

9장. 인셉션 모델과 레스넷 모델 : 꽃 이미지 분류 신경망


1. 개요

이 실험에서는 인셉션 모델과 레스넷 모델을 지원하는 합성곱 신경망 모델을 만들어 거대 심층 구조를 형성해 꽃 이미지 분류 신경망에 적용해보는 작업을 수행하였다. 인셉션 모델이란 병렬 처리 합성곱 신경망 구조인 인셉션 모듈을 반복 활용해 신경망의 규모를 크게 늘린 모델을 의미한다. 또, 레스넷 모델은 매우 많은 수의 레지듀얼 블록을 차례로 직렬 연결한, 딥러닝 모델 중에서도 특히 깊은 구조의 신경망 모델을 뜻한다. 이 실험에서는 레지듀얼-꽃 모델에 대해 자세히 살펴보고, 인셉션 모듈과 레스넷 모델의 비교 및 보틀넥 모듈의 특징 및 장단점에 대해 더 생각해보고자 한다.

2. 실험 결과


2.1 인셉션 모델 적용

(1) 모델 1

구분	내용
데이터 정확도	30.6%
학습 시간	2860s
결과 화면	<pre>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)</pre> <p>Model model_flower_LAB train started: Epoch 2: cost=1.529, accuracy=0.308/0.240 (570/570 secs) Epoch 4: cost=1.492, accuracy=0.313/0.230 (572/1142 secs) Epoch 6: cost=1.552, accuracy=0.294/0.250 (569/1711 secs) Epoch 8: cost=1.540, accuracy=0.306/0.230 (571/2282 secs) Epoch 10: cost=1.512, accuracy=0.330/0.290 (578/2860 secs) Model model_flower_LAB train ended in 2860 secs: Model model_flower_LAB test report: accuracy = 0.306, (22 secs)</p> <p>Model model_flower_LAB Visualization</p>  <p>추정 확률분포 [4,19,12,33,32] => 추정 sunflower : 정답 tulip => X 추정 확률분포 [33,39,18, 2, 8] => 추정 dandelion : 정답 daisy => X 추정 확률분포 [22,31,16,15,17] => 추정 dandelion : 정답 dandelion => 0</p>


2.2 레스넷 모델 적용

(1) 모델 2

구분	내용
조건	<ul style="list-style-type: none"> - 구조 : 플레인-34 - 모델 : plain_flower - 에포크 횟수 : 10
데이터 정확도	35.8%
학습 시간	1218s
결과 화면	<pre> CnnExtModel.set_macro('plain_flower', ['serial', ['conv', {'ksize':7, 'stride':2, 'chn':16, 'actions': '#act'}], ['max', {'stride':2}], ['loop', {'repeat':4}, ['conv', {'ksize':3, 'chn':16, 'actions': '#act'}]], ['custom', {'name': 'pn', 'args': {'#cnt1':3, '#n':32, '#act': '#act'}}], ['custom', {'name': 'pn', 'args': {'#cnt1':3, '#n':64, '#act': '#act'}}], ['avg', {'stride':4}]]]) 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) Total parameter count: 173637 Model plain_flower train started: Epoch 2: cost=1.375, accuracy=0.380/0.360 (243/243 secs) Epoch 4: cost=1.393, accuracy=0.390/0.260 (241/484 secs) Epoch 6: cost=1.325, accuracy=0.406/0.360 (245/729 secs) Epoch 8: cost=1.311, accuracy=0.415/0.350 (246/975 secs) Epoch 10: cost=1.305, accuracy=0.410/0.440 (243/1218 secs) Model plain_flower train ended in 1218 secs: Model plain_flower test report: accuracy = 0.358, (7 secs) Model plain_flower Visualization </pre> <div style="display: flex; justify-content: space-around; align-items: center;">  </div> <p> 추정확률분포 [4,96, 0, 0, 0] => 추정 dandelion : 정답 rose => X 추정확률분포 [26,33,17, 9,15] => 추정 dandelion : 정답 tulip => X 추정확률분포 [0, 0,72, 0,28] => 추정 rose : 정답 tulip => X </p>


(2) 모델 3

구분	내용
조건	<ul style="list-style-type: none"> - 구조 : 레지듀얼-34 - 모델 : residual flower - 에포크 횟수 : 10
데이터 정확도	46.3%
학습 시간	1216s

결과 화면	<pre> CnnExtModel.set_macro('residual_flower', ['serial', ['conv', {'ksize':7, 'stride':2, 'chn':16, 'actions': '#act'}], ['max', {'ksize':3, 'stride':2}], ['custom', {'name': 'rfull', 'args': {'#cnt':2, '#n':16, '#act': '#act'}}], ['custom', {'name': 'rhalf', 'args': {'#cnt':1, '#n':32, '#act': '#act'}}], ['custom', {'name': 'rhalf', 'args': {'#cnt':1, '#n':64, '#act': '#act'}}], ['avg', {'stride':4}]] 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) </pre> <p>Total parameter count: 173637 Model residual_flower train started: Epoch 2: cost=1.257, accuracy=0.473/0.250 (239/239 secs) Epoch 4: cost=1.174, accuracy=0.521/0.230 (244/483 secs) Epoch 6: cost=1.103, accuracy=0.554/0.420 (241/724 secs) Epoch 8: cost=1.031, accuracy=0.590/0.490 (245/969 secs) Epoch 10: cost=0.986, accuracy=0.608/0.500 (247/1216 secs) Model residual_flower train ended in 1216 secs: Model residual_flower test report: accuracy = 0.463, (6 secs)</p> <p>Model residual_flower Visualization</p>  <p>추정확률분포 [3,59, 7,21,10] => 추정 dandelion : 정답 dandelion => 0 추정확률분포 [0, 0,30, 0,69] => 추정 tulip : 정답 rose => X 추정확률분포 [0,83, 1, 9, 8] => 추정 dandelion : 정답 sunflower => X</p>
-------	--

(3) 모델 4

구분	내용
조건	<ul style="list-style-type: none"> - 구조 : 보틀넥-152 - 모델 : bottleneck_flower - 에포크 횟수 : 10
데이터 정확도	42.0%
학습 시간	1403s
결과 화면	<pre> CnnExtModel.set_macro('bottleneck_flower', ['serial', ['conv', {'ksize':7, 'stride':2, 'chn':16, 'actions': '#act'}], ['max', {'ksize':3, 'stride':2}], ['custom', {'name': 'bfull', 'args': {'#cnt':1, '#n1':16, '#n4': 64, '#act': '#act'}}], ['custom', {'name': 'bhalf', 'args': {'#cnt1':2, '#n1':32, '#n4':128 '#act': '#act'}}], ['custom', {'name': 'bhalf', 'args': {'#cnt1':1, '#n1':64, '#n4':256, '#act': '#act'}}], ['avg', {'stride':4}]] 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) </pre>

	<p>Total parameter count: 189925 Model bottleneck_flow train started: Epoch 2: cost=1.221, accuracy=0.502/0.300 (279/279 secs) Epoch 4: cost=1.080, accuracy=0.578/0.550 (281/560 secs) Epoch 6: cost=0.985, accuracy=0.609/0.430 (278/838 secs) Epoch 8: cost=0.956, accuracy=0.634/0.350 (280/1118 secs) Epoch 10: cost=0.874, accuracy=0.664/0.410 (285/1403 secs) Model bottleneck_flow train ended in 1403 secs: Model bottleneck_flow test report: accuracy = 0.420, (8 secs)</p> <p>Model bottleneck_flow Visualization</p>  <p>추정확률분포 [1,96, 2, 1, 1] => 추정 dandelion : 정답 dandelion => 0 추정확률분포 [0,98, 0, 2, 0] => 추정 dandelion : 정답 sunflower => X 추정확률분포 [1,21,24, 9,44] => 추정 tulip : 정답 rose => X</p>
--	---

3. 결과 분석

실험 내용을 표로 정리하면 다음과 같다.

모델	인셉션-v3	플레인-34	레지듀얼-34	보틀넥-152
정확도(%)	30.6	35.8	46.3	42.0
소요 시간(s)	2860	1218	1216	1403

표를 통해 볼 수 있듯이, 인셉션 모델을 사용했을 때보다 레스넷 모델을 사용했을 때, 전반적으로 정확도가 더 높게 나타났다. 또한 소요시간도 레스넷 모델이 거의 2배가량 낮게 나타났다는데, 이는 입력과 출력을 더하는 스킵 연결로 각 계층의 출력을 그대로 가져오며, 이것을 합산하는 방식으로 학습 과정을 단순화 했기 때문임을 유추해볼 수 있다. 플레인-34, 레지듀얼-34, 보틀넥-152를 이용한 실험 모두 괜찮은 결과를 얻었기에, 이를 통해 레스넷 모델이 다양한 과제에 원활히 작동한다는 것을 확인할 수 있었다.

3.1 추가 분석

(1) 보틀넥 모듈의 특징 및 장단점

보틀넥 구조는 딥러닝 모델에서 메모리와 연산 비용을 줄이는 기법 중 하나로 주로 레스넷과 같은 심층 신경망 아키텍처에서 사용된다.

- 특징 : 레지듀얼 블록의 변형으로 볼 수 있으며, 연산량을 줄이면서도 상대적으로 입출력

채널 수를 크게 늘리기 때문에 효과적인 처리가 가능하다. 전후의 1x1 합성곱 계층을 이용해 채널 수를 축소 및 복원할 수 있고, 이로 인해 합성곱 연산에 필요한 곱셈 횟수 역시 같은 비율로 감소하게 된다. 즉, 작은 커널 크기와 적은 출력 채널 수를 사용하여 중간 레이어의 특징 맵의 크기를 줄이는 것이다.

- 장점 : 더 적은 수의 파라미터와 더 적은 양의 연산 수행으로 효율이 높아진다. 하지만 외부적으로는 오히려 정보의 채널을 늘려 제공하는 것과 같은 효과가 있다. 즉, 효율적인 연산을 통해 계산 비용을 낮출 수 있을 뿐 아니라, 더 높은 성능과 정확도를 얻을 수 있다.

- 단점 : 보틀넥 블록을 거칠 때마다 영향을 미치는 입력 영역의 폭이 넓어지는데, 출력 픽셀에 반영되는 입력 영역의 크기가 감소한다. 따라서 작은 커널과 적은 채널 수가 모델의 표현 능력을 감소시킬 수도 있다. 하지만 깊은 구조의 레스넷 모델에서는 이러한 단점이 문제가 되지는 않는다.

(2) 인셉션 모델과 레스넷 모델의 비교

- 인셉션 모델 : 병렬 구조로, 서로 다른 convolution 필터 크기와 차원의 조합을 이용해 정보를 추출하는데 중점을 둔다. 이는 인셉션 모듈이라 불리는 병렬 구조로 구성이 된다. 또, 입력 데이터를 여러 가지 크기의 필터로 동시에 처리해 다양한 특징을 추출하고 이들을 합친다.

- 레스넷 모델 : 레지듀얼 연결을 사용해 신경망의 깊이를 증가시키면서도 gradient 소실 문제를 완화한다. 입력과 출력을 더하는 스킵 연결로 각 계층의 출력을 그대로 가져오며, 이것을 합산하는 방식으로 학습 과정을 단순화할 수 있다. 스킵 연결은 각 레이어의 출력을 이전의 층에 더해주는 방식을 말한다.

특징	인셉션	레스넷
구조	병렬 구조 (인셉션 모듈)	직렬 깊은 구조 (레스넷 모듈)
주요 특징	다양한 컨볼루션 필터 크기 및 차원 결합	스킵 연결로 그레이디언트 소실 완화
네트워크 깊이	비교적 깊은 네트워크	매우 깊은 네트워크
그레이디언트 소실 문제	주목하지 않음	스킵 연결로 완화
성능	이미지 인식 및 분류에 효과적	다양한 과제에 원활히 작동
구조 유연성	다양한 크기의 필터 조합 가능	다양한 깊이 및 너비 조절 가능

위의 표는 인셉션 모델과 레스넷 모델의 비교 내용을 간단히 표로 정리한 내용이다.

4. 결론

인셉션 모델보다 레스넷 모델에서 정확도가 높고, 소요시간이 적게 나타났다. 이는 입력과 출력을 더하는 스킵 연결로 각 계층의 출력을 그대로 가져오며, 이것을 합산하는 방식으로 학습 과정을 단순화 했기 때문임을 유추해볼 수 있다. 또한 실험을 통해 인셉션 모델은 다양한 크기의 필터를 조합할 수 있으나 레스넷 모델은 다양한 깊이 및 너비가 조절이 가능하다는 점도 알 수 있었다.