

합성곱신경망: 컴퓨터비전 (1부)

소개

- 대뇌의 시각피질 연구에서 출발하여 1980년대부터 이미지인식 분야에서 사용됨.
- 일부 복잡한 이미지 처리문제에서 사람의 능력 초과함.
- 주요 활용 예제: 이미지검색 서비스, 자율주행 자동차, 영상 자동분류 등
- 음성인식, 자연어 처리 등 다양한 분야에서도 활용됨.

주요 내용

- CNN의 구성요소
- TF와 케라스를 이용한 CNN의 구현
- 가장 뛰어난 성능의 CNN 구조 살펴보기
- 활용예제
 - 객체탐지(object detection)
 - * 이미지 상에서 여러 객체 구분하고 객체 주위에 경계상자(bounding box) 그리기
 - 의미분할(semantic segmentation, 시맨틱 분할)
 - * 서로 다른 객체에 속한 픽셀 분류하기

시각피질 구조와 합성곱 신경망

- 시각피질 안의 많은 뉴런이 작은 국부수용장(local receptive field)를 가진 것으로 보임.

#####

A bit of history:

Hubel & Wiesel,

1959

RECEPTIVE FIELDS OF SINGLE NEURONES IN THE CAT'S STRIATE CORTEX

1962

RECEPTIVE FIELDS, BINOCULAR INTERACTION AND FUNCTIONAL ARCHITECTURE IN THE CAT'S VISUAL CORTEX

1968...

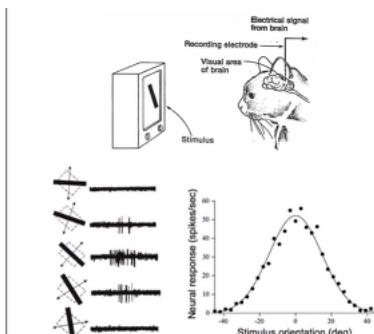


Figure 1: homl14-02

- 국부수용장 모델 모방

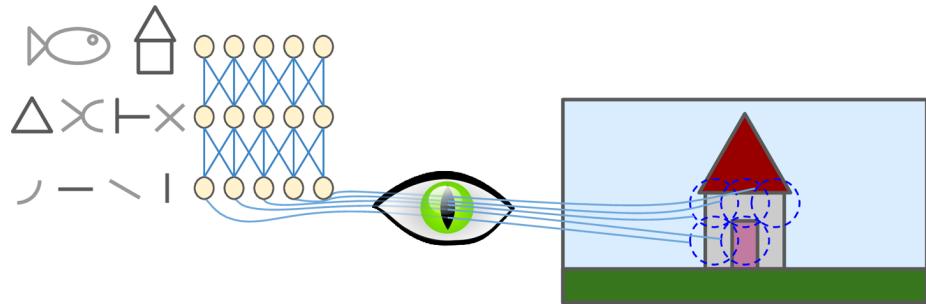


Figure 2: homl14-01

- 합성곱신경망(CNN)으로 발전
 - 합성곱 층(convolution layer)
 - 풀링 층(pooling layer)

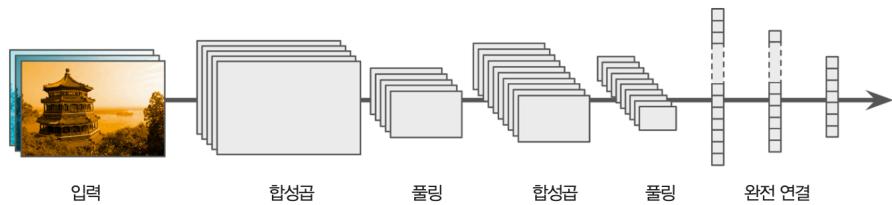


Figure 3: homl14-03

합성곱 층(Convolutional layer)

- CNN의 가장 중요한 구성요소
- 사각 형태의 국부수용장을 모방한 CNN 층
- 층과 제로패딩 사이의 연결
 - padding="SAME"
- 보폭(스트라이드, stride) 2를 사용한 차원축소 지원 CNN 층

필터 (합성곱 커널)

- 입력뉴런에 사용될 가중치 역할 수행
- 필터의 모양과 크기가 국부수용장의 모양과 크기를 지정함.
 - 필터는 넘파이 어레이로 지정됨.

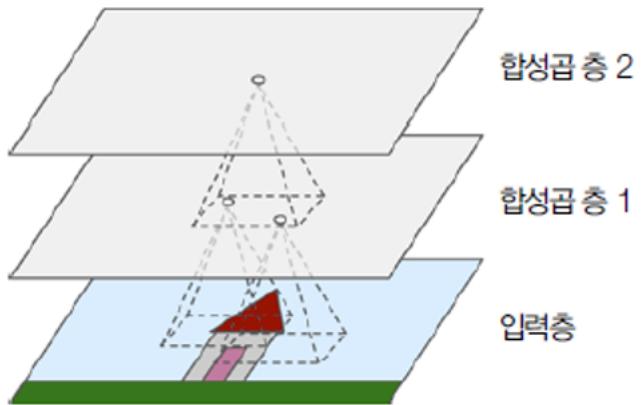


Figure 4: homl14-05a

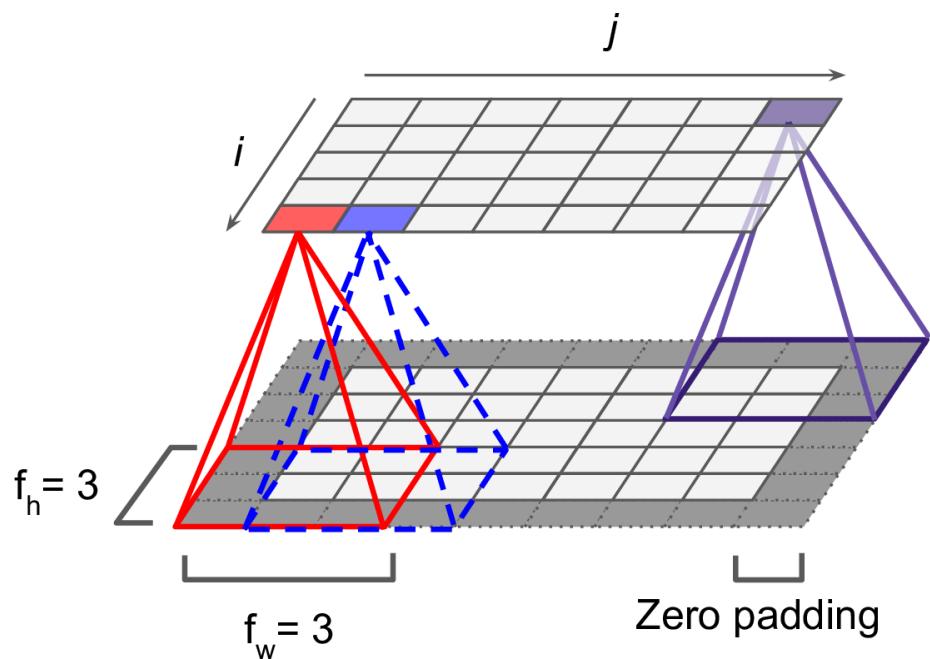


Figure 5: homl14-06

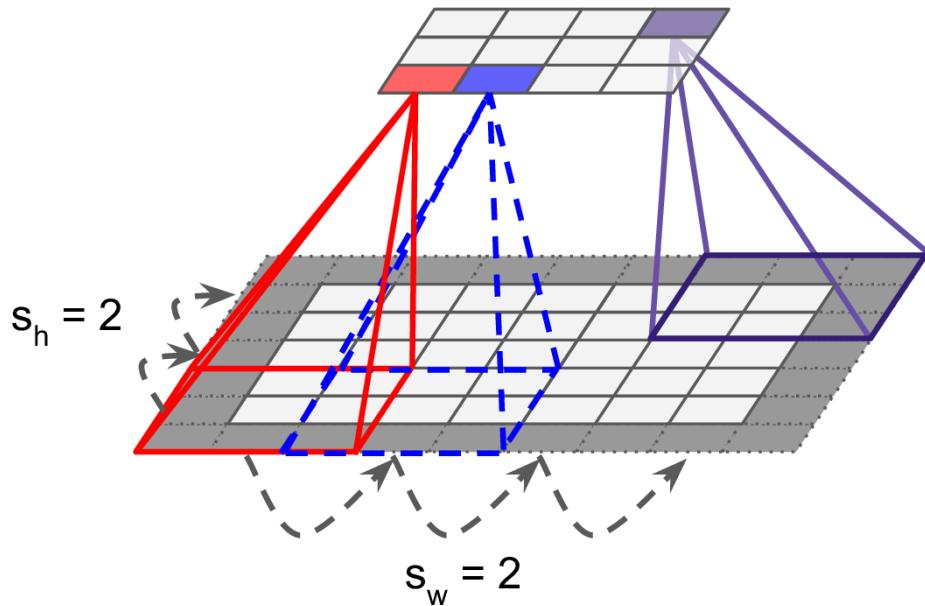


Figure 6: homl14-07

- 다양한 필터 사용

- 필터 수는 하이퍼파라미터로 지정.

- 예제

```
filters = np.zeros(shape=(7, 7, channels, 2), dtype=np.float32)
filters[:, :, 0] = 1 # 수직필터
filters[3, :, :, 1] = 1 # 수평필터
```

특성지도(feature map)

- 특성지도: 필터 각각을 사용하여 생성된 출력값
- 수십, 수백 개의 필터를 사용함.
- 각 특성지도의 픽셀이 하나의 뉴런에 해당
- 필터에 포함된 모든 뉴런은 동일한 가중치와 편향 사용
- 하지만 필터마다 사용되는 가중치와 편향은 다름.

컬러채널(color channel)

- 이미지를 대상으로 하는 합성곱 층은 아래와 같이 3차원으로 표현 가능

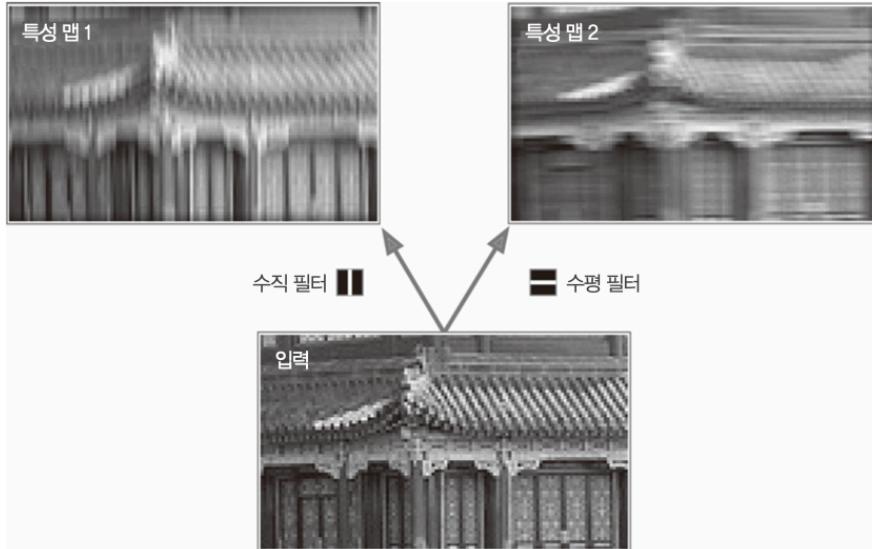


Figure 7: homl14-08

- 입력 이미지가 컬러인 경우 R, G, B 세 개의 채널, 흑백인 경우 하나의 채널 사용

뉴런의 출력값

- 각 뉴런의 출력값: 입력에 대한 가중치의 합에 편향을 더한 값

풀링 층(pooling layer)

- 계산량과 메모리 사용량을 줄이면서 과대적합의 위험도를 줄여주는 용도로 사용됨.
- 가중치 사용하지 않지만, 보폭(stride)를 사용하여 차원을 축소시키는 기능 수행.

최대 풀링 층(max pooling layer)

- 예제
 - 2×2 크기의 풀링 커널(pooling kernel)
 - * 네 개의 셀에 속한 값들 중에서 가장 큰 값만 상위 층으로 전달됨.
 - 보폭(stride): 2
 - * 하위 입력층에서 두 칸씩 건너 뛰며 풀링커널 적용. 상위층의 뉴런 수가 1/4로 줄어듦.
 - 패딩 없음(padding="valid")

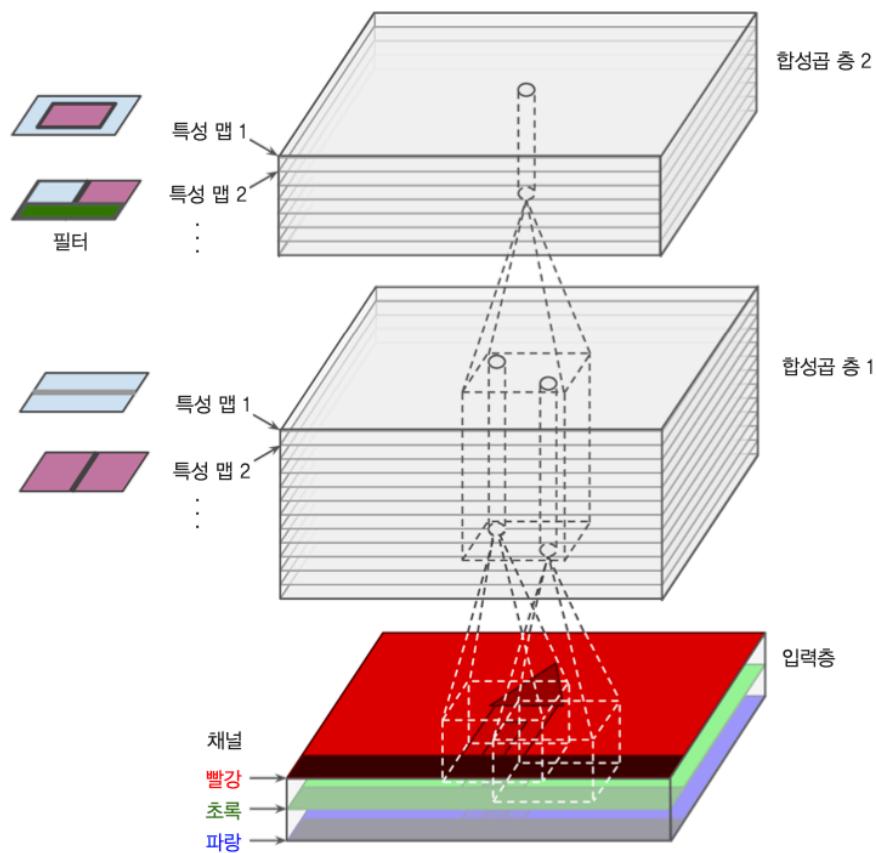


Figure 8: homl14-09

$$z_{i,j,k} = b_k + \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{k'=0}^{f_r-1} x_{i',j',k'} \times w_{u,v,k',k} \quad \text{여기서 } \begin{cases} i' = i \times s_h + u \\ j' = j \times s_w + v \end{cases}$$

Figure 9: homl14-13

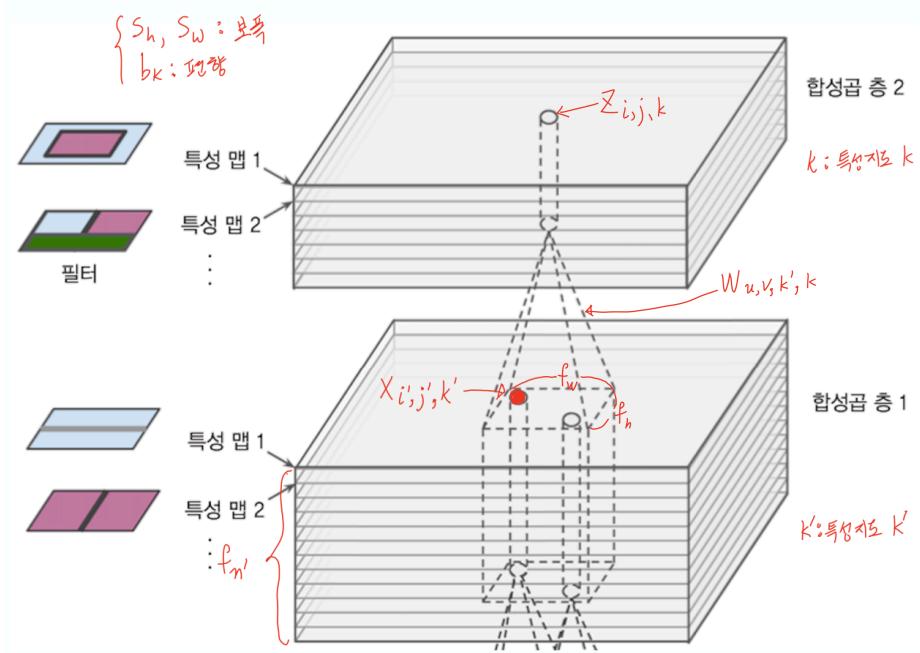


Figure 10: homl14-09d

* 보폭에 따라 일부 행과 열이 무시될 수 있음.

- 최대 풀링층은 아래와 같이 만들 수 있음.

```
tf.keras.layers.MaxPool2D(  
    pool_size=(2, 2), strides=None, padding='valid', data_format=None, **kwargs  
)
```

특징

- 파라미터 수를 획기적으로 줄여 계산량, 메모리 사용량을 줄어줌.
- 많은 정보를 잃게 되지만 그래도 잘 작동함.
- 작은 변화에 대한 어느 정도의 불변성 보장됨(아래 그림 참조).
- 하지만: 시맨틱 분할의 경우에는 불변성 대신에 등변성(equivariance)이 중요하기도 함.
 - 시맨틱 분할: 픽셀이 속한 객체에 따라 픽셀 구분하기

평균 풀링층

- 풀링커널 구역내의 평균값 활용
- MaxPool2D 대신에 AvgPool2D 사용함.

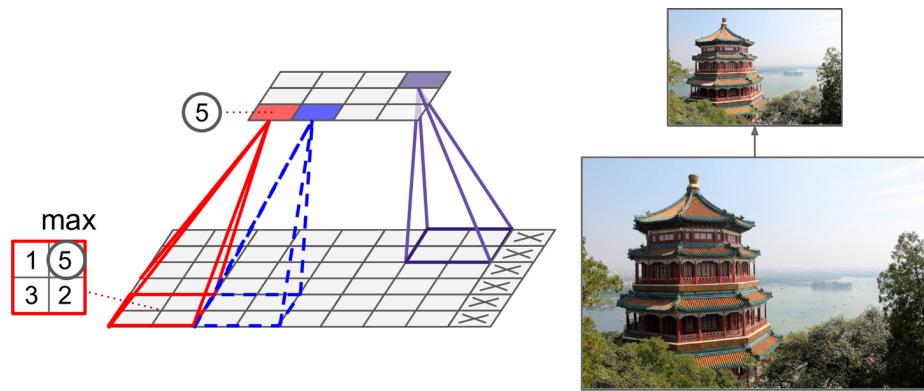


Figure 11: homl14-10

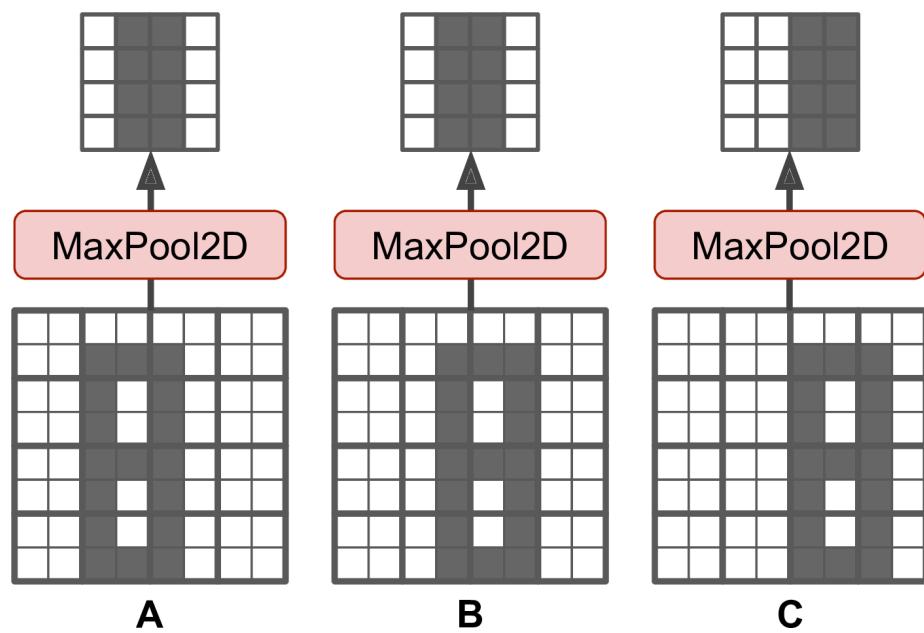


Figure 12: homl14-11

- 하지만 최대 풀링층에 비해 성능 떨어짐.
 - 최대값을 유지하여 보다 강한 특성이 학습에 사용되기 때문임.

전역평균 풀링층

- 각 특성지도의 평균 계산

```
global_avg_pool = keras.layers.GlobalAvgPool2D()
```

또는

```
global_avg_pool = keras.layers.Lambda(lambda X: tf.reduce_mean(X, axis=[1, 2]))
```

- 샘플의 각 특성지도에 대해 하나의 숫자 출력
- 출력층에서 유용하게 활용될 수 있음.
- 현대 신경망 구조에서 종종 활용됨.

깊이별 최대/평균 풀링층

- 각각 특성지도에 대해 공간(너비와 높이)별로 최대/평균을 계산하는 것 대신에 지정된 수만큼의 특성지도를 대상으로 최대/평균을 계산하는 풀링층
- 특성지도 수가 줄어듦. 풀링커널 크기를 1로 지정하여 특성지도의 크기는 유지하는 것이 기본임.

활용

- 다양한 특성에 대한 불변성 학습 가능
- 예제: 손글씨 인식을 위해 동일한 패턴을 회전시키는 여러 개의 필터를 사용하는 경우 활용 가능
 - 두께, 밝기, 왜곡, 색상 등 회전에 상관 없는 불변성 확인 가능.
- 케라스에서 지원하지 않지만 쉽게 구현 가능

```
class DepthMaxPool(keras.layers.Layer):
    def __init__(self, pool_size, strides=None, padding="VALID", **kwargs):
        super().__init__(**kwargs)
        if strides is None:
            strides = pool_size
        self.pool_size = pool_size
        self.strides = strides
        self.padding = padding
    def call(self, inputs):
        return tf.nn.max_pool(inputs,
                             ksize=(1, 1, 1, self.pool_size),
```

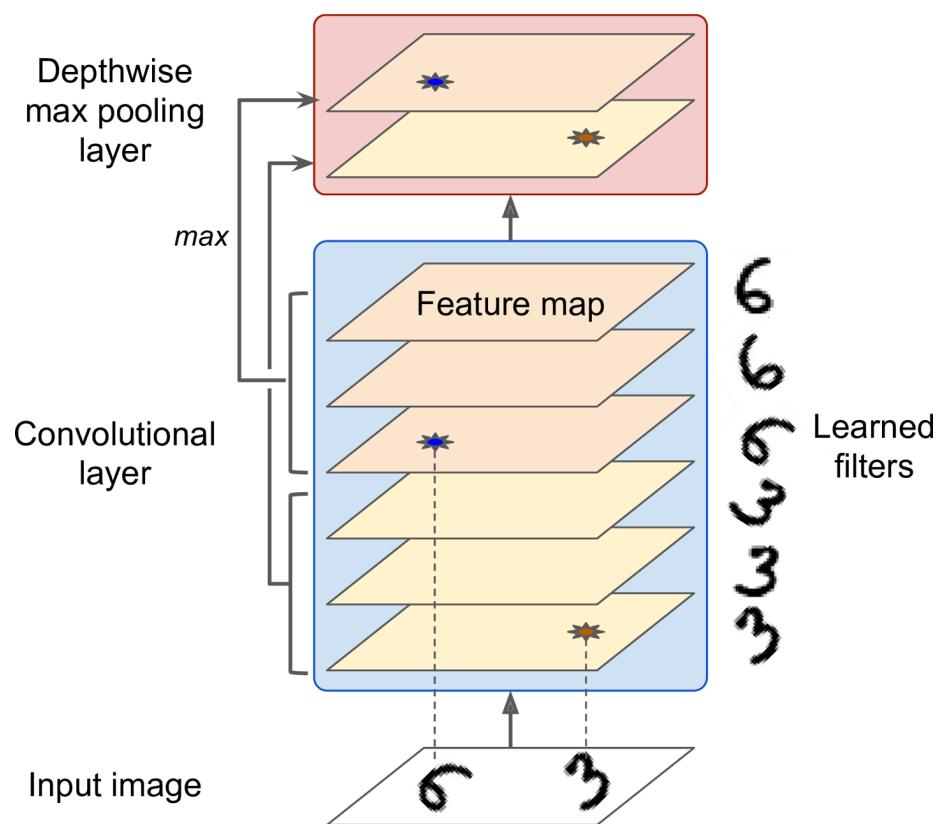


Figure 13: homl14-12

```
strides=(1, 1, 1, self.pool_size),  
padding=self.padding)
```

CNN 구조

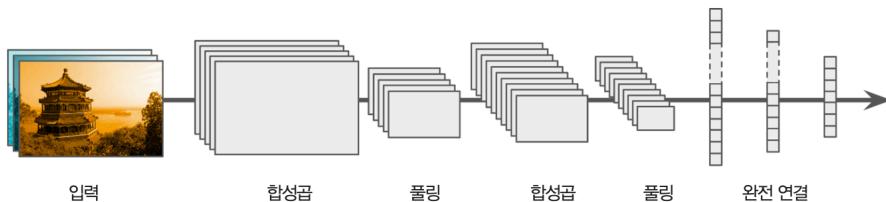


Figure 14: homl14-03

케라스 활용: 패션 MNIST

- 아래 합성곱 모델이 92% 정도의 정확도 성능 발휘
- 10장의 밀집 네트워크보다 좋은 성능임.

```
DefaultConv2D = partial(keras.layers.Conv2D,  
kernel_size=3, activation='relu', padding="SAME")
```

```
model = keras.models.Sequential([  
    DefaultConv2D(filters=64, kernel_size=7, input_shape=[28, 28, 1]),  
    keras.layers.MaxPooling2D(pool_size=2),  
    DefaultConv2D(filters=128),  
    DefaultConv2D(filters=128),  
    keras.layers.MaxPooling2D(pool_size=2),  
    DefaultConv2D(filters=256),  
    DefaultConv2D(filters=256),  
    keras.layers.MaxPooling2D(pool_size=2),  
    keras.layers.Flatten(),  
    keras.layers.Dense(units=128, activation='relu'),  
    keras.layers.Dropout(0.5),  
    keras.layers.Dense(units=64, activation='relu'),  
    keras.layers.Dropout(0.5),  
    keras.layers.Dense(units=10, activation='softmax'),  
])
```

CNN 예제: LeNet-5

- 1998년에 소개됨.

<그림 출처: LeNet-T CNN>

| 층 | 종류 | 특성 맵 크기 | 커널 크기 | 스트라이드 | 활성화 함수 |
|----|-------|---------|-------|-------|--------|
| 출력 | 완전 연결 | - | 10 | - | - |
| F6 | 완전 연결 | - | 84 | - | - |
| C5 | 합성곱 | 120 | 1×1 | 5×5 | 1 |
| S4 | 평균 풀링 | 16 | 5×5 | 2×2 | 2 |
| C3 | 합성곱 | 16 | 10×10 | 5×5 | 1 |
| S2 | 평균 풀링 | 6 | 14×14 | 2×2 | 2 |
| C1 | 합성곱 | 6 | 28×28 | 5×5 | 1 |
| 입력 | 입력 | 1 | 32×32 | - | - |

Figure 15: homl14-14a

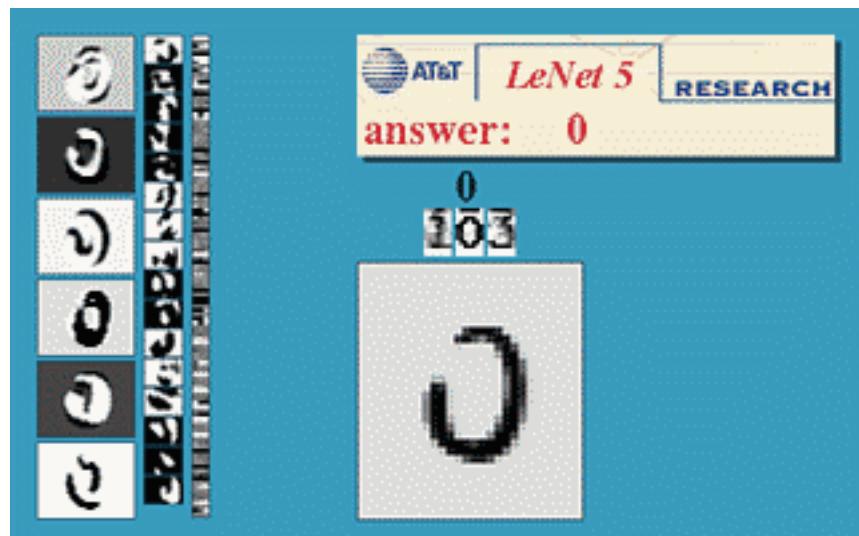


Figure 16: homl14-16

CNN 예제: AlexNet

- 2012년 ILSVRC 우승 모델
 - 톱-5 에러율: 17%

| 층 | 종류 | 특성 맵 | 크기 | 커널 크기 | 스트라이드 | 패딩 | 활성화 함수 |
|-----|-------|---------|---------|-------|-------|-------|---------|
| 출력 | 완전 연결 | - | 1,000 | - | - | - | Softmax |
| F10 | 완전 연결 | - | 4,096 | - | - | - | ReLU |
| F9 | 완전 연결 | - | 4,096 | - | - | - | ReLU |
| F8 | 최대 풀링 | 256 | 6×6 | 3×3 | 2 | valid | - |
| C7 | 합성곱 | 256 | 13×13 | 3×3 | 1 | same | ReLU |
| C6 | 합성곱 | 384 | 13×13 | 3×3 | 1 | same | ReLU |
| C5 | 합성곱 | 384 | 13×13 | 3×3 | 1 | same | ReLU |
| S4 | 최대 풀링 | 256 | 13×13 | 3×3 | 2 | valid | - |
| C3 | 합성곱 | 256 | 27×27 | 5×5 | 1 | same | ReLU |
| S2 | 최대 풀링 | 96 | 27×27 | 3×3 | 2 | valid | - |
| C1 | 합성곱 | 96 | 55×55 | 11×11 | 4 | valid | ReLU |
| 입력 | 입력 | 3 (RGB) | 227×227 | - | - | - | - |

Figure 17: homl14-15a

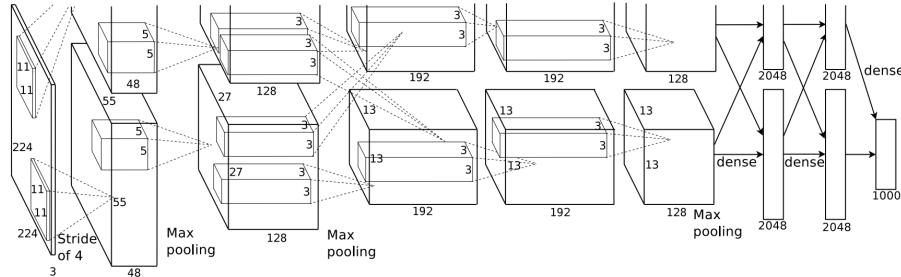


Figure 18: homl14-15b

<그림 참조: ImageNet Classification with Deep Convolutional Neural Networks>

특징

- 규제 1: F9, F10에서 드롭아웃 50% 활용
- 규제 2: 데이터증식 활용
 - 데이터증식: 훈련샘플을 인공적으로 생성하는 기법. 수평 뒤집기, 간격 이동, 조명 변경 등 활용.

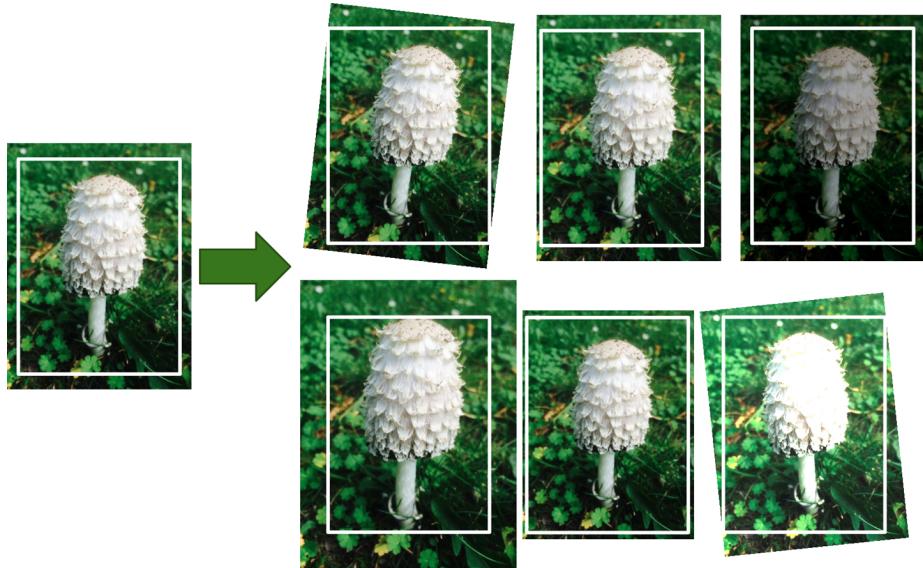


Figure 19: homl14-17

- 정규화: LRN(local response normalization)
 - 뉴런의 출력값을 보다 경쟁적으로 만드는 정규화 기법
 - 어떤 특성지도에 속한 하나의 뉴런의 활성화 함수의 반환값이 클 경우 주변 특성지도의 동일한 위치에 뉴런의 활성화 함수값을 크게 만들어줌.
 - 각각의 특성지도를 보다 특별하게 만들어 보다 다양한 특성을 탐색할 수 있도록 도와줌.

CNN 예제: GoogLeNet

- 2014년 ILSVRC 우승 모델
 - 틈-5 에러율: 7%
- 인셉션 모듈이라는 서브 네트워크 활용
 - $3 \times 3 + 1(S)$: 3×3 커널, 보폭 1, “same” 패딩
 - 합성곱 층: ReLU 활성화 함수 사용
 - 둘째 층: 다양한 패턴을 다양한 스케일로 파악하기 위한 용도
 - 1×1 커널 사용 층:
 - * 깊이별 패턴을 확인하며, 다른 합성곱 층과 연계하는 역할 수행.
- 깊이 연결
 - 4개의 합성곱 층의 결과를 쌓은 후 출력
 - tf.concat() 함수 활용

구조

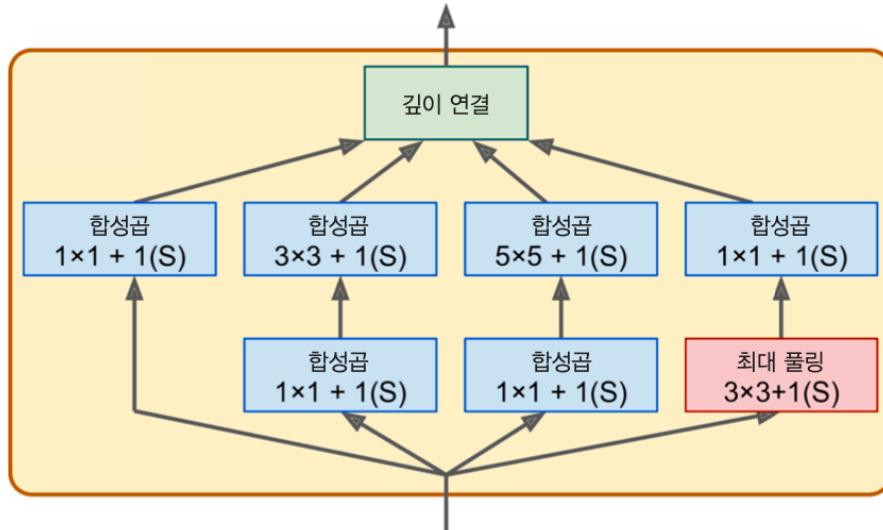


Figure 20: homl14-18

CNN 예제: VGGNet

- 2014년 ILSVRC 준우승 모델
 - 톱-5 에러율: 8~10%
- (2~3개의 합성곱 층 + 풀링층)의 단순 반복
- 총 16~19개의 합성곱 층 사용
- 추가적으로 2개의 은닉층과 출력층으로 구성된 밀집층 사용.
- 다수의 3x3 커널 필터 사용

CNN 예제: ResNet

- 2015년 ILSVRC 우승 모델
 - 톱-5 에러율: 3.6%
- ResNet-152: 152개의 합성곱 층 사용. 하지만 파라미터 수는 그렇게 많지 않음.
- ResNet-34: 34개의 합성곱 층 모델도 있음. (구글 코랩 노트북 참조)

잔차 유닛(residual unit, RU)

- 많은 층으로 인한 많은 계산을 줄이기 위해 잔차 유닛(RU) 활용
- RU를 활용한 잔차학습(residual learning) 효과: 스kip 연결로 인한 보다 수월한 학습 가능

ResNet 구조

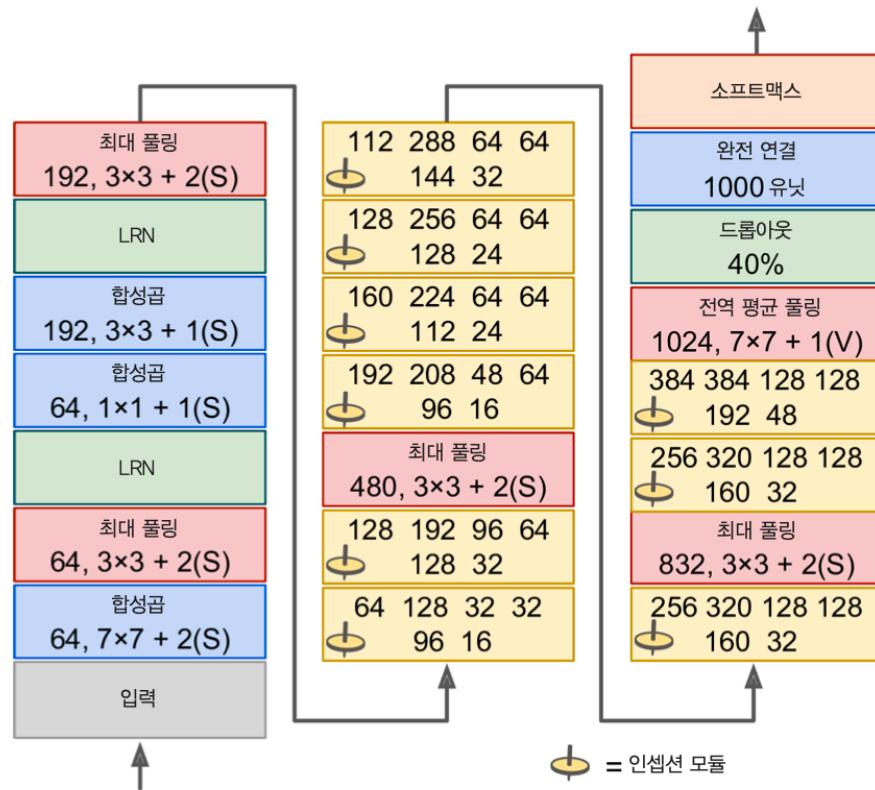


Figure 21: homl14-19

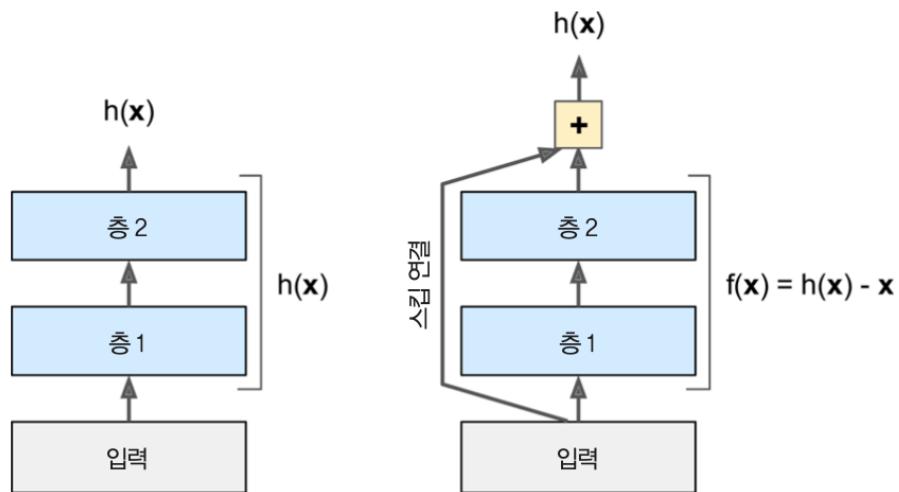


Figure 22: homl14-20

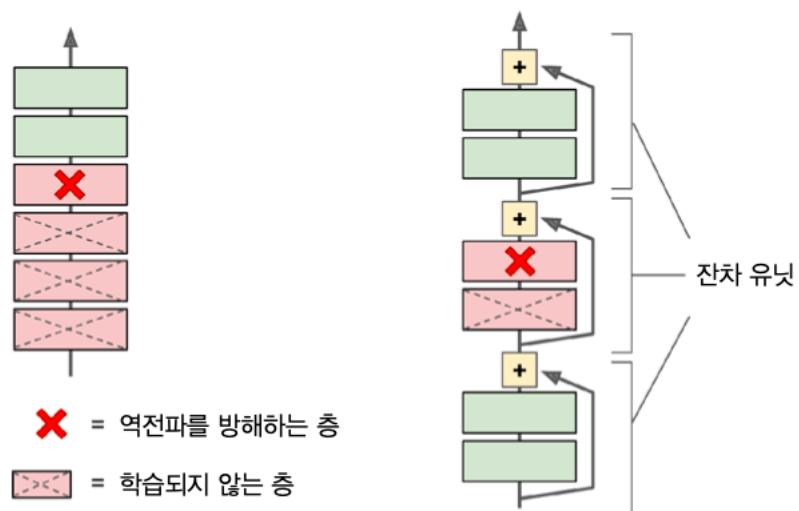


Figure 23: homl14-21

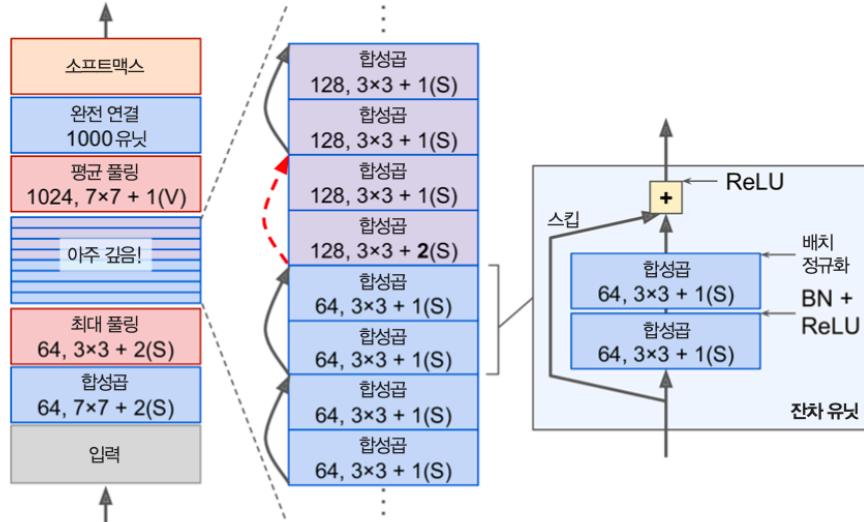


Figure 24: homl14-22

- 특성지도의 수가 몇 단계마다 두 배로 늘어남.
- 반면에 뉴런 수를 반씩 줄임($3 \times 3 + 2$ 사용, 즉, 폭이 2임.)
- 입력과 출력의 모양을 맞추기 위해 빨강색 점선 스킵 부분에 폭이 2인 합성곱 활용(아래 그림 참조)

구글의 Inception-v4

- GoogLeNet과 ResNet 모델의 합성
- 톱-5 에러율: 3% 정도

CNN 예제: Xception

- 2016년에 소개됨.
 - 케라스의 창시자인 프랑수아 솔레가 제안.
- GoogLeNet과 ResNet의 합성 버전
- GoogLeNet의 인셉션 모듈 대신 깊이별 분리합성곱 층 사용

분리 합성곱 층

- 공간별 패턴인식 합성곱 층과 깊이별 패턴인식 합성곱 층을 분리하여 연속적으로 적용
- 공간별 패턴인식: 형태 인식 (입력 특성지도마다 한개만 탐색)

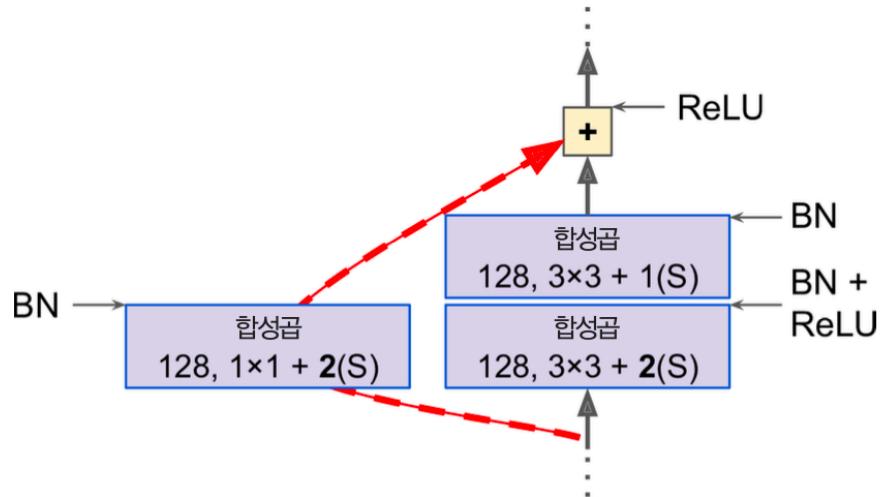


Figure 25: homl14-23

- 깊이별 패턴인식: 입, 코, 눈 으로부터 얼굴을 인식하듯 채널 사이의 패턴인식
- #####

Xception 구조 <그림 출처: Xception: Deep Learning with Depthwise Separable Convolutions>

특징

- 입력층에 많은 채널(특성지도)가 존재할 경우에만 활용
 - 따라서 보통 두 개 정도의 정상적인 합성곱 층으로 시작한 후에 깊이별 분리합성곱 층 적용 (소개된 모델은 34개 사용).
- 보다 적은 수의 파라미터, 보다 적은 양의 메모리, 보다 적은 양의 계산 요구되지만, 성능은 더 좋음.
- 합성곱 신경망의 기본 모델로 강추

CNN 예제: SENet

- 2017년 ILSVRC 우승 모델
 - 톱-5 에러율: 2.25%
- GoogLeNet과 ResNet의 합성 버전

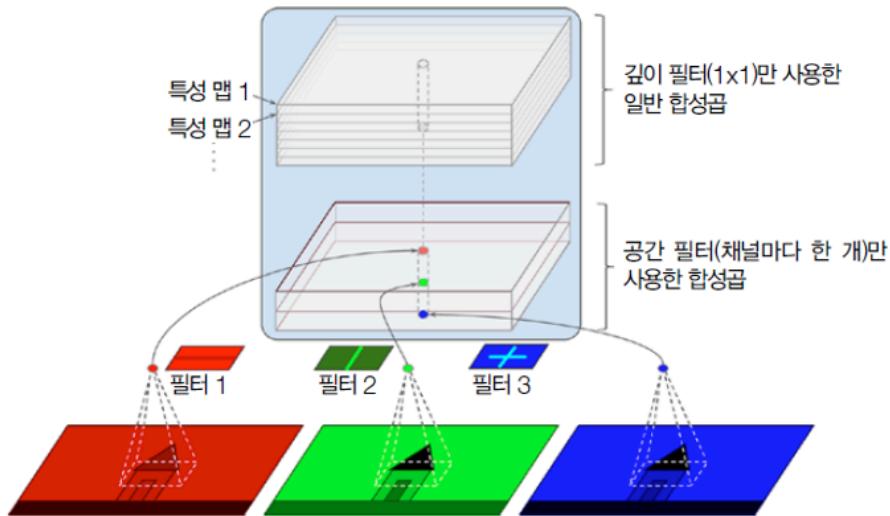


Figure 26: homl14-24b

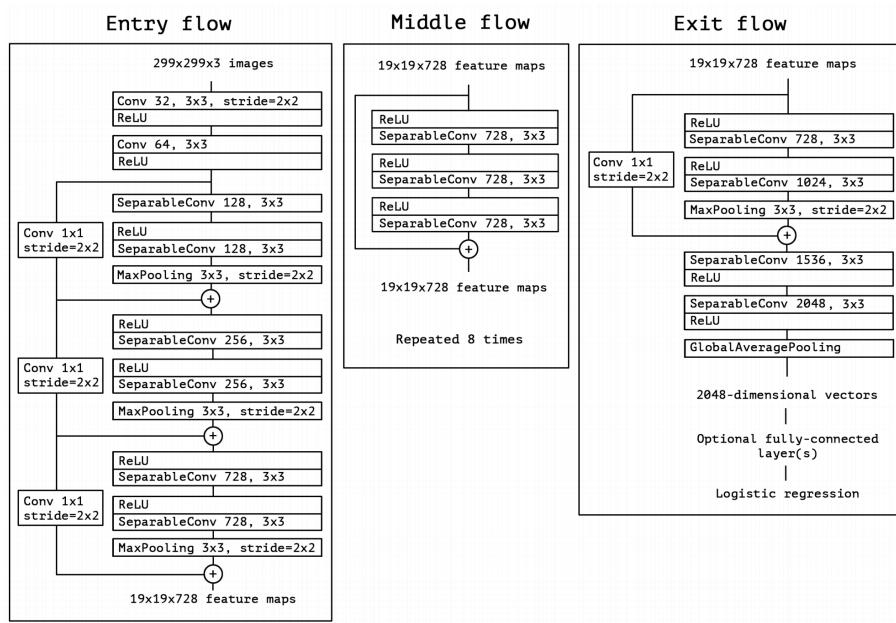


Figure 27: homl14-24c

- GoogLeNet의 인셉션 모듈과 ResNet의 잔차유닛(RU)에 SE block을 추가하여 모다 좋은 성능 발휘

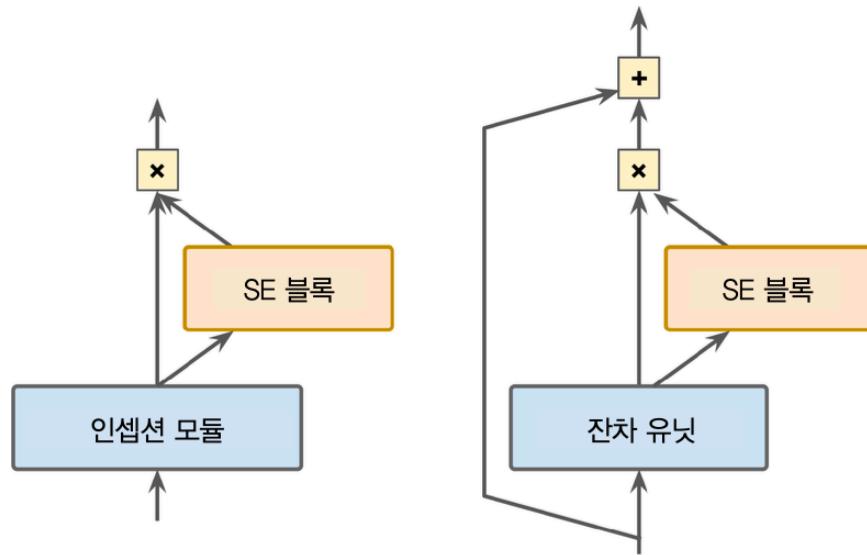


Figure 28: homl14-25

SE block 사용법

SE block 기능

- 입력된 특성지도를 대상으로 깊이별 패턴 특성 분석
- 패턴 특성들을 파악한 후 출력값 보정
 - 예제: 코와 입의 패턴이 보일 때 눈의 특성지도를 강화시킴.

SE block 구조

- 전역평균 풀링층 -> 밀집층 -> 밀집층
- 첫째 밀집 층: 뉴런 수를 16분의 1로 줄임(squeeze)
 - 특성지도들 사이의 연관성 학습 강요
- 둘째 밀집 층: 뉴런 수를 정상화시킴(excitation)
 - 학습된 연관성을 이용하여 입력 특성지도를 보정할 가중치 출력

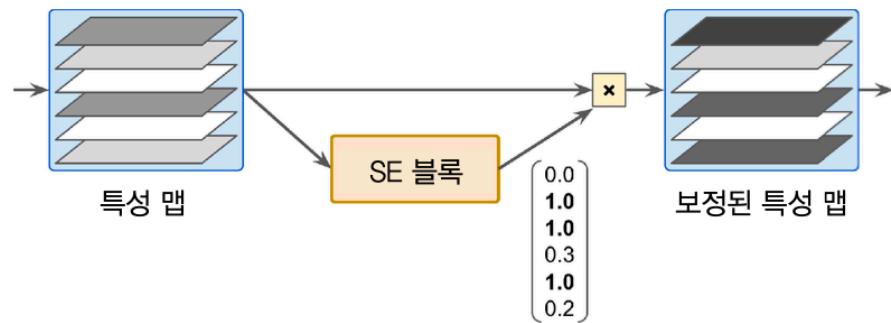


Figure 29: homl14-26

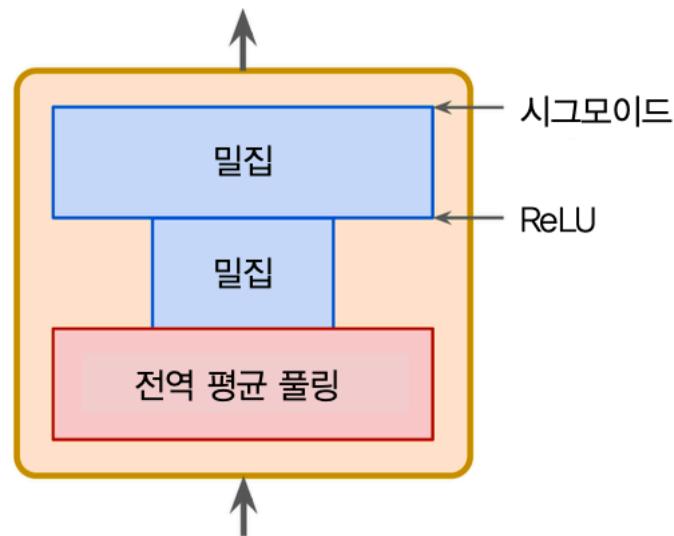


Figure 30: homl14-27