9장. 인셉션 모델과 레스넷 모델 : 꽃 이미지 분류 신경망

부산대학교 항공우주공학과

정대현 201527137

1. 개요

이번 실험에서는 주요 합성곱 신경망 모델 가운데 '거대 구조'의 대표적 사례라고 할 수 있는 인셉션 모델과 '깊은 구조'의 대표적 사례라고 할 수 있는 레스넷 모델을 학습하게 됩니다. 다양한 구조의 모델 지원에 필요한 다섯 가지 새로운 복합계층, 커널 크기와 보폭에 제한이 없는 확정된 합성곱 계층과 풀링 계층, 마지막으로 인셉션 모델과 레스넷 모델 구조 생성 및 간소화된 모델의 학습을 다루게 됩니다.

2. 보틀넥 모듈의 특징과 장단점

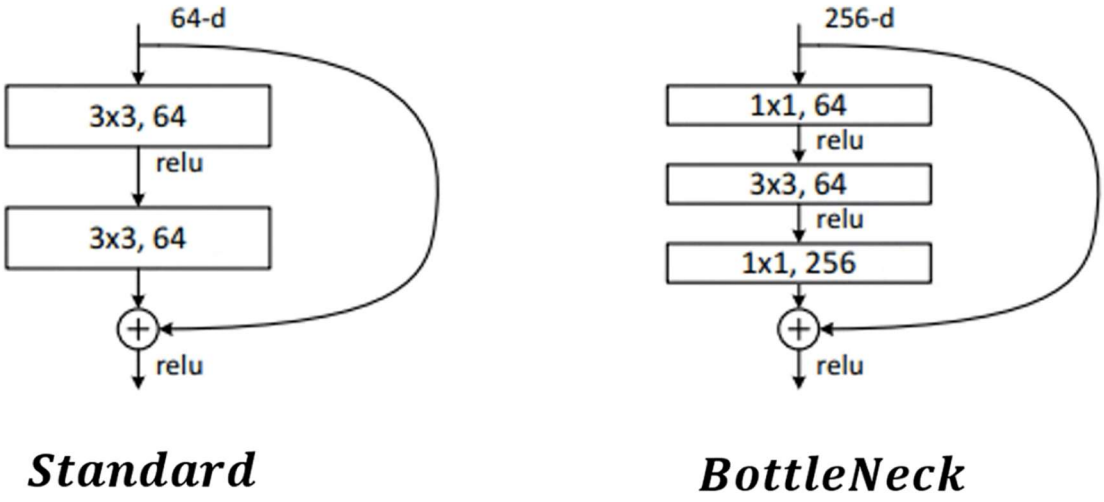

Fig. 1. 보틀넥(BottleNeck)적용 모델

Fig. 1. 그림은 왼쪽의 레지듀얼 블록을 오른쪽 그림처럼 변형시키면 연산량을 줄이면서도 입출력 채널 개수를 크게 늘려 더욱 효과적인 처리가 가능합니다. 이런 점에 착안하여 보틀넥 블록을 통해 매우 깊은 구조의 레스넷 모델들을 만들 수 있습니다. 단점은 input과 output의 dimension을 통일해야 하는 제한이 있습니다.

3. 인셉션 모델

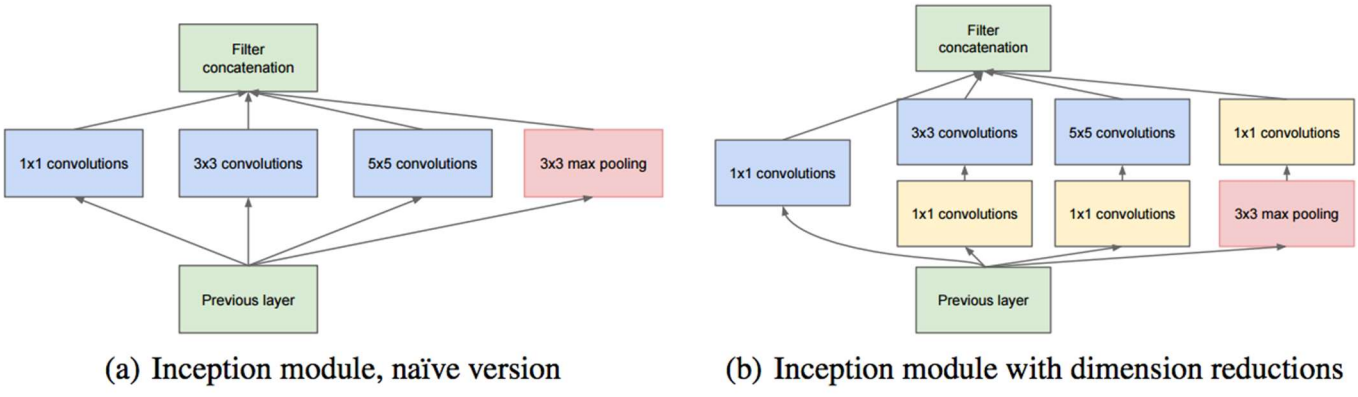

Fig. 2. (a) 기본형 인셉션 모듈, (b) 개선된 인셉션 모듈

인셉션 모델은 인셉션 모듈이라고 하는 병렬 처리 합성곱 신경망 구조를 반복적으로 활용하여 신경망의 규모를 크게 늘린 모델입니다. 인셉션 모델은 다양한 병렬 구조를 이용하며 인셉션 모듈이라는 중간 구성 단위를 둔다. 인셉션 모듈은 하나의 입력을 몇몇 분기에서 병렬처리 후 결과값을 모아 다음 단계로 전달합니다. (a)의 기본형 인셉션 모듈은 합성곱 풀링 계층과 각기 다른 커널 크기가 섞여 있습니다. 이는 한가지 방법에 의존하기 보다 여러 방법으로 분석하고 이를 종합하는 편이 좋은 결과를 가져올 것이라는 믿음에 근거한 방법입니다.

(b)는 개선된 인셉션 모듈로 네 개의 분기를 가지며 각 분기에서 수행하는 일도 비슷합니다. 여기서 추가하는 1x1 합성곱은 픽셀별로 입력 채널 전체와 출력 채널 전체를 연결하는 구조로 각 픽셀 위치에 대해 출력 채널별로 입력 채널 정보를 종합한 내용을 생성합니다.

4. 레스넷 모델

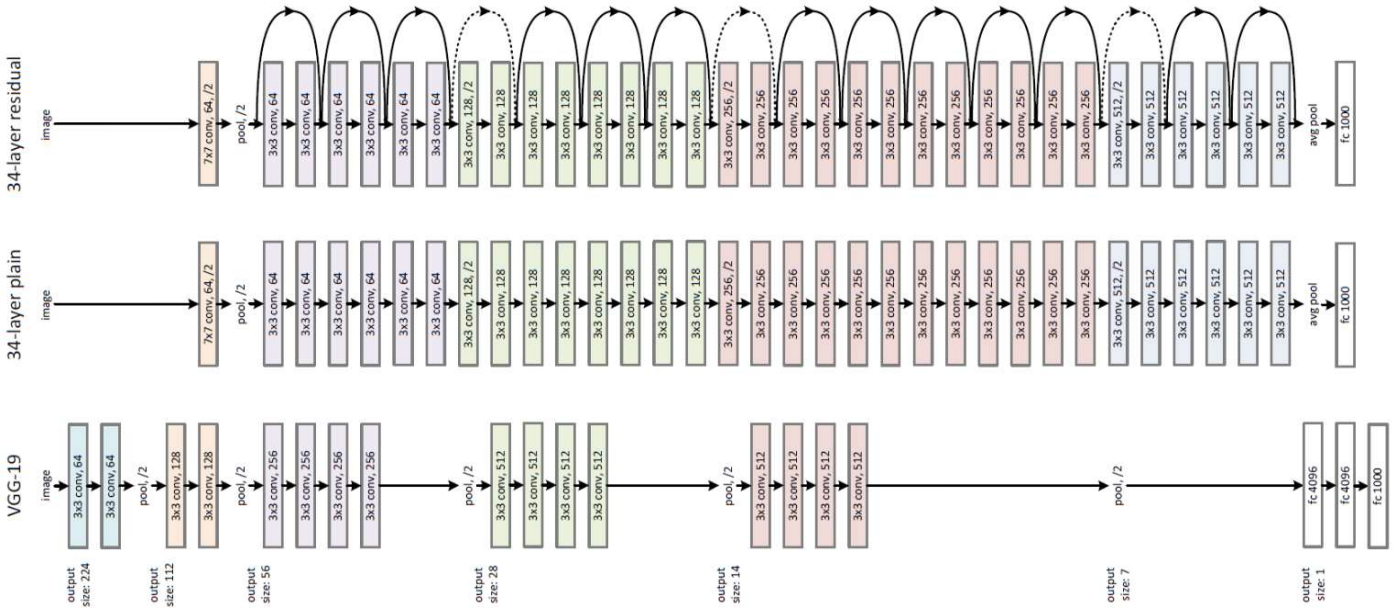

Fig. 3. 34-layer ResNet with Skip / Shortcut Connection (Top), 34-layer Plain Network (Middle), 19-layer VGG-19 (Bottom)

레스넷(ResNet) 모델은 여러 딥러닝 모델 가운데에서도 깊은 구조의 신경망입니다. 매우 많은 수의 레지듀얼 블록을 차례대로 연결해서 만듭니다. 여기서 레지듀얼 블록은 합성곱 계층의 처리 결과에 원래의 입력을 더하는 방법을 이용해 각 합성곱 계층의 학습이 미분 수준의 작은 변화에 집중되도록 만들어주는 병렬처리 구조입니다.

다시 정리하자면 인셉션 모델은 병렬 구조가, 레스넷 모델은 신경망 처리 결과와 입력의 합산이 특징입니다.

5. 실험

5.1 인셉션-꽃 모델 선형_비선형

```
Model model_flower_LA train started:
    Epoch 2: cost=1.605, accuracy=0.240/0.200 (547/547 secs)
    Epoch 4: cost=1.601, accuracy=0.244/0.230 (566/1113 secs)
    Epoch 6: cost=1.601, accuracy=0.244/0.270 (575/1688 secs)
    Epoch 8: cost=1.604, accuracy=0.239/0.220 (576/2264 secs)
    Epoch 10: cost=1.725, accuracy=0.238/0.250 (587/2851 secs)
Model model_flower_LA train ended in 2851 secs:
Model model_flower_LA test report: accuracy = 0.301, (21 secs)
Model model_flower_LA Visualization
```

```
추정확률분포 [15,22,19,19,24] => 추정 tulip : 정답 tulip => O
추정확률분포 [16,24,19,18,23] => 추정 dandelion : 정답 dandelion => O
추정확률분포 [16,24,18,18,23] => 추정 dandelion : 정답 daisy => X
```

5.2 인셉션-꽃 모델 선형_비선형_배치정규화

```
Model model_flower_LAB train started:
    Epoch 2: cost=1.459, accuracy=0.339/0.220 (664/664 secs)
    Epoch 4: cost=1.495, accuracy=0.326/0.250 (606/1270 secs)
    Epoch 6: cost=1.465, accuracy=0.337/0.300 (594/1864 secs)
    Epoch 8: cost=1.482, accuracy=0.329/0.240 (594/2458 secs)
    Epoch 10: cost=1.420, accuracy=0.372/0.120 (608/3066 secs)
Model model_flower_LAB train ended in 3066 secs:
Model model_flower_LAB test report: accuracy = 0.245, (21 secs)
추정확률분포 [ 0, 2,25,13,59] => 추정 tulip : 정답 sunflower => X
추정확률분포 [ 1,25, 3,57,14] => 추정 sunflower : 정답 sunflower => O
추정확률분포 [ 1, 2,44, 3,50] => 추정 tulip : 정답 daisy => X
```

5.3 레지듀얼 입력이 없는 플레인-꽃 모델 학습

```
custom plain_flower
  serial
    1: conv, (64, 64, 3)=>[32, 32, 16] pm:7x7x3x16+16=2368
    2: max, [32, 32, 16]=>[16, 16, 16]
    loop
      3: conv, [16, 16, 16]=>[16, 16, 16] pm:3x3x16x16+16=2320
      4: conv, [16, 16, 16]=>[16, 16, 16] pm:3x3x16x16+16=2320
      5: conv, [16, 16, 16]=>[16, 16, 16] pm:3x3x16x16+16=2320
      6: conv, [16, 16, 16]=>[16, 16, 16] pm:3x3x16x16+16=2320
    custom pn
      serial
        7: conv, [16, 16, 16]=>[8, 8, 32] pm:3x3x16x32+32=4640
        loop
          8: conv, [8, 8, 32]=>[8, 8, 32] pm:3x3x32x32+32=9248
          9: conv, [8, 8, 32]=>[8, 8, 32] pm:3x3x32x32+32=9248
          10: conv, [8, 8, 32]=>[8, 8, 32] pm:3x3x32x32+32=9248
    custom pn
      serial
        11: conv, [8, 8, 32]=>[4, 4, 64] pm:3x3x32x64+64=18496
        loop
          12: conv, [4, 4, 64]=>[4, 4, 64] pm:3x3x64x64+64=36928
          13: conv, [4, 4, 64]=>[4, 4, 64] pm:3x3x64x64+64=36928
          14: conv, [4, 4, 64]=>[4, 4, 64] pm:3x3x64x64+64=36928
    15: avg, [4, 4, 64]=>[1, 1, 64]
16: full, [1, 1, 64]=>[5] pm:64x5+5=325
Total parameter count: 173637
Model plain_flower train started:
```

```
    Epoch 2: cost=1.496, accuracy=0.330/0.210 (123/123 secs)
    Epoch 4: cost=1.323, accuracy=0.410/0.230 (128/251 secs)
    Epoch 6: cost=1.301, accuracy=0.416/0.330 (130/381 secs)
    Epoch 8: cost=1.297, accuracy=0.426/0.530 (126/507 secs)
    Epoch 10: cost=1.269, accuracy=0.455/0.450 (128/635 secs)
Model plain_flower train ended in 635 secs:
Model plain_flower test report: accuracy = 0.357, (4 secs)
추정확률분포 [12,15, 8,58, 6] => 추정 sunflower : 정답 sunflower => O
추정확률분포 [ 0, 0, 0, 0,100] => 추정 tulip : 정답 tulip => O
추정확률분포 [ 0, 0, 6, 1,94] => 추정 tulip : 정답 sunflower => X
```

5.4 레지듀얼 입력을 갖는 레지듀얼-꽃 모델 학습

```
custom residual_flower
  serial
    1: conv, (64, 64, 3)=>[32, 32, 16] pm:7x7x3x16+16=2368
    2: max, [32, 32, 16]=>[16, 16, 16]
    custom rfull
      serial
        loop
          custom rf
            add
              serial
                3: conv, [16, 16, 16]=>[16, 16, 16] pm:3x3x16x16+16=2320
                4: conv, [16, 16, 16]=>[16, 16, 16] pm:3x3x16x16+16=2320
          custom rf
            add
              serial
                5: conv, [16, 16, 16]=>[16, 16, 16] pm:3x3x16x16+16=2320
                6: conv, [16, 16, 16]=>[16, 16, 16] pm:3x3x16x16+16=2320
    custom rhalf
      serial
        custom rh
          add
            serial
              7: conv, [16, 16, 16]=>[8, 8, 32] pm:3x3x16x32+32=4640
              8: conv, [8, 8, 32]=>[8, 8, 32] pm:3x3x32x32+32=9248
            9: avg, [16, 16, 16]=>[8, 8, 16]
        loop
          custom rf
            add
              serial
                10: conv, [8, 8, 32]=>[8, 8, 32] pm:3x3x32x32+32=9248
                11: conv, [8, 8, 32]=>[8, 8, 32] pm:3x3x32x32+32=9248
    custom rhalf
```

```
        serial
          custom rh
            add
              serial
                12: conv, [8, 8, 32]=>[4, 4, 64] pm:3x3x32x64+64=18496
                13: conv, [4, 4, 64]=>[4, 4, 64] pm:3x3x64x64+64=36928
              14: avg, [8, 8, 32]=>[4, 4, 32]
          loop
            custom rf
              add
                serial
                  15: conv, [4, 4, 64]=>[4, 4, 64] pm:3x3x64x64+64=36928
                  16: conv, [4, 4, 64]=>[4, 4, 64] pm:3x3x64x64+64=36928
    17: avg, [4, 4, 64]=>[1, 1, 64]
18: full, [1, 1, 64]=>[5] pm:64x5+5=325
Total parameter count: 173637
Model residual_flower train started:
    Epoch 2: cost=1.252, accuracy=0.481/0.260 (133/133 secs)
    Epoch 4: cost=1.158, accuracy=0.534/0.320 (136/269 secs)
    Epoch 6: cost=1.049, accuracy=0.583/0.270 (136/405 secs)
    Epoch 8: cost=0.986, accuracy=0.599/0.390 (139/544 secs)
    Epoch 10: cost=0.940, accuracy=0.637/0.600 (145/689 secs)
Model residual_flower train ended in 689 secs:
Model residual_flower test report: accuracy = 0.603, (4 secs)
추정확률분포 [93, 4, 2, 0, 1] => 추정 daisy : 정답 daisy => O
추정확률분포 [23,59, 3, 3,12] => 추정 dandelion : 정답 daisy => X
추정확률분포 [ 0,41, 0,59, 0] => 추정 sunflower : 정답 sunflower => O
```

5.5 보틀넥 블록을 이용하는 보틀넥-꽃 모델 학습

```
custom bottleneck_flower
  serial
    1: conv, (64, 64, 3)=>[32, 32, 16] pm:7x7x3x16+16=2368
    2: max, [32, 32, 16]=>[16, 16, 16]
    custom bfull
      serial
        loop
          custom bf
            add
              serial
                3: conv, [16, 16, 16]=>[16, 16, 16] pm:1x1x16x16+16=272
                4: conv, [16, 16, 16]=>[16, 16, 16] pm:3x3x16x16+16=2320
                5: conv, [16, 16, 16]=>[16, 16, 64] pm:1x1x16x64+64=1088
    custom bhalf
      serial
```

```
            custom bh
              add
                serial
                  6: conv, [16, 16, 64]=>[8, 8, 32] pm:1x1x64x32+32=2080
                  7: conv, [8, 8, 32]=>[8, 8, 32] pm:3x3x32x32+32=9248
                  8: conv, [8, 8, 32]=>[8, 8, 128] pm:1x1x32x128+128=4224
                9: avg, [16, 16, 64]=>[8, 8, 64]
            loop
              custom bf
                add
                  serial
                    10: conv, [8, 8, 128]=>[8, 8, 32] pm:1x1x128x32+32=4128
                    11: conv, [8, 8, 32]=>[8, 8, 32] pm:3x3x32x32+32=9248
                    12: conv, [8, 8, 32]=>[8, 8, 128] pm:1x1x32x128+128=4224
              custom bf
                add
                  serial
                    13: conv, [8, 8, 128]=>[8, 8, 32] pm:1x1x128x32+32=4128
                    14: conv, [8, 8, 32]=>[8, 8, 32] pm:3x3x32x32+32=9248
                    15: conv, [8, 8, 32]=>[8, 8, 128] pm:1x1x32x128+128=4224
      custom bhalf
        serial
          custom bh
            add
              serial
                16: conv, [8, 8, 128]=>[4, 4, 64] pm:1x1x128x64+64=8256
                17: conv, [4, 4, 64]=>[4, 4, 64] pm:3x3x64x64+64=36928
                18: conv, [4, 4, 64]=>[4, 4, 256] pm:1x1x64x256+256=16640
              19: avg, [8, 8, 128]=>[4, 4, 128]
          loop
            custom bf
              add
                serial
                  20: conv, [4, 4, 256]=>[4, 4, 64] pm:1x1x256x64+64=16448
                  21: conv, [4, 4, 64]=>[4, 4, 64] pm:3x3x64x64+64=36928
                  22: conv, [4, 4, 64]=>[4, 4, 256] pm:1x1x64x256+256=16640
      23: avg, [4, 4, 256]=>[1, 1, 256]
24: full, [1, 1, 256]=>[5] pm:256x5+5=1285
Total parameter count: 189925
Model bottleneck_flower train started:
      Epoch 2: cost=1.263, accuracy=0.486/0.220 (147/147 secs)
      Epoch 4: cost=1.089, accuracy=0.575/0.470 (158/305 secs)
      Epoch 6: cost=1.010, accuracy=0.608/0.470 (162/467 secs)
      Epoch 8: cost=0.956, accuracy=0.633/0.370 (159/626 secs)
```

```
    Epoch 10: cost=0.888, accuracy=0.662/0.410 (165/791 secs)
Model bottleneck_flower train ended in 791 secs:
Model bottleneck_flower test report: accuracy = 0.431, (5 secs)
추정확률분포 [12,85, 0, 0, 3] => 추정 dandelion : 정답 daisy => X
추정확률분포 [ 0, 5, 0, 0,95] => 추정 tulip : 정답 tulip => O
추정확률분포 [ 0, 1,32, 0,67] => 추정 tulip : 정답 rose => X
```

모델	정확도
인셉션-꽃 모델 선형_비선형	0.301
인셉션-꽃 모델 선형_비선형_배치정규화	0.245
레지듀얼 입력이 없는 플레인-꽃 모델	0.357
레지듀얼 입력을 갖는 레지듀얼-꽃 모델	0.603
보틀넥 블록을 이용하는 보틀넥-꽃 모델	0.431

6. 결론

첫번째 인셉션-꽃 모델 LA는 커널을 이용한 합성곱 연산을수행한후 비선형활성화 함수, 즉 ReLU 함수를 적용하는 처리입니다. 실행 결과는 0.301로 오랜 시간 동안 학습하였지만 정확도는 많이 낮은 수준이며 추정확률분포를 살펴보면 모두 동일한 것을 알 수 있습니다. 이는 평가 데이터의 크기에 따라서 결정된 것을 볼 수 있으며, 학습이 크게 부족한 상태라고 할 수 있습니다. 더해서 cost의 변화는 미미하기에 학습을 더한다고 해도 나은 결과를 얻을 수 있다는 확신은 없습니다. 그 다음으로는 LAB로 배치 정규화를 추가한 모델입니다. 정확도는 0.245로 오히려 감소하였지만 추정확률분포는 데이터의 크기가 아니라 학습 결과로 변화한 것을 알 수 있습니다. 정확도는 떨어졌지만 오히려 첫번째 모델보다 학습이 이루어졌다고 볼 수 있으며 배치 정규화는 데이터의 불균형을 어느정도 잡을 수 있는 것을 확인할 수 있습니다.

플레인-꽃 모델은 플레인-34 모델 구성에 사용된 매크로 모듈들을 이용해 정의됩니다. 13개의 합성곱 계층과 완전 연결 계층인 출력 계층 등 파라미터를 갖는 계층 14개와 풀링 계층 2개 등 모두 16개 계층으로 구성되며 약 17만개의 파라미터를 갖습니다. 정확도는 35,8%로 이전 모델에 비해서는 미치지 못하지만 학습 진행에 따라서 cost가 줄어드는 것을 볼 수 있으며 학습량이 많아진다면 나은 결과를 얻을 수 있을 것으로 봅니다. 이어 레지듀얼-꽃 모델의 경우 정확도는 0.603이 얻어졌습니다. 교재에 나온 정확도는 0.463이었던 것을 감안하면 학습의 편차가 큰 것을 관찰할 수 있습니다. 하지만 플레인-꽃 모델 보다는 정확도를 10%이상 올렸다는 점에서 인상적입니다. 보틀넥-꽃 모델은 정확도를 0.431을 기록했습니다. 레지듀얼-꽃 모델보다 낮게 나타나기는 했지만 cost 값이 줄어드는 것을 살펴보면 마찬가지로 학습 시간을 늘리거나 학습데이터를 더 준다면 개선될 여지가 있습니다. 전체적으로 정확도만 놓고 보았을 때는 인상적인 결과를 얻지 못했습니다. 이는 개인 컴퓨터에서 학습하는 시간과 학습 데이터 양이 영향을 끼친 것으로 보이며 더 많은 데이터와 학습량을 준다면 더 나은 정확도를 얻을 수 있을 것입니다.

지금까지 기존 합성곱 계층과 풀링 계층의 기능을 사용하는 데 제한이 없는 형태로 확장하고 새로운 다섯 가지 복합 계층을 추가해 인셉션 모델이나 레스넷 모델 또는 그 변형을 구성할 수 있게 했습니다.