인공지능 챕터 9 과제

202055518 김병현

인셉션 모델과 레스넷 모델 : 꽃 이미지 분류 신경망

cnn_inception_test

인셉션 모델에 사용할 각종 모듈들을 매크로로 정의
- v3_preprop, v3_inception1, v3_resize1, v3_inception2,
- v3_resize2, v3_inception3, v3_postproc

인셉션-**v3** 모델 매크로로 정의
- 인셉션-v3 전체 구조를 매크로로 정의

```
CnnExtModel.set_macro('inception_v3',
    ['serial',
        ['custom', {'name':'v3_preproc'}],
        ['custom', {'name':'v3_inception1', 'args':{'#chn':32}}],
        ['custom', {'name':'v3_inception1', 'args':{'#chn':64}}],
        ['custom', {'name':'v3_inception1', 'args':{'#chn':64}}],
        ['custom', {'name':'v3_resize1'}],
        ['custom', {'name':'v3_inception2', 'args':{'#chn':128}}],
        ['custom', {'name':'v3_inception2', 'args':{'#chn':160}}],
        ['custom', {'name':'v3_inception2', 'args':{'#chn':160}}],
        ['custom', {'name':'v3_inception2', 'args':{'#chn':192}}],
        ['custom', {'name':'v3_resize2'}],
        ['custom', {'name':'v3_inception3'}],
        ['custom', {'name':'v3_inception3'}],
        ['custom', {'name':'v3_postproc'}]])
```

인셉션-**v3** 모델 객체 생성
- 인셉션-v3 매크로 정보를 사용자 정의 계층으로 지정해 모듈 생성
- 구조 출력 결과를 통해 인셉션-v3 모델의 구조 및 규모 확인
- 110계층, 32,401,768개의 파라미터

```
inception_v3 = CnnExtModel('inception_v3', imagenet,
            [['custom', {'name':'inception_v3'}]], dump_structure=True)
110: full, [1, 1, 2048]=>[200] pm:2048x200+200=409800
Total parameter count: 32401768
```

꽃-인셉션 모델에 사용할 각종 모듈들을 매크로로 정의
- flower_preprop, flower_inception1, flower_resize,
- flower_inception2, flower_postproc

꽃-인셉션 모델 매크로로 정의

- 앞의 모듈 구조를 이용해 꽃-인셉션 전체 구조를 매크로로 정의

```
CnnExtModel.set_macro('inception_flower',
    ['serial',
        ['custom', {'name':'flower_preproc', 'args':{'#act':'#act'}}],
        ['custom', {'name':'flower_inception1', 'args':{'#act':'#act'}}],
        ['custom', {'name':'flower_resize', 'args':{'#act':'#act'}}],
        ['custom', {'name':'flower_inception1', 'args':{'#act':'#act'}}],
        ['custom', {'name':'flower_resize', 'args':{'#act':'#act'}}],
        ['custom', {'name':'flower_inception2', 'args':{'#act':'#act'}}],
        ['custom', {'name':'flower_resize', 'args':{'#act':'#act'}}],
        ['custom', {'name':'flower_inception2', 'args':{'#act':'#act'}}],
        ['custom', {'name':'flower_postproc', 'args':{'#act':'#act'}}]])
```

꽃-인셉션**-LA** 모델 객체 생성

- actions 값을 'LA'(원래의 처리 방식에 해당)로 지정해 객체 생성

```
conf_flower_LA = ['custom', {'name':'inception_flower', 'args':{'#act':'LA'}}]
model_flower_LA = CnnExtModel('model_flower_LA', fd,
                              conf_flower_LA, dump_structure=True)
```

```
48: full, [1, 1, 48]=>[5] pm:48x5+5=245
Total parameter count: 43885
```

꽃-인셉션**-LA** 모델 학습

- 정확도 24.4%, 일률적 확률분포: 학습이 제대로 되지 않음 확인

```
model_flower_LA.exec_all(report=2)
```

```
Model model_flower_LA train started:
    Epoch 2: cost=1.602, accuracy=0.241/0.200 (664/664 secs)
    Epoch 4: cost=2.450, accuracy=0.223/0.230 (696/1360 secs)
    Epoch 6: cost=1.630, accuracy=0.236/0.270 (655/2015 secs)
    Epoch 8: cost=1.609, accuracy=0.236/0.310 (662/2677 secs)
    Epoch 10: cost=1.607, accuracy=0.233/0.280 (640/3317 secs)
Model model_flower_LA train ended in 3317 secs:
Model model_flower_LA test report: accuracy = 0.244, (33 secs)
```

꽃-인셉션**-LAB** 모델 학습시키기

- actions 값을 'LAB'로 바꾸어 각 합성곱 계층의 배치정규화 기능 가동
- 정확도 34.2%, 미비하나마 학습이 시작되고 있음을 확인

```
conf_flower_LAB = ['custom', {'name':'inception_flower', 'args':{'#act':'LAB'}}]
model_flower_LAB = CnnExtModel('model_flower_LAB', fd,
                               conf_flower_LAB, dump_structure=False)
model_flower_LAB.exec_all(epoch_count=10, report=2)
```

```
Model model_flower_LAB train ended in 2987 secs:
Model model_flower_LAB test report: accuracy = 0.342, (50 secs)
```

cnn_resnet_test

VGG-19 모델 객체 만들고 구조 확인

- p24, vgg_19
- 24계층, 143,667,240개의 파라미터

```
CnnExtModel.set_macro('p24',
    ['serial',
        ['loop', {'repeat':'#repeat'}, ['conv', {'ksize':3, 'chn':'#chn'}]],
        ['max', {'stride':2}]])

CnnExtModel.set_macro('vgg_19',
    ['serial',
        ['custom', {'name':'p24', 'args':{'#repeat':2, '#chn':64}}],
        ['custom', {'name':'p24', 'args':{'#repeat':2, '#chn':128}}],
        ['custom', {'name':'p24', 'args':{'#repeat':4, '#chn':256}}],
        ['custom', {'name':'p24', 'args':{'#repeat':4, '#chn':512}}],
        ['custom', {'name':'p24', 'args':{'#repeat':4, '#chn':512}}],
        ['loop', {'repeat':2}, ['full', {'width':4096}]]])

vgg19 = CnnExtModel('vgg_19', imagenet,
        ['custom', {'name':'vgg_19'}], dump_structure=True)
```

```
24: full, [4096]=>[1000] pm:4096x1000+1000=4097000
Total parameter count: 143667240
```

플레인**-34** 모델 객체 만들고 구조 확인하기

- pn, plain_34
- 36계층, 21,616,232개의 파라미터

```
CnnExtModel.set_macro('pn',
    ['serial',
        ['conv', {'ksize':3, 'stride':2, 'chn':'#n', 'actions':'#act'}],
        ['loop', {'repeat':'#cnt1'},
                ['conv', {'ksize':3, 'chn':'#n', 'actions':'#act'}]]])

CnnExtModel.set_macro('plain_34',
    ['serial',
        ['conv', {'ksize':7, 'stride':2, 'chn':64, 'actions':'#act'}],
        ['max', {'stride':2}],
        ['loop', {'repeat':6}, ['conv', {'ksize':3, 'chn':64, 'actions':'#act'}]]
        ['custom', {'name':'pn', 'args':{'#cnt1':7, '#n':128, '#act':'#act'}}],
        ['custom', {'name':'pn', 'args':{'#cnt1':11, '#n':256, '#act':'#act'}}],
        ['custom', {'name':'pn', 'args':{'#cnt1':5, '#n':512, '#act':'#act'}}],
        ['avg', {'stride':7}]])

plain_34 = CnnExtModel('plain_34', imagenet,
    ['custom', {'name':'plain_34', 'args':{'#act':'LA'}}], dump_structure=True
```

```
36: full, [1, 1, 512]=>[1000] pm:512x1000+1000=513000
Total parameter count: 21616232
```

레지듀얼-34 모델 객체 만들고 구조 확인하기

- rf, rh, rfull, rhalf, residual_34
- 39계층, 21,616,232개의 파라미터
- 레지듀얼 입력을 사용하는 점 외에는 플레인-34 모델과 같은 구조
  ※ 해상도 줄이는 부분의 레지듀얼 입력 처리에 avg 계층 사용
  * 겉보기 계층 수가 셋 늘었지만 사실은 같은 구조

```
CnnExtModel.set_macro('rf',
    ['add', {'x':True},
        ['serial', ['conv', {'ksize':3, 'chn':'#n', 'actions':'#act'}],
                   ['conv', {'ksize':3, 'chn':'#n', 'actions':'#act'}]]])

CnnExtModel.set_macro('rh',
    ['add', {'x':False},
        ['serial', ['conv', {'ksize':3, 'stride':2, 'chn':'#n', 'actions':'#act'}
                   ['conv', {'ksize':3, 'chn':'#n', 'actions':'#act'}]],
        ['avg', {'stride':2}]])

CnnExtModel.set_macro('rfull',
    ['serial',
        ['loop', {'repeat':'#cnt'},
                 ['custom', {'name':'rf', 'args':{'#n':'#n', '#act':'#act'}}]]])

CnnExtModel.set_macro('rhalf',
    ['serial',
        ['custom', {'name':'rh', 'args':{'#n':'#n', '#act':'#act'}}],
        ['loop', {'repeat':'#cnt1'},
                 ['custom', {'name':'rf', 'args':{'#n':'#n', '#act':'#act'}}]]])

CnnExtModel.set_macro('residual_34',
    ['serial',
        ['conv', {'ksize':7, 'stride':2, 'chn':64, 'actions':'#act'}],
        ['max', {'stride':2}],
        ['custom', {'name':'rfull', 'args':{'#cnt':3, '#n':64, '#act':'#act'}}],
        ['custom', {'name':'rhalf', 'args':{'#cnt1':3, '#n':128, '#act':'#act'}}]
        ['custom', {'name':'rhalf', 'args':{'#cnt1':5, '#n':256, '#act':'#act'}}]
        ['custom', {'name':'rhalf', 'args':{'#cnt1':2, '#n':512, '#act':'#act'}}]
        ['avg', {'stride':7}]])

residual_34 = CnnExtModel('residual_34', imagenet,
    ['custom', {'name':'residual_34', 'args':{'#act':'LA'}}], dump_structure=True
```

```
39: full, [1, 1, 512]=>[1000] pm:512x1000+1000=513000
Total parameter count: 21616232
```

보틀넥-152 모델 객체 만들고 구조 확인하기

- bf, bh, bfull, bhalf, bottleneck_152
- 157계층, 57,344,360개의 파라미터

```
CnnExtModel.set_macro('bf',
    ['add', {'x':True},
        ['serial',
            ['conv', {'ksize':1, 'chn':'#n1', 'actions':'#act'}],
            ['conv', {'ksize':3, 'chn':'#n1', 'actions':'#act'}],
            ['conv', {'ksize':1, 'chn':'#n4', 'actions':'#act'}]]])

CnnExtModel.set_macro('bh',
    ['add', {'x':False},
        ['serial',
            ['conv', {'ksize':1, 'stride':2, 'chn':'#n1', 'actions':'#act'}],
            ['conv', {'ksize':3, 'chn':'#n1', 'actions':'#act'}],
            ['conv', {'ksize':1, 'chn':'#n4', 'actions':'#act'}]],
        ['avg', {'stride':2}]])

CnnExtModel.set_macro('bfull',
    ['serial',
        ['loop', {'repeat':'#cnt'},
            ['custom', {'name':'bf', 'args':{'#n1':'#n1', '#n4':'#n4',
                '#act':'#act'}}]]])

CnnExtModel.set_macro('bhalf',
    ['serial',
        ['custom', {'name':'bh', 'args':{'#n1':'#n1', '#n4':'#n4',
            '#act':'#act'}}],
        ['loop', {'repeat':'#cnt1'},
            ['custom', {'name':'bf', 'args':{'#n1':'#n1', '#n4':'#n4',
                '#act':'#act'}}]]])
```

```
CnnExtModel.set_macro('bottleneck_152',
    ['serial',
        ['conv', {'ksize':7, 'stride':2, 'chn':64, 'actions':'#act'}],
        ['max', {'ksize':3, 'stride':2}],
        ['custom', {'name':'bfull','args':{'#cnt':3,'#n1':64,'#n4':256,'#act':'#act'}}],
        ['custom', {'name':'bhalf','args':{'#cnt1':7,'#n1':128,'#n4':512,
            '#act':'#act'}}],
        ['custom', {'name':'bhalf','args':{'#cnt1':35,'#n1':256,'#n4':1024,
            '#act':'#act'}}],
        ['custom', {'name':'bhalf','args':{'#cnt1':2,'#n1':512,'#n4':2048,
            '#act':'#act'}}],
        ['avg', {'stride':7}]])

bottleneck_152 = CnnExtModel('bottleneck_152', imagenet,
        ['custom', {'name':'bottleneck_152', 'args':{'#act':'LAB'}}],
                dump_structure=True)
```

```
157: full, [1, 1, 2048]=>[1000] pm:2048x1000+1000=2049000
Total parameter count: 57344360
```

플레인-꽃 모델 학습시키기
  - actions 키값은 LAB로 지정
  - 정확도 44.2%, 미비하나마 학습이 시작되고 있음을 확인

```
plain_flower = CnnExtModel('plain_flower', fd,
        ['custom', {'name':'plain_flower', 'args':{'#act':'LAB'}}],
                    dump_structure=True)
plain_flower.exec_all(epoch_count=10, report=2)
```

```
Model plain_flower train ended in 1133 secs:
Model plain_flower test report: accuracy = 0.442, (8 secs)
```

레지듀얼-꽃 모델 학습시키기
  - 정확도 55.4%, 레지듀얼 입력 효과 확인

```
residual_flower = CnnExtModel('residual_flower', fd,
      ['custom', {'name':'residual_flower', 'args':{'#act':'LAB'}}],
                  dump_structure=True)
residual_flower.exec_all(epoch_count=10, report=2)
```

```
Model residual_flower train ended in 1187 secs:
Model residual_flower test report: accuracy = 0.554, (6 secs)
```

보틀넥-꽃 모델 학습시키기
  - 정확도 51.8%, 추가 학습으로 성능 제고 가능성 큼 확인

```
bottleneck_flower = CnnExtModel('bottleneck_flower', fd,
    ['custom', {'name':'bottleneck_flower', 'args':{'#act':'LAB'}}],
        dump_structure=True)
bottleneck_flower.exec_all(epoch_count=10, report=2)
```

```
Model bottleneck_flower train ended in 1354 secs:
Model bottleneck_flower test report: accuracy = 0.518, (8 secs)
```