FishNet: A Versatile Backbone for Image, Region, and Pixel Level Prediction

2023.01.27

Paper Review

Introduction

- high resolution 과 deep features 에서의 pixel-level(segmentation), region-level, and image-level(classification)의 예측 모델을 위한 구조는 거의 없음
  - FishNet 은 모든 resolution 의 정보를 preserve, refine
- 기존의 방법이 gradient 정보를 깊은 layer 에서 얕은 layer로 직접적으로 전파하지 못하지만 FishNet은 해당 문제를 더 잘 처리
  - directly propagated(BP)를 통해 깊은 layer 에서 gradient 가 얕은 layer 로의 직접적 전파가 가능
- DenseNet 과 ResNet 에 비해 더 적은 파라미터로 높은 성능 달성
  - 직관과는 다르게 이미지 분류에서 전통적인 CNN 에 비해 parameter 수와 정확도 간의 trade-off 가 더 잘 수행

2. Identity Mappings in Deep Residual Networks and Isolated Convolution

- ResNet 의 경우 같은 해상도가 반복되는 구간 blocks 존재
- identity mapping 이 적용된 ResNet block
  - $\mathbf{x}_{l+1} = \mathbf{x}_l + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l).$ (3)
- 재귀적인 부분을 합치면
  - $\mathbf{x}_L = \mathbf{x}_l + \sum_{i=l}^{L-1} \mathcal{F}(\mathbf{x}_i, \mathcal{W}_i),$ (4)

- 이를 통해 깊은 x_L 을 얕은 x_l 을 이용한 잔차 형식으로 표현 가능

2. Identity Mappings in Deep Residual Networks and Isolated Convolution

- 이는 역전파 과정에서 이점을 가져옴
- $$\frac{\partial \mathcal{E}}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}\frac{\partial \mathbf{x}_L}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}\left(1 + \frac{\partial}{\partial \mathbf{x}_l}\sum_{i=l}^{L-1}\mathcal{F}(\mathbf{x}_i, \mathcal{W}_i)\right). \quad (5)$$
  - 1항 $\frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}$ 은 다른 weight 간의 연결이 없음
  - 2항 $\frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}\left(\frac{\partial}{\partial \mathbf{x}_l}\sum_{i=l}^{L-1}\mathcal{F}\right)$ 은 weight 간의 연결로 구성
- 만일 identity mapping 이 없다면
  - $$\mathbf{x}_{l+1} = \lambda_l \mathbf{x}_l + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l), \quad \mathbf{x}_L = (\prod_{i=l}^{L-1}\lambda_i)\mathbf{x}_l + \sum_{i=l}^{L-1}\hat{\mathcal{F}}(\mathbf{x}_i, \mathcal{W}_i), \quad (7)$$
  - $$\frac{\partial \mathcal{E}}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L}\left((\prod_{i=l}^{L-1}\lambda_i) + \frac{\partial}{\partial \mathbf{x}_l}\sum_{i=l}^{L-1}\hat{\mathcal{F}}(\mathbf{x}_i, \mathcal{W}_i)\right). \quad (8)$$
  - $\lambda_i > 1$ 일 경우 곱연산으로 인해 기하급수적으로 커짐
  - $\lambda_i < 1$ 일 경우 곱연산으로 인해 기하급수적으로 작아지거나 사라짐

2. Identity Mappings in Deep Residual Networks and Isolated Convolution

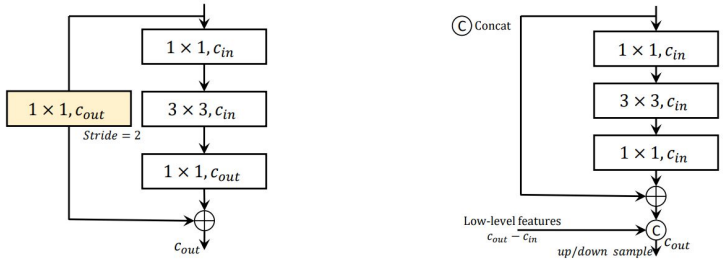


Figure 1: The up/down-sampling block for ResNet (*left*), and FishNet (*right*). The 1×1 convolution layer in yellow indicates the *Isolated convolution (I-conv, see Section 2)*, which makes the direct BP incapable and degrades the gradient from the output to shallow layers.

- Resnet 의 경우 해상도,채널 수 변경시, up/down-sampling 시 채널 변환 함수가 필요
  - 채널 수의 차이로 인해 1x1 2 stride CNN인 Isolated convolution(I-conv)적용
  - I-conv가 있으면 gradient 전송이 직접적으로 되지 않음
- 이러한 문제를 해결하기 위해 FishNet 제안

3. The FishNet

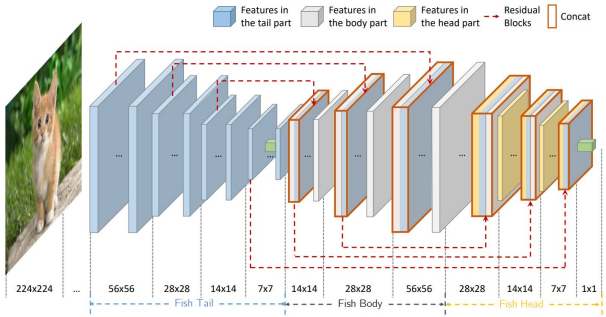
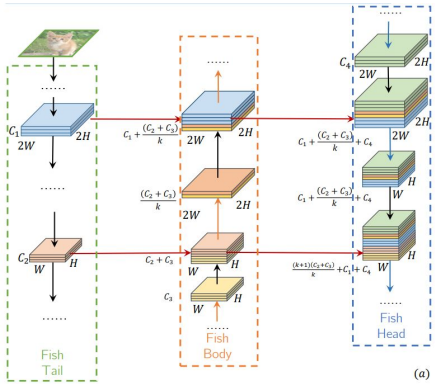


Figure 2: Overview of the FishNet. It has three parts. *Tail* uses existing works to obtain deep low-resolution features from the input image. *Body* obtains high-resolution features of high-level semantic information. *Head* preserves and refines the features from the three parts.



- FishNet 은 크게 3부위로 구분
  - Fish Tail : 기존의 CNN 기반 모델(e.g. ResNet)
  - Fish body : 꼬리와 몸통의 features 를 refine 하기 위한 up-sampling 과 refining blocks 으로 구성
  - Fish head : 꼬리, 몸통 및 머리의 features 를 preserve, refine 하기위한 down-sampling 과 refining blocks 으로 구성

3.1. Feature refinement

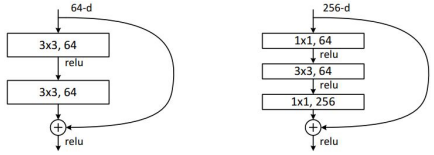
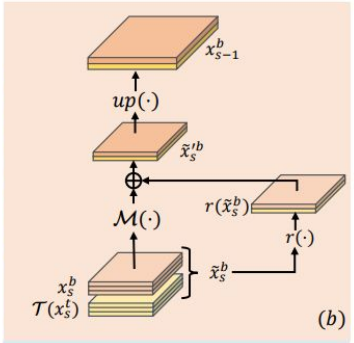
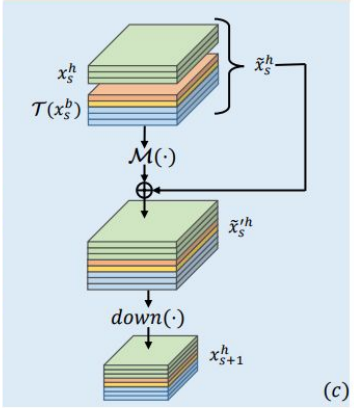


Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.





- Up-sampling & Refinement block(UR-block)
  - T 는 residual block
  - up(*) up-sampling 함수
  - M 은 resnet 의 F 와 유사
    - 3 convolutional layer 로 bottleneck Residual Unit
  - r 은 k 개의 채널을 합하여 채널 축소

$$x^b_{s-1} = up(\tilde{x}'^b_s), \tag{5}$$

$$\tilde{x}'^b_s = r(\tilde{x}^b_s) + \mathcal{M}(\tilde{x}^b_s), \tag{6}$$

$$\tilde{x}^b_s = concat(x^b_s, \mathcal{T}(x^t_s)), \tag{7}$$

$$r(x) = \hat{x} = [\hat{x}(1), \hat{x}(2), \ldots, \hat{x}(c_{out})],\ \ \hat{x}(n) = \sum_{j=0}^{k} x(k \cdot n + j), n \in \{0, 1, ..., c_{out}\}, \tag{8}$$

- Down-sampling & Refinement block(DR-block)
  - down-sampling 에 2x2 max-pooling
  - 채널 감소 함수를 사용하지 않음으로 gradient 유지

$$\begin{aligned} x^h_{s+1} &= down(\tilde{x}'^h_s), \\ \tilde{x}'^h_s &= \tilde{x}^h_s + \mathcal{M}(\tilde{x}^h_s), \\ \tilde{x}^h_s &= concat(x^h_s, \mathcal{T}(x^b_s)), \end{aligned} \tag{9}$$

3.2. Detailed design and discussion

- Design of FishNet for handling the gradient propagation problem
  - 모든 stage 의 features 가 head로 합쳐도록 I-conv를 제외하여 설계
- Selection of up/down-sampling function
  - kernel size 는 겹침 방지를 위해 2X2 stride 2 로 down-sampling
  - I-conv 를 피하기 위해 nearest neighbor interpolation 적용 up-sampling
- Bridge module between the fish body and tail
  - tail 에서의 1x1 down sample features 를 7x7 upsample.
  - 해당 과정에 SE-block 활용

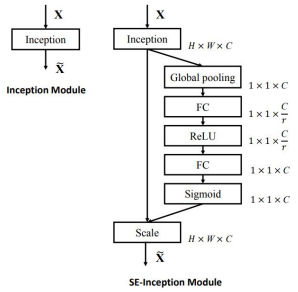


Fig. 2. The schema of the original Inception module (left) and the SE-Inception module (right).

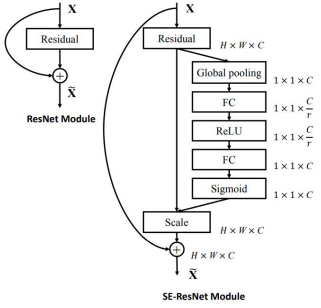


Fig. 3. The schema of the original Residual module (left) and the SE-ResNet module (right).

4.1 Implementation details on image classification

- 훈련은 ImageNet을 224 x 224 로 randomly crop 하여 batch size 256
- stochastic gradient descent (SGD)
  - learning rate 0.1
  - weight decay 10^{-4} , momentum 0.9
  - 100 epochs
  - 각 30 epochs 마다 learning late 감소
- 정규화 과정
  - 각 픽셀의 값을 [0, 1] 간격으로 변환
  - 평균을 빼고 RGB의 각 채널에 대한 분산을 각각 나누는 방식으로 진행
- augmentation (random crop, horizontal flip , standard color augmentation)
- 실험을 위해
  - FishNet : identity mapping 이 포함된 Residual block 을 기본 블록으로 사용
  - FishNeXt : identity mapping 과 grouping 이 포함된 Residual block 을 기본 블록으로 사용

4.2 Experimental results on ImageNet

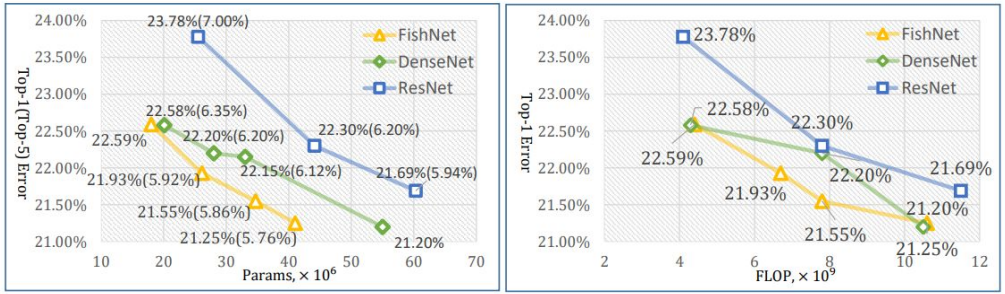


Figure 4: The comparison of the classification top-1 (top-5) error rates as a function of the number of parameters (left) and FLOP (right) for FishNet, DenseNet and ResNet (single-crop testing) on the validation set of ImageNet.

- 왼쪽 : 파라미터 수 대비 top 1(top 5) 에러
- 오른쪽 : FLOPs(FLoating point OPerations) 별 top-1 에러

Reference

- [FishNet: A Versatile Backbone for Image, Region, and Pixel Level Predictio] (https://arxiv.org/pdf/1901.03495.pdf)
  - [NIPS 2018 review comment] (https://proceedings.neurips.cc/paper/2018/file/75fc093c0ee742f6dddaa13fff98f104-Reviews.html)
  - [FishNet Medium review] (https://yckim.medium.com/%EC%A0%95%EB%A6%AC-fishnet-a-versatile-backbone-for-image-region-and-pixel-level-prediction-b86a493f114d)
- [ResNet Blog review](https://computistics.tistory.com/3)
- [Identity Mappings in Deep Residual Networks](https://arxiv.org/pdf/1603.05027.pdf)
  - [Blog Review] (https://velog.io/@kangtae/%EB%85%BC%EB%AC%B8%EB%A6%AC%EB%B7%B0-Identity-Mappings-in-Deep-Residual-Networks)
- [Squeeze-and-Excitation Networks](https://arxiv.org/abs/1709.01507)
  - [SENet review 1](https://deep-learning-study.tistory.com/539)
  - [SENet review 2](https://jayhey.github.io/deep%20learning/2018/07/18/SENet/)

Code

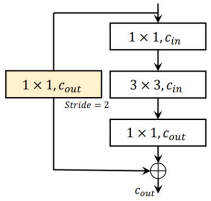
구현 전

- 기존의 224x224 이미지 사이즈의 모델을 32x32 로 변경
- ResNet18 을 기본으로 하여 진행
- ResNet bottleneck 구조
  - basic block을 Bottleneck 으로 변경
  - Identity Mappings in Deep Residual Networks 에서 제시된 방법으로 변경
    - (conv-> batch norm -> relu)*3 에서 (batch norm -> relu -> conv)*3 로 변경
- up/down - sampling
  - Tail
    - body, head의 경우 전 stage 의 데이터를 가져와 down/up sampling 을 해결했으나 tail 은 과거에 가져올 데이터가 없음
    - Resnet을 그대로 적용
  - Body, Head
    - 정확하게 어느 layer 의 정보를 넘거야 되나
    - 그림으로 유추했을때 stage 입력 직전의 데이터를 넘겨야됨

구현 중 - SENet

- SENet 의 적용을 어떻게 하나?
  - ImageNet 의 224X224 의 경우 average pooling layer전의 해상도는 7X7 이를 1X1 으로 average pooling 후 SENet으로 Upsampling
  - Cifar10의 32X32의 경우 average pooling 전에도 1x1
  - 5번의 downsampling 과정에서 최소값인 1로 축소
    - 방안 1 : 진행
      - upsampling 을 1x1 -> 1x1 으로 진행하도록 코드를 설정 차후 고해상도 이미지를 위한 대비
      - Resnet 이라는 모델을 크게 변동하지 않고 진행 가능
    - 방안 2 : layer 축소
      - Stage 을 4번 사용하는 대신 3번 사용하여 average pooling 전 2x2 로 도달하도록 설정
      - Stage 수 자체가 줄어버리는 상황 발생
  - 방안 2 선택
    - 현재의 cifar 10 데이터의 경우 데이터 크기가 작음 (약 177mb) 데이터 파라미터 수를 줄이는게 성능 향상에 도움이 되어 stage 수 축소

구현 중 - Bottleneck



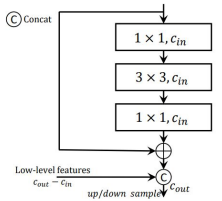


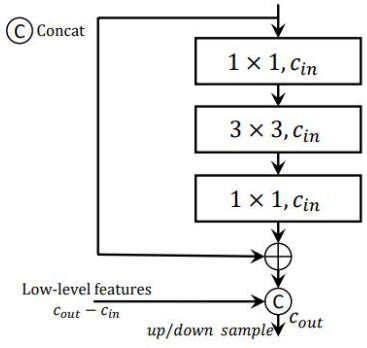
Figure 1: The up/down-sampling block for ResNet (*left*), and FishNet (*right*). The 1×1 convolution layer in yellow indicates the *Isolated convolution (I-conv, see Section 2)*, which makes the direct BP incapable and degrades the gradient from the output to shallow layers.

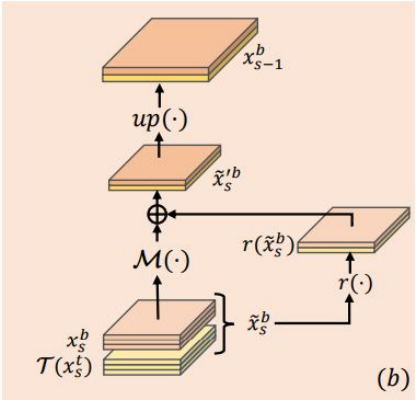
ResNet/Bottleneck	BatchNorm2d-75	[1, 2048, 1, 1]	4,096	4,096
ResNet/Bottleneck	BatchNorm2d-76	[1, 2048, 1, 1]	4,096	4,096
ResNet/Bottleneck	ReLU-77	[1, 2048, 1, 1]	0	0
ResNet/Bottleneck	Conv2d-78	[1, 2048, 1, 1]	1,048,576	1,048,576
ResNet/Bottleneck	BatchNorm2d-79	[1, 512, 1, 1]	1,024	1,024
ResNet/Bottleneck	ReLU-80	[1, 512, 1, 1]	0	0
ResNet/Bottleneck	Conv2d-81	[1, 512, 1, 1]	2,359,296	2,359,296
ResNet/Bottleneck	BatchNorm2d-82	[1, 512, 1, 1]	1,024	1,024
ResNet/Bottleneck	ReLU-83	[1, 512, 1, 1]	0	0
ResNet/Bottleneck	Conv2d-84	[1, 512, 1, 1]	1,048,576	1,048,576
ResNet	AdaptiveAvgPool2d-85	[1, 2048, 1, 1]	0	0
ResNet	Linear-86	[1, 2048]	20,490	20,490

FishNet/Bottleneck	BatchNorm2d-53	[1, 256, 2, 2]	512	512
FishNet/Bottleneck	BatchNorm2d-54	[1, 256, 2, 2]	512	512
FishNet/Bottleneck	ReLU-55	[1, 256, 2, 2]	0	0
FishNet/Bottleneck	Conv2d-56	[1, 256, 2, 2]	65,536	65,536
FishNet/Bottleneck	BatchNorm2d-57	[1, 256, 2, 2]	512	512
FishNet/Bottleneck	ReLU-58	[1, 256, 2, 2]	0	0
FishNet/Bottleneck	Conv2d-59	[1, 256, 2, 2]	589,824	589,824
FishNet/Bottleneck	BatchNorm2d-60	[1, 256, 2, 2]	512	512
FishNet/Bottleneck	ReLU-61	[1, 256, 2, 2]	0	0
FishNet/Bottleneck	Conv2d-62	[1, 256, 2, 2]	65,536	65,536
FishNet	AdaptiveAvgPool2d-63	[1, 256, 2, 2]	0	0
FishNet	Linear-64	[1, 256]	2,570	2,570

- 논문 Figure 1의 경우 모든 channel 가 동일
  - Bottleneck 구조의 Expansion 을 기본 4에서 1로 수정
- 좌 기존 ResNet
  - Bottleneck, expansion, 4 stage
- 우 FishNet 변형
  - Bottleneck, unexpansion , 3 stage

구현 중 - Bottleneck





- 논문 Figure 1 의 경우 모든 layer 의 channel 수가 일정
- 반대로 논문 Figure 3(b) 의 경우 M 과정에서 차원 축소가 필요
- 마지막 1X1, C_{in}에서 channel 축소 하도록 설정

Conclusion

- Test set 을 이용 성능 비교
  - 논문의 Top-1 error 형식
- FishNet 보다 Resnet 성능이 더 높음
  - 충분하지 못한 epoch?
    - epoch를 200으로 재 훈련 결과 거의 동일한 성능
  - Bottleneck 구현 문제?
    - bottleneck 구조의 경우 축소 후 확장이 중요 사항 이를 적용 시키지 못함
  - over fitting, early stop 문제?
  - 너무 높은 learning late(0.1)
    - 0.1 learning late 로 인해 loss 값이 overflow 되는 현상 발생
    - 임의로 0.01 로 축소 했으나 이로 인한 성능 하락 가능성
  - 위와 같은 가능성을 고려하여 수정 필요
- 그 외
  - 1X1 에서 7X7 로 확장할때만 SENet 을 쓴 이유?
    - 적은 데이터에 upsampling 과정의 손해를 막기 위해서
    - 모든 과정에 SENet 적용해도 좋지 않았나
  - parameter 수 관리를 위해서로 예상

Model	Top-1 Error (%)	Params × 10^6	Params
cnn_100	69	0.06	62006
resnet_100	22	13	13962698
resnet_200	22	13	13962698
fishnet_100	27	7	7998666
fishnet_200	26	7	7998666