파라미터 저장하고 불러오기
& 사전 학습된 모델 불러오기

오늘의 학습내용

- 파라미터 저장하고 불러오기
- Colab 실습을 통한 파라미터 저장하고 불러오기
- 사전 학습된 CNN 모델 사용하기
- 전이 학습(Transfer Learning)
- Colab 실습을 통한 사전 학습된 CNN 모델 사용하기

파라미터와 모델 저장하고 불러오기

- 학습이 끝난 모델의 파라미터를 저장하고 불러오는 방법을 살펴봅니다.
- 파이토치에서는 보통 PyTorcH model의 약자로 .pth 확장자로 모델 파라미터를 저장합니다.
- 모델의 파라미터만 저장하고 불러오는 방법
- 파라미터 저장하기

```
model = models.vgg16(pretrained=True)
torch.save(model.state_dict(), 'model_weights.pth')
```

- 파라미터 불러오기

```
model = models.vgg16(pretrained=True)
model.load_state_dict(torch.load('model_weights.pth'))
```

파라미터와 모델 저장하고 불러오기

- 모델과 파라미터를 함께 저장하고 불러오기
- 모델과 파라미터 함께 저장하기

```
torch.save(model, 'model.pth')
```

- 모델과 파라미터 함께 불러오기

```
model = torch.load('model.pth')
```

파이토치(PyTorch)에서 모델과 파라미터 저장하고 불러오기

- 파이토치(PyTorch)를 이용해서 모델과 파라미터를 저장하고 불러오는 방법을 실습을 통해 살펴봅시다.
- 5강_PyTorch를_이용해서_파라미터_저장하고_불러오기.ipynb
- https://colab.research.google.com/drive/1bSIBWy5LrdZuosUGMh2F-pT5L4exzxZQ?usp=sharing

대표적인 CNN 모델들 - AlexNet, VGGNet, GoogLenet, ResNet

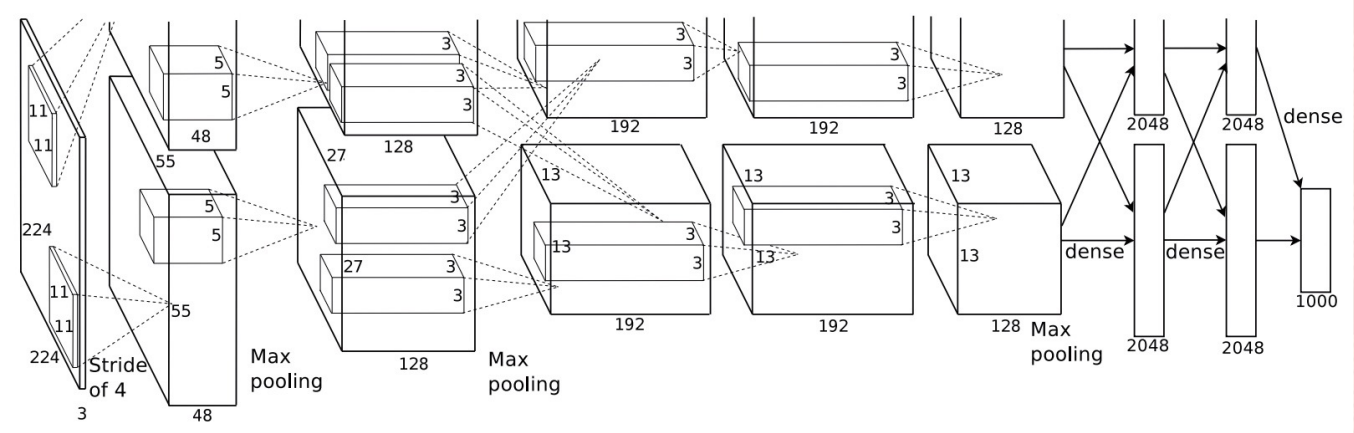
- 풀고자 하는 문제가 있을 때, 해당 문제를 풀기에 가장 적절한 CNN 구조를 디자인하는 것은 많은 시행착오가 필요한 일입니다. 예를 들어서, CNN의 층(레이어)을 몇 개로 할 것인지, 각 층의 노드 개수는 몇 개로 할 것인지, 컨볼루션의 필터 사이즈를 어느 크기로 할 것인지 등 수많은 변수들을 모두 고려해야만 합니다.
- 하지만 다행히도 연구자들이 많은 시행착오 끝에 특정 문제에 적절한 CNN 구조를 발견하고 논문으로 발표하였습니다. 이는 일종의 참고서로써 많은 사람이 CNN 구조를 디자인할 때 이 구조들을 차용하고 있습니다.
- 그렇다면 어떤 문제에 어떤 CNN 구조가 적절한지 어떻게 알 수 있을까요? 가장 직관적인 답변은 테스트 데이터셋에 대한 정확도가 가장 높은 CNN 구조가 가장 좋은 구조일 것입니다. 또한 해당 정확도가 다양한 상황에서 잘 동작한 것임을 증명하기 위해서 다양한 환경으로 구성된 대량의 테스트 데이터셋이 필요합니다.
- 이런 목적을 위해서 머신러닝 연구자들은 ILSVRC라는 대회를 매년 개최하고 있습니다. ILSVRC는 ImageNet Large Scale Visual Recognition Challenge의 약자로 이미지 분류 경진 대회입니다. 구체적으로 1000개의 레이블을 가진 대량의 이미지를 분류해서 가장 높은 정확도를 내는 팀이 우승하는 형태입니다.
- 앞으로 알아볼 모델들은 매년 열리는 ILSVRC 대회에서 1등이나 2등을 차지하며 유명해진 CNN 구조들입니다. 이제 각각의 모델들의 구조와 특징을 자세히 살펴봅시다.

AlexNet

- Alexnet은 2012년 ILSVRC 대회에서 우승을 하며 유명해진 CNN 모델입니다. Alexnet이란 이름은 Alexnet 구조를 제안한 연구자인 Alex Krizhevsky의 이름에서 따왔습니다. 특히, Alexnet은 딥러닝 붐을 일으키는 시발점이 된 모델이라는 점에서 큰 의미가 있습니다.
- 앞서 설명했듯이 매년 많은 연구자들이 ILSVRC 대회에 참가해서 이미지 분류 문제에 대해서 자신의 모델의 얼마나 좋은지를 경쟁합니다.
- 최초의 ILSVRC 대회는 2010년에 열렸습니다. 2010년도 우승 모델의 에러율은 28.2%였고, 이어진 2011년 대회의 우승 모델의 에러율은 25.8%였습니다.
- 2010, 2011년 우승 모델은 딥러닝을 사용하지 않은 모델이었습니다. 하지만 2012년도에 우승을 차지한 Alexnet은 16.4%의 에러율로 기존 우승 모델과 큰 격차를 보여주며 우승을 차지하게 됩니다. 딥러닝을 사용한 Alexnet이 압도적인 격차를 보이며 우승을 차지하자, 많은 연구자들이 딥러닝 모델의 성능을 인정하고 2013년도부터는 대부분의 대회 참가자들이 딥러닝 모델을 사용하게 됩니다.

AlexNet

- 이제 구체적인 Alexnet의 구조를 살펴봅시다. 그림은 Alexnet의 구조를 보여줍니다.
- Alexnet은 2개의 GPU를 이용해서 학습을 진행했습니다. 따라서 그림의 위쪽 부분은 1번째 GPU에서 수행하는 연산, 아래쪽 부분은 2번째 GPU에서 수행하는 연산을 의미합니다. Alexnet은 5개의 컨볼루션층과 3개의 완전 연결층을 합쳐서 총 8개의 층으로 구성됩니다.



AlexNet

❖ 각각의 층(레이어) 이름과 파라미터, 출력 결과를 표 형태로 나타내면 표 7-9와 같습니다.

층(레이어) 이름	파라미터	출력 결과
Input Layer	없음	227×227×3
CONV1	커널 크기 : 11×11 스트라이드 : 4 제로패딩 : 0 커널 개수 : 96	55×55×96
MAX POOL1	커널 크기 : 3×3 스트라이드 : 2	27×27×96
CONV2	커널 크기 : 3×3 스트라이드 : 2 제로패딩 : 2 커널 개수 : 256	27×27×256
MAX POOL2	커널 크기 : 3x3 스트라이드 : 2	13×13×256
CONV3	커널 크기 : 3×3 스트라이드 : 1 제로패딩 : 1 커널 개수 : 384	13×13×384
CONV4	커널 크기 : 3×3 스트라이드 : 1 제로패딩 : 1 커널 개수 : 384	13×13×384
CONV5	커널 크기 : 3×3 스트라이드 : 1 제로패딩 : 1 커널 개수 : 256	13×13×256
MAX POOL3	커널 크기 : 3×3 스트라이드 : 2	6×6×256
FC6	노드 개수 : 4096	4096
FC7	노드 개수 : 4096	4096
FC8	노드 개수 : 1000	4096

VGGNet

- VGGNet은 2014년 ILSVRC 대회에서 준우승을 차지한 모델입니다. VGGNet이란 이름은 모델을 창안한 연구자들이 속한 Oxford 대학의 Visual Geometry Group에서 따왔습니다.

- VGGNet 연구자들은 그림과 같이 6개의 CNN 구조로 실험을 진행해서 각각의 모델들의 성능을 비교했습니다. 그림의 행들을 해석하면 conv3-64는 64개의 3×3 컨볼루션 필터를 적용한다는 것을 의미합니다. 마찬가지로 conv1-512는 512개의 1×1 컨볼루션 필터를 적용한다는 것을 의미합니다.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

VGGNet

❖ VGGNet 구조를 보면 Alexnet 구조와 몇 가지 다른 점을 찾을 수 있습니다.

1) AlexNet의 11×11 컨볼루션 필터를 제거하고, 상대적으로 작은 크기의 3x3 컨볼루션 필터와 1×1 컨볼루션 필터만을 사용하고 있습니다.

2) 층의 깊이가 8에서 11, 13, 16, 19까지 늘어났습니다.

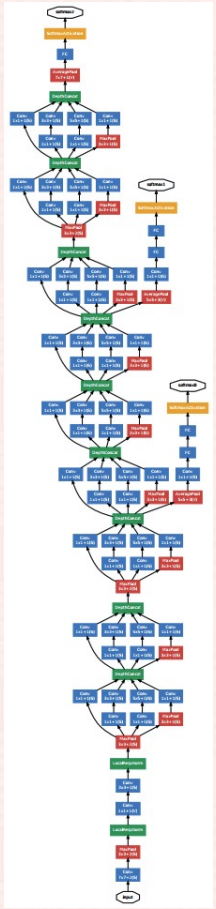
3) 1×1 컨볼루션을 사용하고 있습니다. 1×1 컨볼루션은 리셉티브 필드Receptive Field의 크기가 개별 픽셀 하나이므로 특징 추출효과는 거의 없다고 볼 수 있습니다. 1×1 컨볼루션 이후에 ReLU 활성 함수를 적용하므로 Non-Linearity가 추가되어 모델의 표현력이 더욱 강해질 수 있습니다.

❖ VGGNet은 2014년도 ILSVRC 대회에서 다음에 살펴볼 GoogLeNet에 밀려 2등을 차지했지만 GoogLeNet보다 훨씬 간결한 구조를 가지고 있기 때문에 GoogLeNet보다 널리 사용되고 있습니다.

GoogLeNet(Inception v1)

- GoogLeNet은 2014년 ILSVRC 대회에서 1등을 차지한 모델입니다. GoogLeNet은 CNN의 한 층을 기존의 컨볼루션층 대신에 구글에서 자체적으로 개발한 인셉션 모듈Inception Module이라는 조금은 복잡한 구조로 대체합니다.
- 인셉션 모듈을 사용하는 목적은 파라미터 수는 감소시키면서 성능은 향상시키기 위해서입니다. 다음 그림은 GoogLeNet의 구조를 보여줍니다.

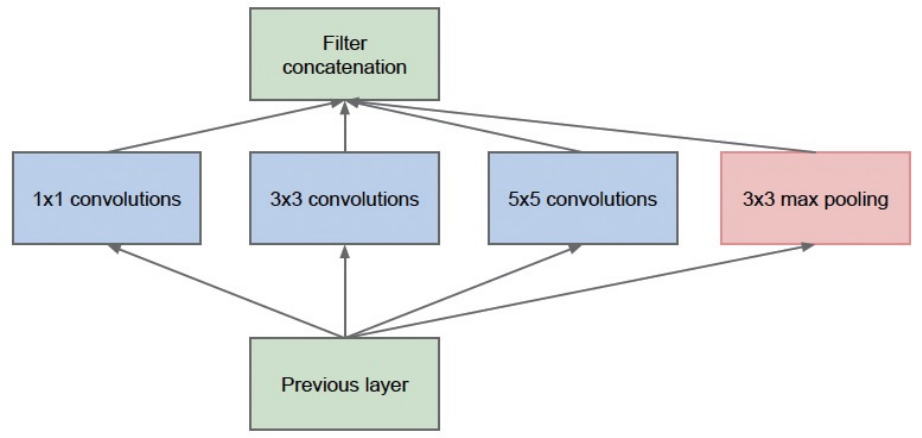
GoogLeNet(Inception v1)

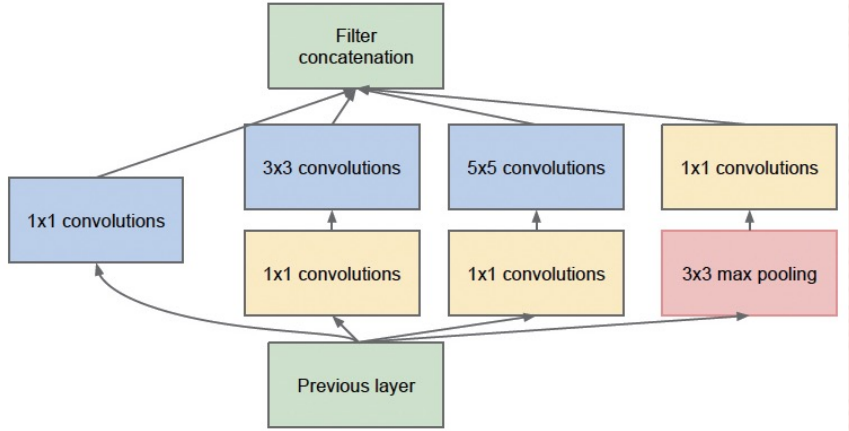


GoogLeNet(Inception v1)

- 다음 그림은 인셉션 모듈의 구조로 일반적인 버전과 차원 축소 버전을 보여줍니다.
- 구체적으로 인셉션 모듈의 한 층은 인풋으로 들어오는 이미지에 대해 1×1, 3×3, 5×5 컨볼루션, 3×3 풀링 4개의 연산을 병렬적으로 수행하고 이들 연산의 출력 결과를 concatenation으로 하나로 모아서 다음 층의 인풋으로 전달합니다. concatenation은 활성화 맵Activation map을 이어 붙이는 기법입니다.
- 예를 들어 1×1 컨볼루션의 결과로 28×28 크기의 32개의 활성화 맵이 출력되고, 3×3 컨볼루션의 결과로 28×28 크기의 64개의 활성화 맵이 출력된다면 이들 활성화 맵을 이어 붙여서 28×28 크기의 96개의 concatenation 결과를 만듭니다. 만약 활성화 맵의 크기가 다르다면 작은 크기의 활성화 맵을 큰 크기의 활성화 맵 크기에 맞추기 위해서 제로패딩을 사용합니다.
- 인셉션 모듈 차원 축소 버전은 3×3, 5×5 컨볼루션 앞과 3×3 풀링 연산 뒤에 1×1 컨볼루션 연산을 추가해서 필요한 연산량을 더욱 축소한 구조입니다. VGGNet에서 살펴 보았듯이 1×1 컨볼루션은 특징 추출 효과는 없습니다. 하지만 VGGNet에서는 ReLU를 추가해서 모델의 Non-Linearity 표현력을 증가시키기 위해서 1×1 컨볼루션층을 추가했습니다. 이에 반해 GoogLeNet은 1×1 컨볼루션을 차원을 축소해서 연산량을 감소시키기 위해서 사용합니다. 예를 들어, 32×32×64 형태의 입력값에 3×3 컨볼루션을 바로 적용하는 것보다 32×32×64 형태의 입력값에 24개의 필터를 가진 1×1 컨볼루션을 적용해서 32×32×24 크기로 입력값의 차원을 축소시킨 후, 3×3 컨볼루션을 적용하면 계산해야하는 연산량을 대폭 줄일 수 있습니다.
- 앞서 설명했듯이 이런 인셉션 모듈을 이용할 경우 파라미터의 수는 감소시키면서 성능은 향상시킬 수 있습니다.

GoogLeNet(Inception v1)





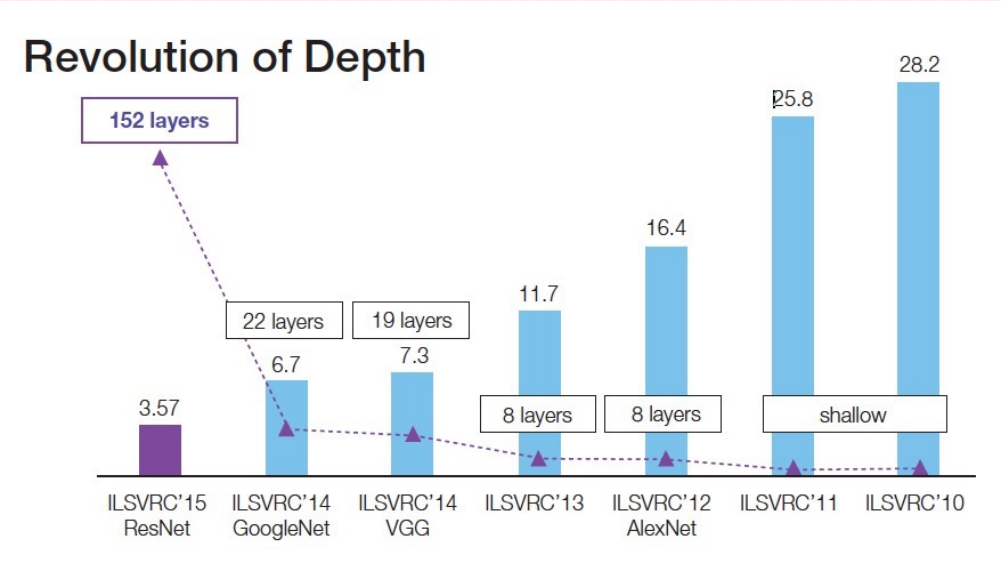
GoogLeNet(Inception v1)

- GoogLeNet의 각 층에서 수행하는 연산의 파라미터와 출력값을 표 형태로 나타내면 그림 7-13와 같습니다.
- 구글은 2014년도 ILSVRC의 우승 모델인 GoogLeNet을 Inception v1로 명명하고, 구조를 계속 개선해서 Inception v2, Inception v3, Inception v4라는 이름으로 state-of-art* CNN 모델을 계속해서 발표하고 있습니다.

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

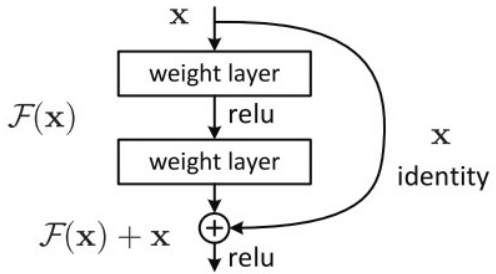
ResNet

- ResNet은 2015년도 ILSVRC 대회에서 우승을 차지한 모델입니다. ResNet은 Residual Networks의 약자로 CNN의 깊이를 152층으로 획기적으로 늘린 모델입니다.
- 그림은 ILSVRC 대회에서 연도별 우승을 차지한 모델들의 층 깊이를 나타냅니다. 2012년도 AlexNet의 등장 이후 해가 갈수록 점진적으로 깊이가 깊어졌지만 ResNet은 획기적으로 큰 폭으로 깊이를 증가시켰습니다.



ResNet

- ResNet이 층의 깊이를 이렇게 획기적으로 늘릴 수 있었던 원동력은 Residual Block이라는 모듈에 있습니다. 그림은 Residual Block을 나타냅니다. Residual Block은 Skip Connection이라는 컨볼루션 연산을 수행하지 않고 이전 층에서 넘어온 인풋 x를 그대로 넘겨주는 별도의 연결을 추가해줍니다. Skip Connection을 추가한 연산 과정을 수식으로 나타내면 다음과 같습니다.



$$H(x) = F(x) + x$$

- Skip Connection의 역할은 옵티마이저가 좀 더 손쉽게 파라미터를 최적화 할 수 있도록 도와줍니다. 다소 거칠게 설명하면 Skip Connection은 옵티마이저가 10만큼 학습을 진행해야 했던 상황에서 7만큼을 Skip Connection으로 미리 넘겨주어서 3만큼만 학습해도 되도록 만들어줍니다. 따라서 층의 깊이가 깊어져 학습해야 하는 파라미터의 수가 늘어나더라도 옵티마이저가 최적의 파라미터를 찾아내 CNN의 성능을 더욱 향상시킬 수 있습니다.

ResNet

- ResNet이 2015년도 ILSVRC에 우승을 차지하자 이후부터 최근까지 등장한 CNN 모델들은 Skip Connection을 추가한 Residual Block을 광범위하게 사용하고 있습니다.

Fine-Tuning(Transfer Learning)

- 전이 학습(Transfer Learning)은 이미 학습된 Neural Networks의 파라미터를 새로운 Task에 맞게 다시 미세조정(Fine-Tuning)합니다.
- Fine-Tuning을 다른 말로 Transfer Learning(전이 학습)이라고도 부릅니다.

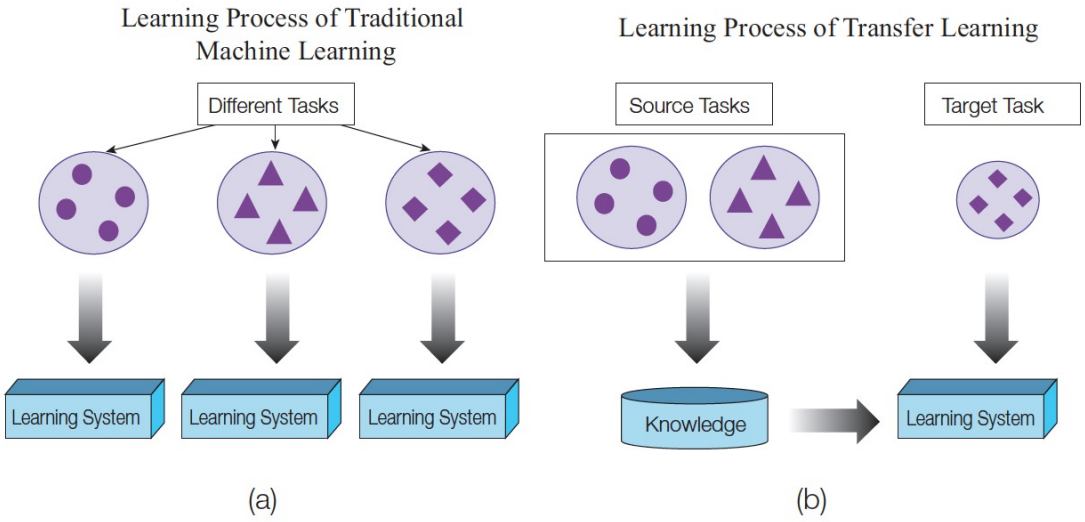


그림 6-5 | 전이 학습의 예

PyTorch에서 제공하는 사전 학습된 모델들

- 지금까지 살펴본 CNN 모델들은 PyTorch에서 미리 사전 학습된 파라미터와 모델 구현체를 제공하기 때문에 손쉽게 가져와 사용할 수 있습니다.
- https://pytorch.org/vision/stable/models.html

Classification

The models subpackage contains definitions for the following model architectures for image classification:

- AlexNet
- VGG
- ResNet
- SqueezeNet
- DenseNet
- Inception v3
- GoogLeNet
- ShuffleNet v2
- MobileNetV2
- MobileNetV3
- ResNeXt
- Wide ResNet
- MNASNet
- EfficientNet
- RegNet

파이토치(PyTorch)에서 사전학습된 CNN 모델 사용 및 파인튜닝하기

- 사전 학습된 CNN 모델 사용 및 파인튜닝하기
- 5강_PyTorch를_이용해서_사전_학습된_CNN_모델_사용_및_fine_tuning.ipynb
- https://colab.research.google.com/drive/1boOyDntwSrEnooii8W4Ppma1cY2rmFGR?usp=sharing