ResNeSt-Split attention Network (IEEE-CVPR 2020.)

Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Haibin Lin, Zhi Zhang, Yue Sun, Tong He, Jonas Mueller, R. Manmatha, Mu Li, Alexander Smola

Student Mentor: -
Sai Harsha Yelleni sir

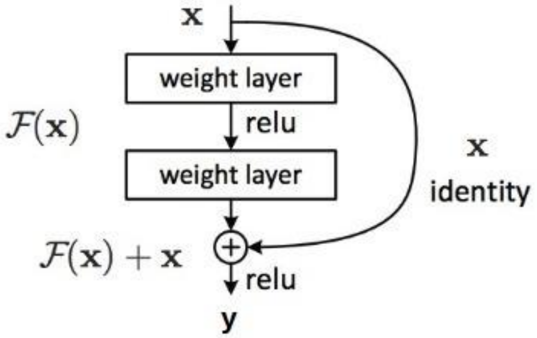
Faculty Mentor: -
Prof. C.K. Mohan sir

Presented By: -
Shubham Jain
(SM20MTECH12007)

Contents: -

ResNet: Identity Connection

- ResNet makes use of the Identity Connection, which helps to protect the network from vanishing gradient problem.

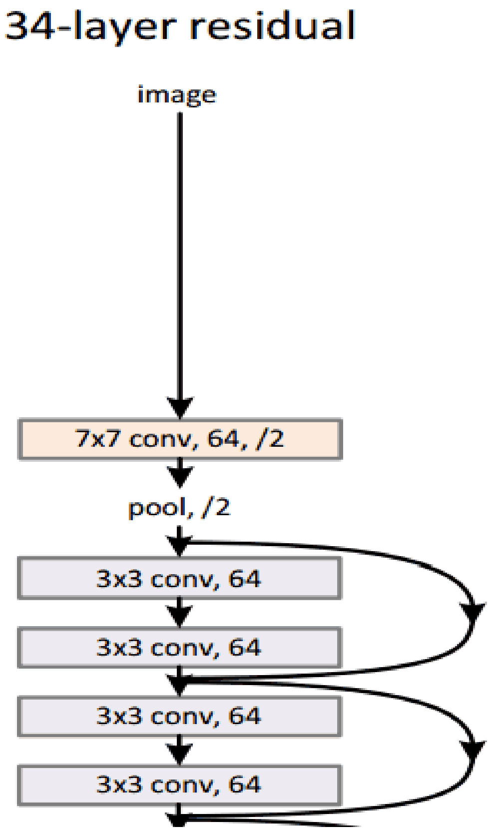


$$y = x + F(x)$$

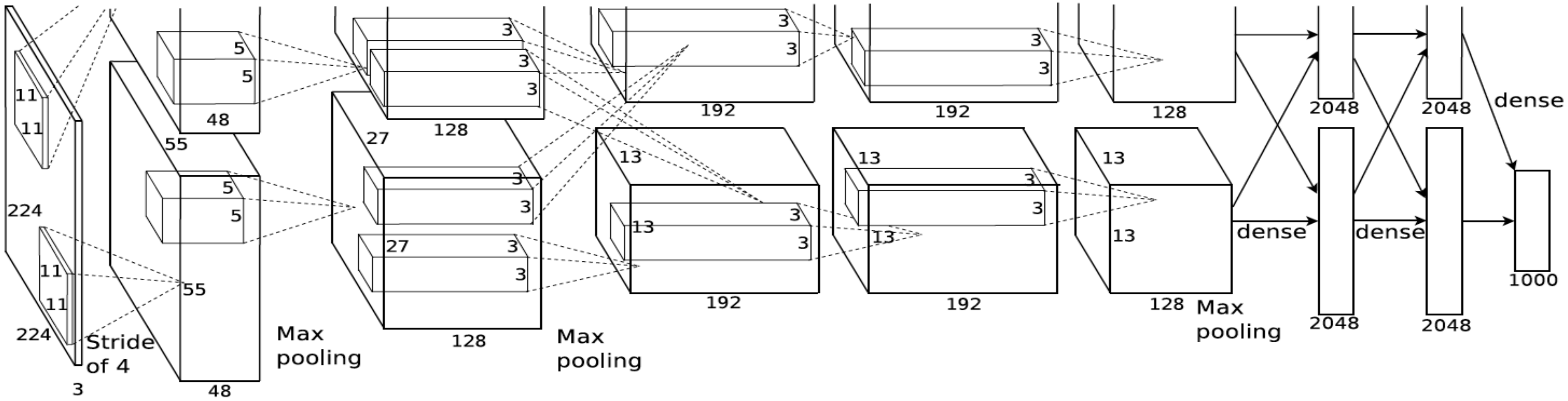
$$\frac{\delta E}{\delta x} = \frac{\delta E}{\delta y} * \frac{\delta y}{\delta x} = \frac{\delta E}{\delta y} * (1 + F'(x))$$

$$= \frac{\delta E}{\delta y} + \frac{\delta E}{\delta y} * F'(x)$$

- ResNet uses Batch Normalization. The problem of covariate shift is mitigated.

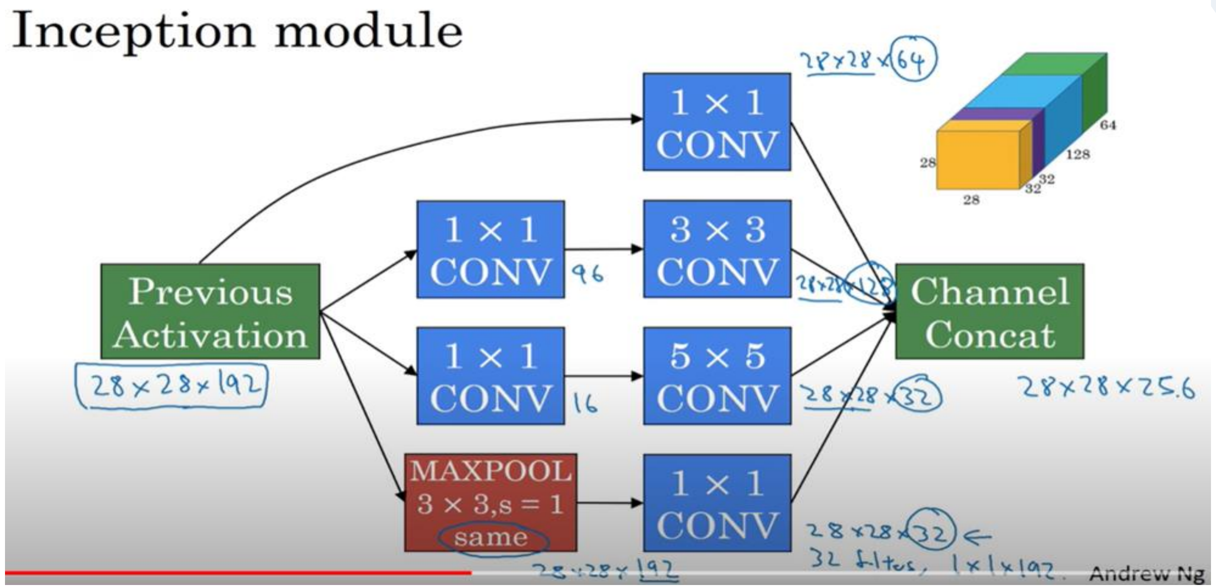


AlexNet: Group Convolution



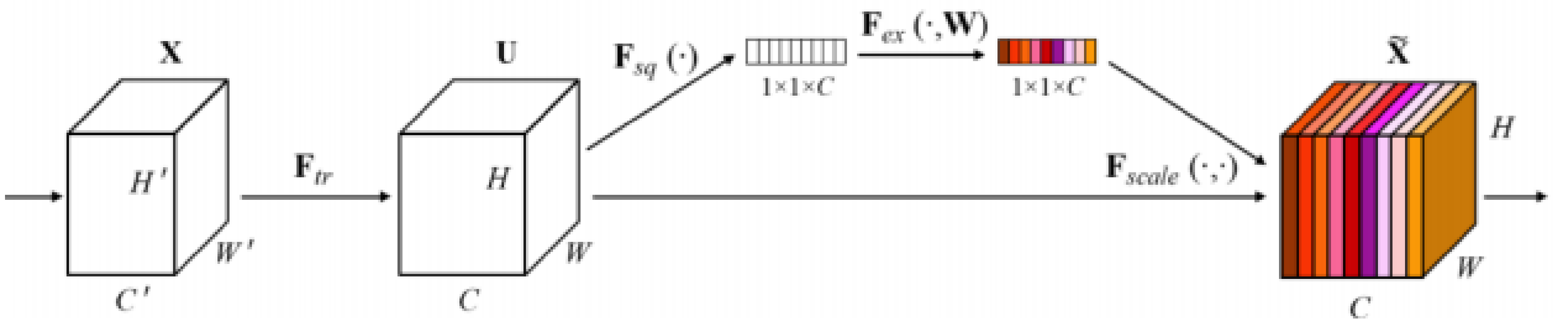
- Requires lesser number of parameter
- Model Parallelism
- Less Correlation

GoogleNet: Inception Network



There are multiple kernels per layer - resulting in multiple channel outputs per layer. This leads to wider networks helping a network learn a varied set of low-level and high-level features.

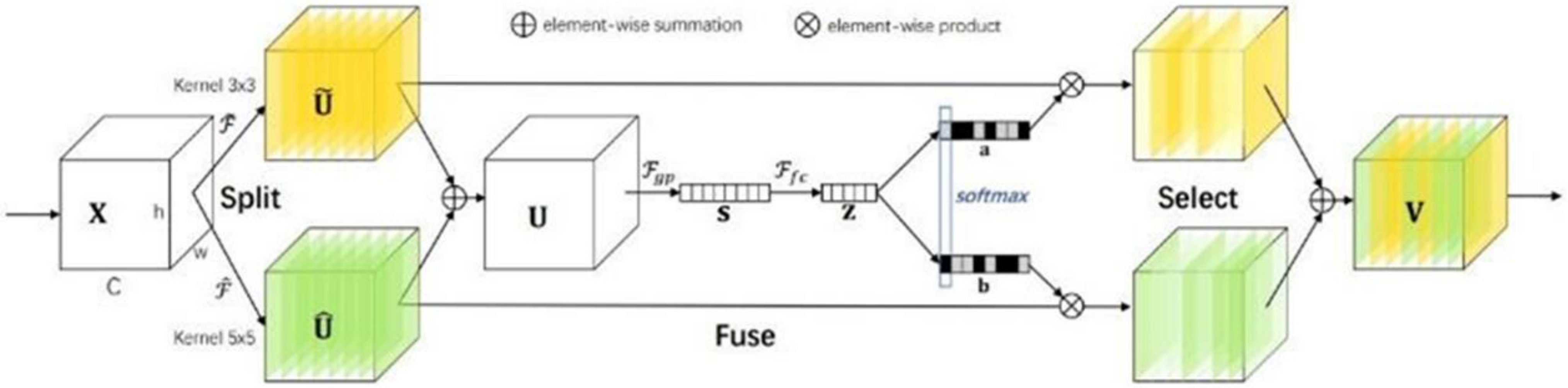
SE-Net: Squeeze and Excitation Network



The figure described detail three steps in Squeeze and Excitation network:

1) Transforming input to features map.
2) Squeezing: applying Global Average Pooling into feature maps in channel dimension
3) Excitation: applying two linear transformations on the result of Squeezing phase, and then gating it by a sigmoid function.

SK-Net: Selective Kernel Network



Feature-maps → Multiple size kernels → Element-wise sum → Global pooling → Dimension reduction → Multiple FC layers → Softmax gating → Element-wise multiply → Split Attention

ResNeSt: Split attention Network

“Picking up concepts from

ResNet

AlexNet

GoogleNet

SE-Net (Squeeze and Excitation)

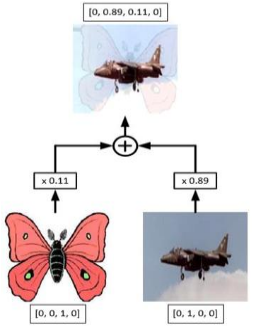
SK-Net (Selective Kernel)

and using them

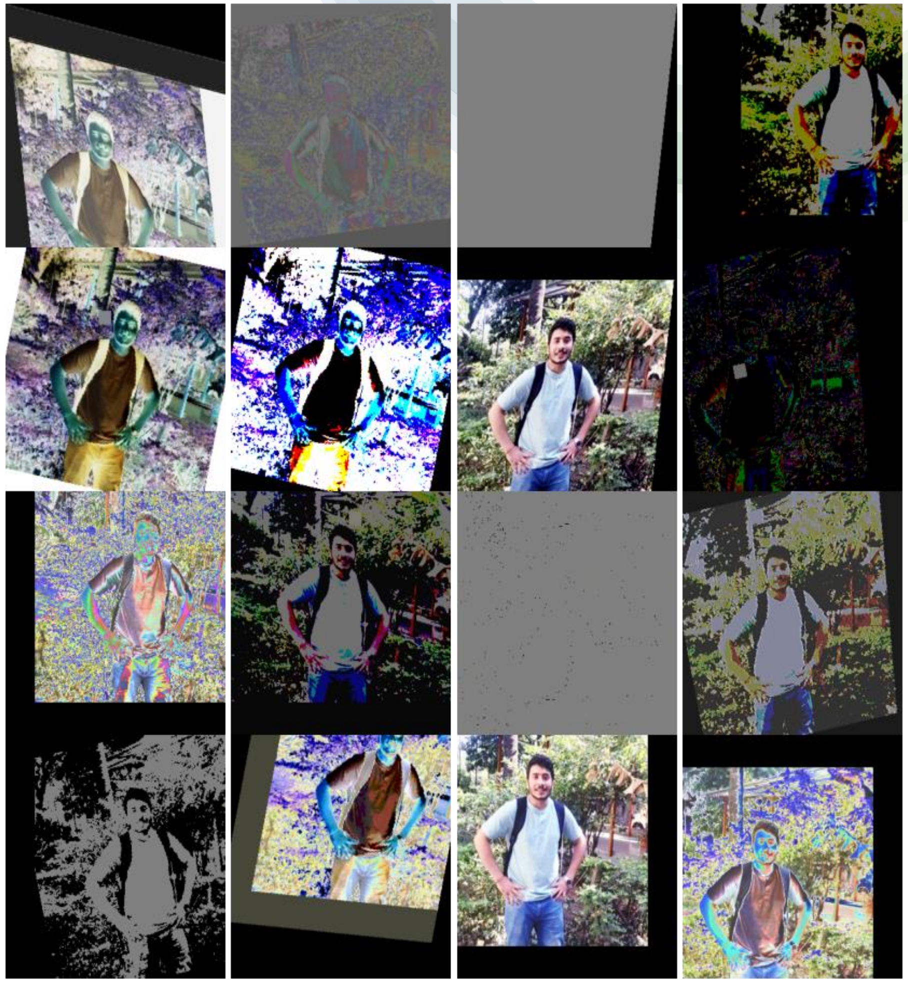
to improve performance is key Idea of ResNeSt.”

Implementation

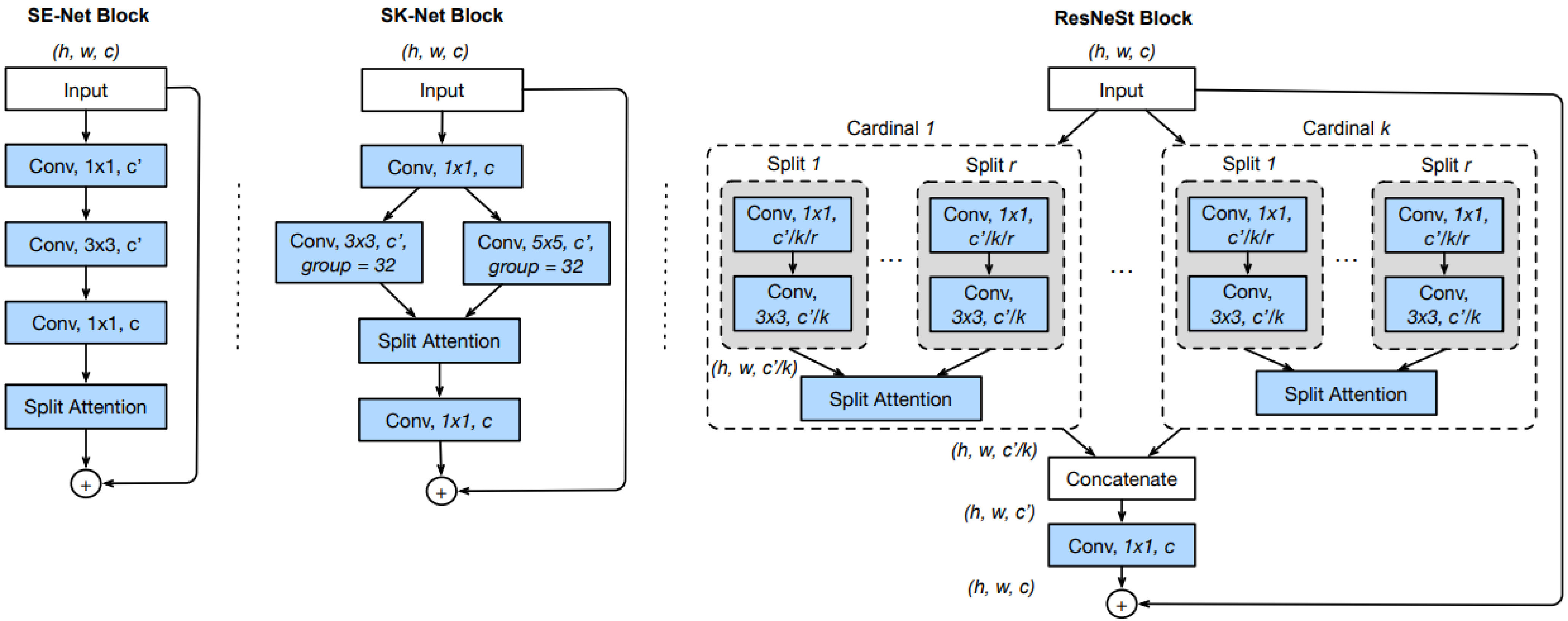
- Dataset Used CIFAR10 (airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks)
- Task:
  - Pre-processing data as given in research paper.
    - Random Crop ⃠
    - Large Crop size ⃠
    - Mix-up Training ✓
    - Auto Augmentation ✓



  - Implementing ResNeSt-50 as backbone for classification on CIFAR10 dataset.
  - Training And Validating.

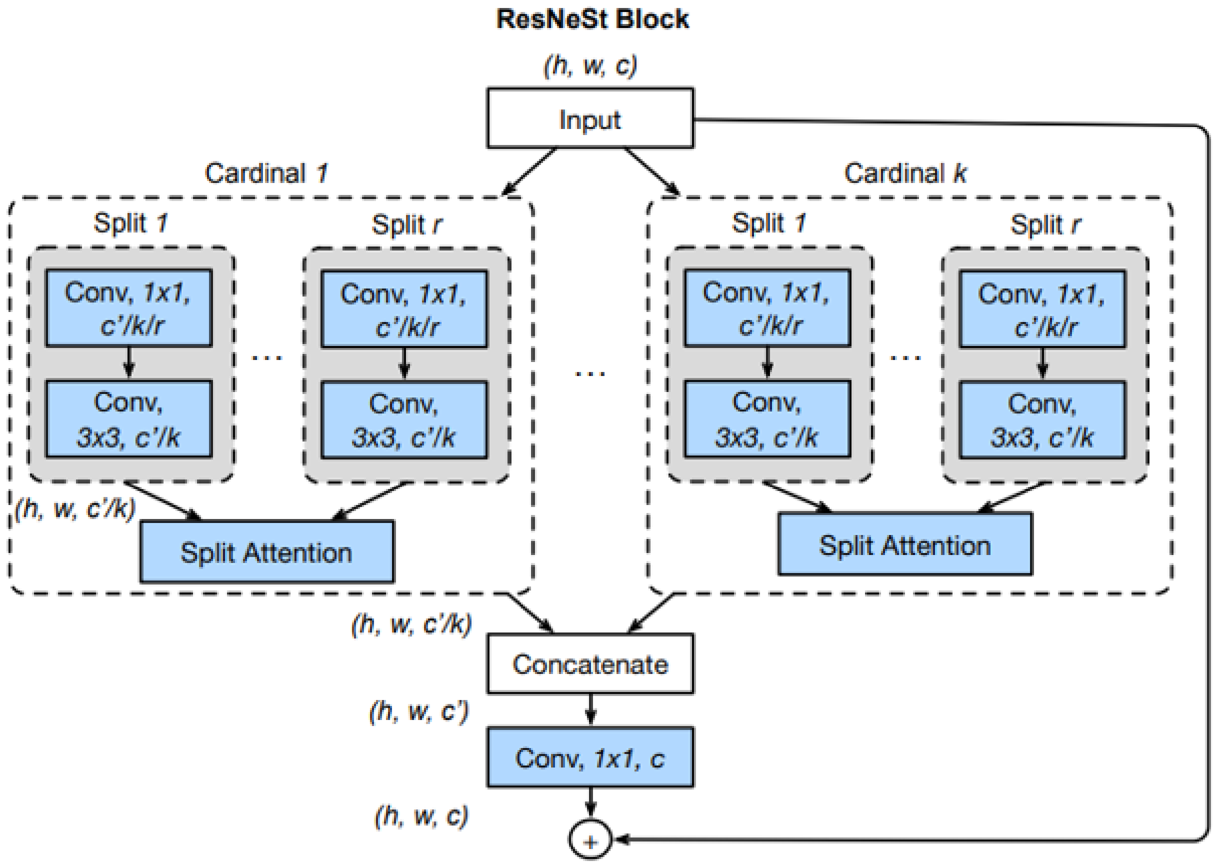


ResNeSt: Split attention Network



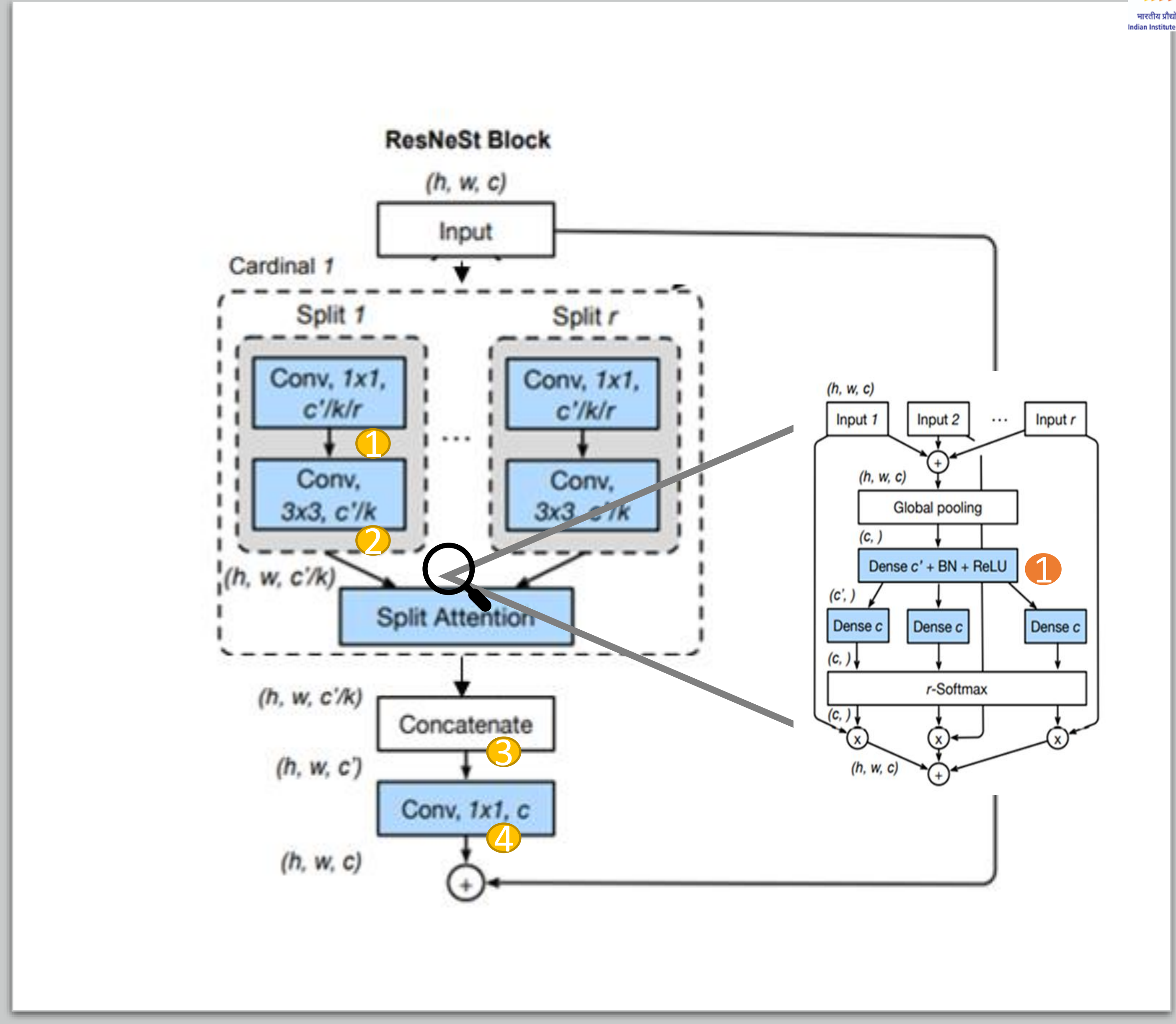
Network Detail

- Features are divided into several groups
  - Cardinality hyperparameter: K
  - Radix hyperparameter: R
  - Total number of feature groups: G = RK
- Feature-maps with the same cardinality-index but different radix index are fused together.

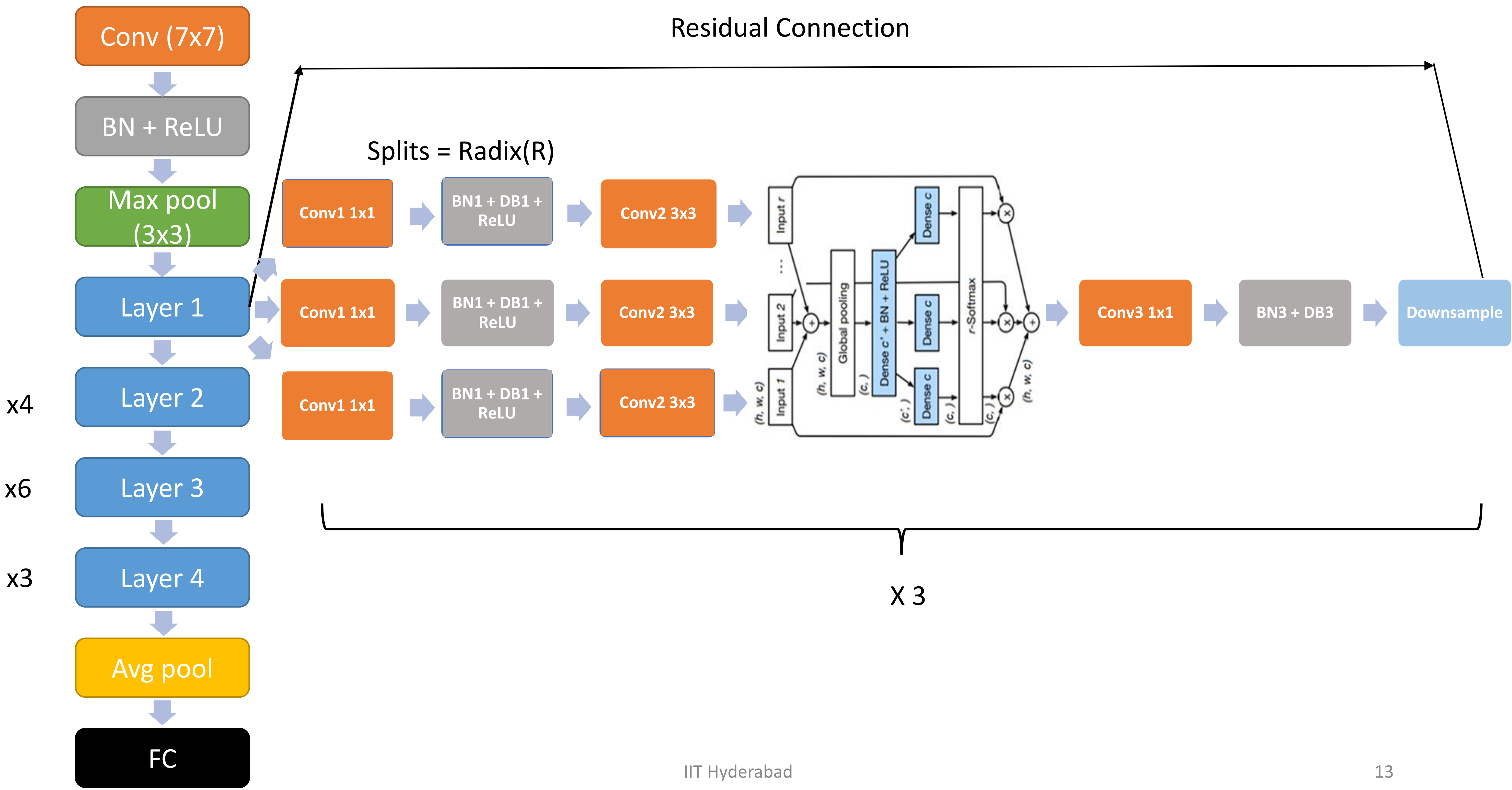


Implementation

- Hyper Parameters used here
  - Radix = 2
  - Cardinality = 1
- Position of
  - DropBlock(P=0.2, mask = 3*3).
  - DropOut (Probability (P) = 0.2)



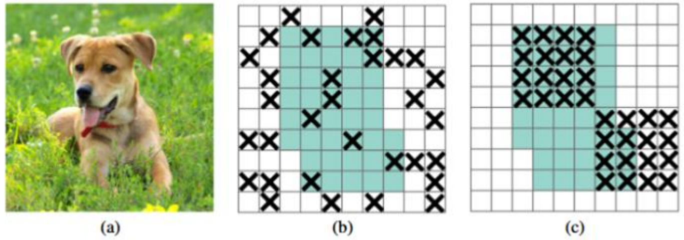
Note: In our task we only have cardinal=1. So there wont be concatenation between different cardinal groups. Hence no Drop Block 4


Conv (7x7)
BN + ReLU
Max pool (3x3)
Layer 1
x4
Layer 2
x6
Layer 3
x3
Layer 4
Avg pool
FC
Residual Connection
Splits = Radix(R)
Conv1 1x1
BN1 + DB1 + ReLU
Conv2 3x3
Conv1 1x1
BN1 + DB1 + ReLU
Conv2 3x3
Conv1 1x1
BN1 + DB1 + ReLU
Conv2 3x3
Input r
Input 2
Input 1
Global pooling
Dense c' + BN + ReLU
Dense c
Dense c
Dense c
r-Softmax
Conv3 1x1
BN3 + DB3
Downsample
X 3

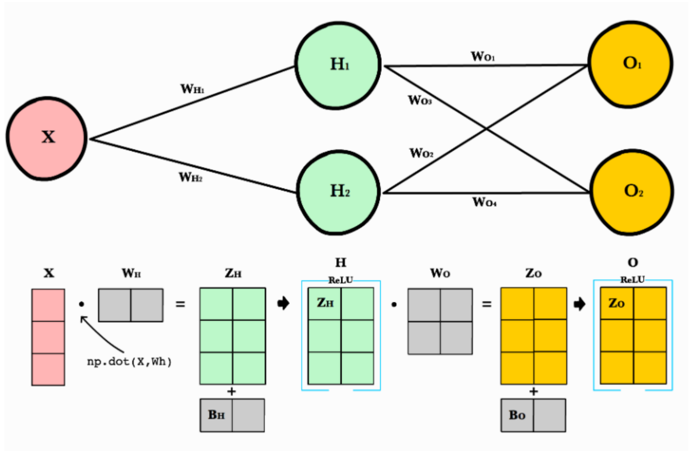
DropOut and DropBlock

These are the regularization techniques: -

- DrobBlock = 3*3 block and drop probability 0.2
- Also Applied Dropout



(a) Is an input image, the green regions in (b) and (c) include the activation units which contain semantic information in the input image.



Results

Without DropBlock, DropOut.

```
Train Epoch: 74 [49920/50000 (100%)]    Loss: 0.885207

Test set: Average loss: 0.7969, Accuracy: 7349/10000 (73%)
```

Results

With DropBlock and DropOut.

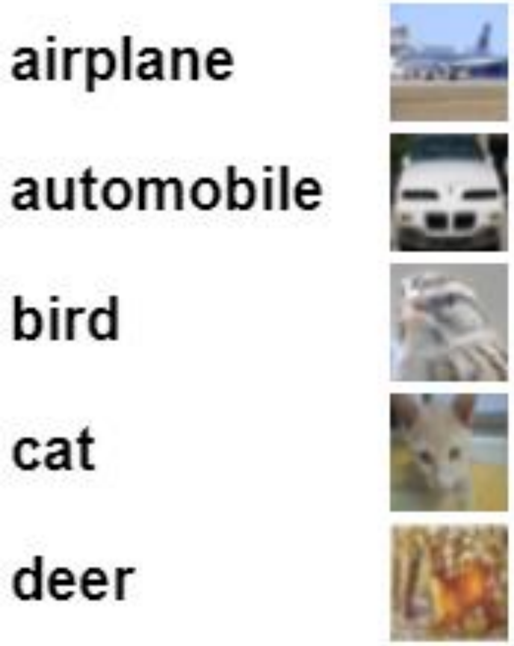
```
Train Epoch: 74 [49920/50000 (100%)]    Loss: 0.953692

Test set: Average loss: 0.7164, Accuracy: 7646/10000 (76%)
```

Results

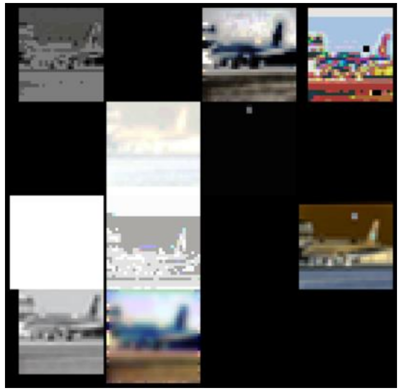
With DropBlock and DropOut And Mix-up training

```
Epoch: 74
Train Loss = 1.714710570595176, Train Accuracy = 45.999908447265625
Test loss = 0.9497059776614873, Test Accuracy = 73.86000061035156
```



Experiments

Auto-Augmentation



Translation, shear Out of frame

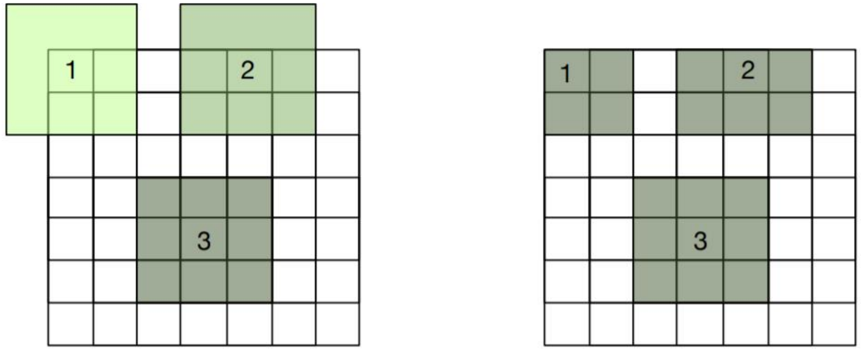
Some failure cases

dog 

bird 

Novel Ideas

- Using Rectified Convolution instead of using zero padding.



3x3 Convolution

3x3 Rectified Convolution

$$y[h,w] = \sum_{i=1}^{m}\sum_{j=1}^{n} k[i,j] \cdot \hat{x}[h-i, w-j],$$

$$y[h,w] = \frac{mn}{v[h,w]} \sum_{i=1}^{m}\sum_{j=1}^{n} k[i,j] \cdot \hat{x}[h-i, w-j],$$

- Scheduled Dropblock:

```
drop_block = LinearScheduler(
                DropBlock2D(block_size=3, drop_prob=0.),
                start_value=0.,
                stop_value=0.25,
                nr_steps=5
            )
```

Conclusions

- We observe that with lower resolution images, Data-Augmentation may not work as expected. So reoptimizing the policies defined.
- DropOut and DropBlock together were able to increase the accuracy by 3%.
- Instead of increasing the just the depth of Backbone network, one can achieve similar performance by increasing width of network.

References

- https://arxiv.org/abs/1810.12890
- https://github.com/zhanghang1989/ResNeSt/
- https://hangzhang.org/RectifiedConv/RectifiedConv.pdf
- https://paperswithcode.com/paper/resnest-split-attention-networks

Thank you