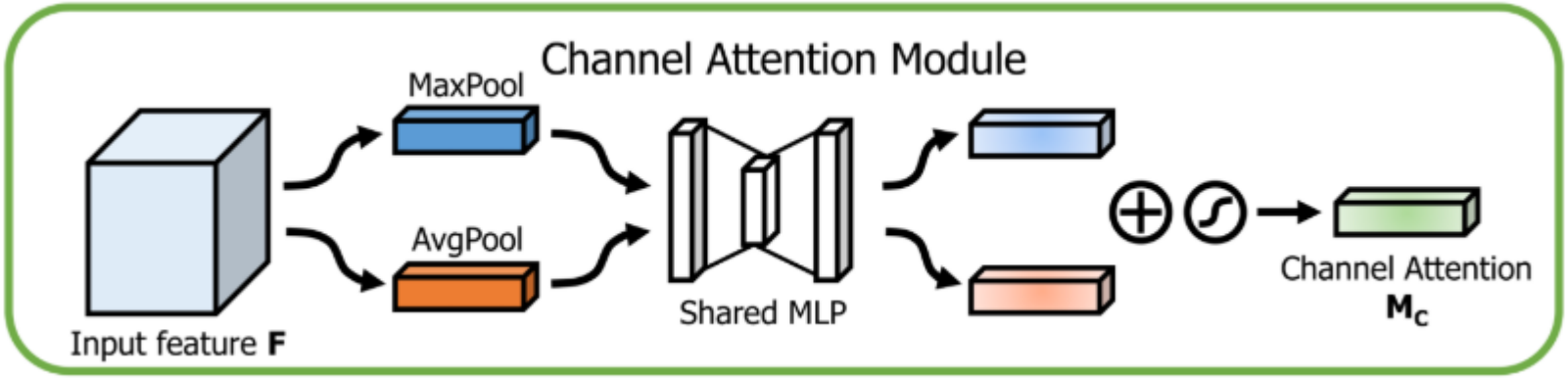
Seoul National University

Graph Interpretation of Convolutional Neural Networks

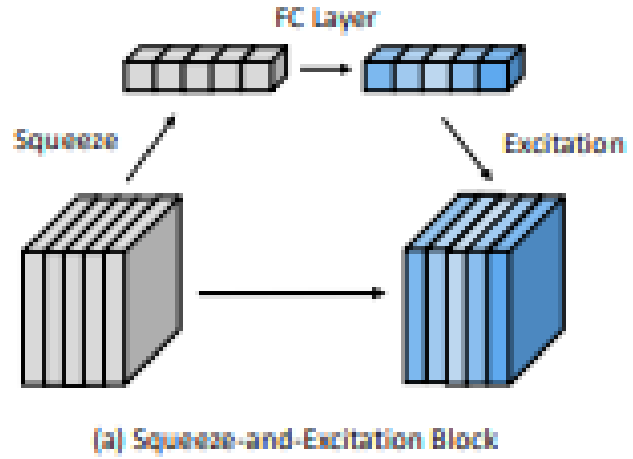
- We can interpret an operation of convolution filters as “aggregating features of spatial-wise adjacent features/pixels”
- Most of Convolutional neural networks mainly focus on spatial-wise feature aggregation.



Channel-wise Aggregation?

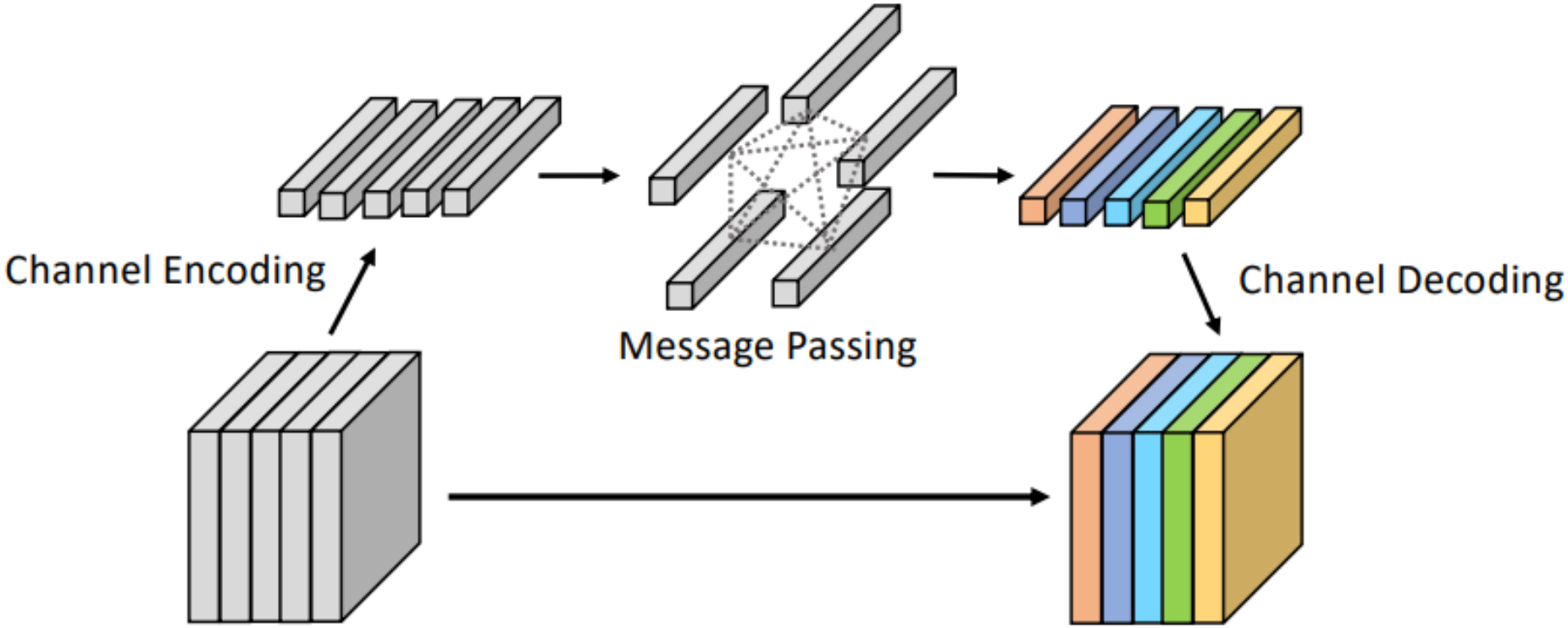


Woo, Sanghyun, et al. "Cbam: Convolutional block attention module." *ECCV*. 2018.



Hu, Jie, Li Shen, and Gang Sun. "Squeeze-and-excitation networks." *CPPR*. 2018.

C3B: Channel-Attention Block



C3B: Channel-Attention Block

$$\boldsymbol{y}_l^i = f_{enc}^{in}(\boldsymbol{x}_l^i), \quad \boldsymbol{z}_l^i = f_{enc}^{out}(\sigma(\boldsymbol{y}_l^i))$$

$$\bar{z}_l^i = \sum_{k=1}^{h_l w_l} \boldsymbol{z}_l^i[k]/(h_l w_l), \quad s_{ij} = -(\bar{z}_l^i - \bar{z}_l^j)^2, \quad \hat{\boldsymbol{z}}_l^i = \sum_{j=1}^{n_l} a_{ji} \boldsymbol{z}_l^j$$

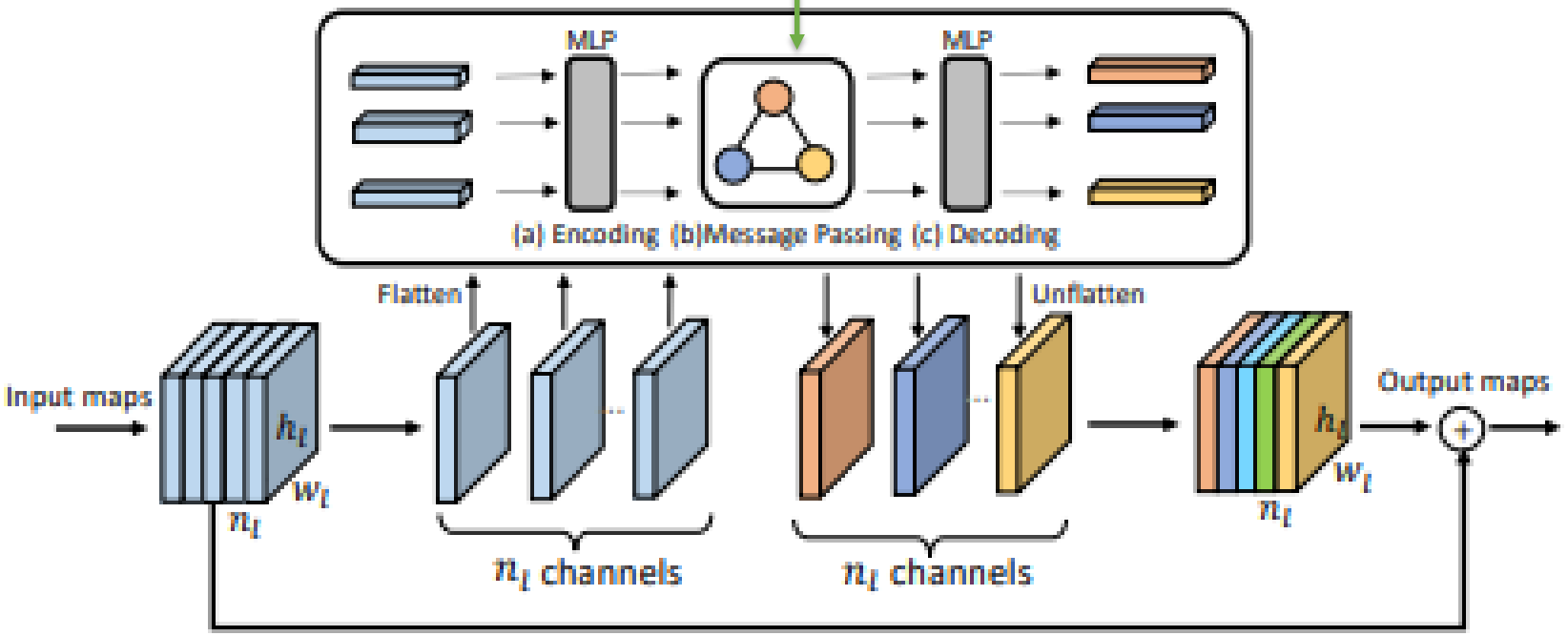


Figure 2: An overview of the Cross-channel Communication (C3) block. The feature responses in channels are passed to an encoder, and then the information is exchanged with other channels using a message passing mechanism. Finally, the features are decoded and added back to the input responses for the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

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

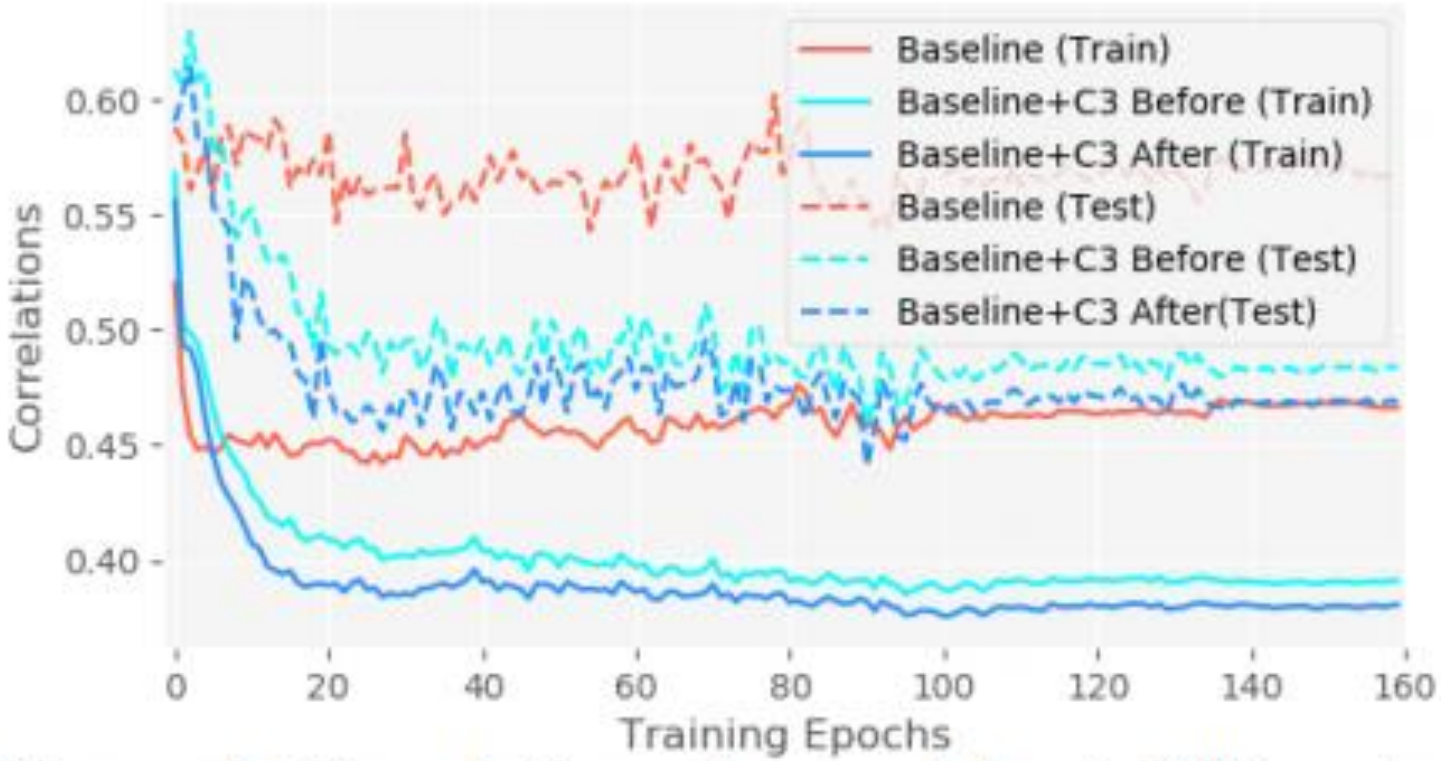


Figure 4: Correlations for models at different training stages.

Experimental Results

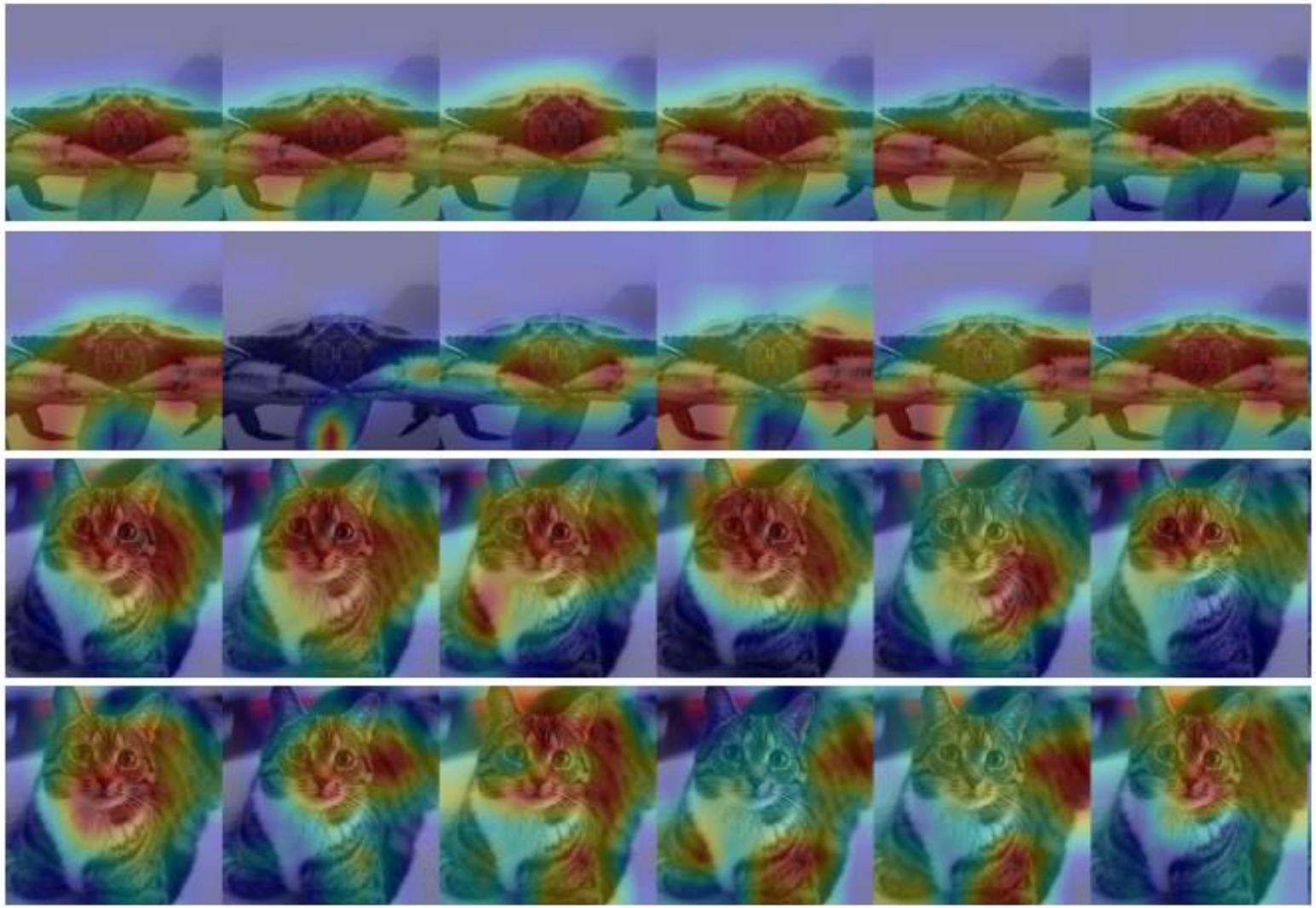


Figure 6: We visualize the top-6 mostly activated channels at the last layer. Odd rows are the class activation map from baseline ResNet-101; Even rows are from our model.

Official Implementation of C3B

```
class _CrossNeuronBlock(nn.Module):
    def __init__(self, in_channels, in_height, in_width,
                 nblocks_channel=4,
                 spatial_height=24, spatial_width=24,
                 reduction=8, size_is_consistant=True):
        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),
        )

        self.bn = nn.BatchNorm1d(self.spatial_area)
    def forward(self, x):
        x_stacked = x_stretch # (b) x c x (h * w)
        x_stacked = x_stacked.view(bt * self.nblocks_channel, c // self.nblocks_channel, -1)
        x_v = x_stacked.permute(0, 2, 1).contiguous() # (b) x (h * w) x c
        x_v = self.fc_in(x_v) # (b) x (h * w) x c
        x_m = x_v.mean(1).view(-1, 1, c // self.nblocks_channel).detach() # (b * h * w) x 1 x c
        score = -(x_m - x_m.permute(0, 2, 1).contiguous())**2 # (b * h * w) x c x c
        # score.masked_fill_(self.mask.unsqueeze(0).expand_as(score).type_as(score).eq(0), -np.inf)
        attn = F.softmax(score, dim=1) # (b * h * w) x c x c
        out = self.bn(self.fc_out(torch.bmm(x_v, attn))) # (b) x (h * w) x c
        out = out.permute(0, 2, 1).contiguous().view(bt, c, h, w)
        return F.relu(residual + out)
```

https://github.com/jwyang/C3Net.pytorch/blob/master/lib/layers/cross_neuron.py