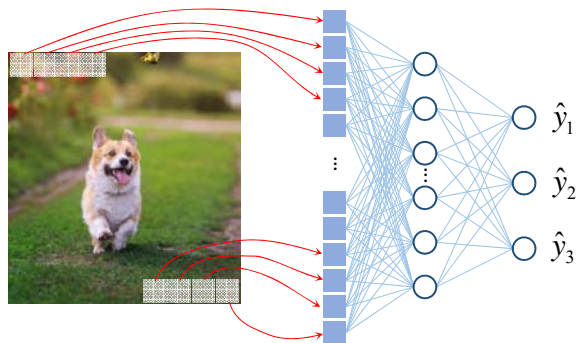


Class Activation Map

Prof. Hyunseok Oh

School of Mechanical Engineering
Gwangju Institute of Science and Technology

Brief Review of ANN and CNN

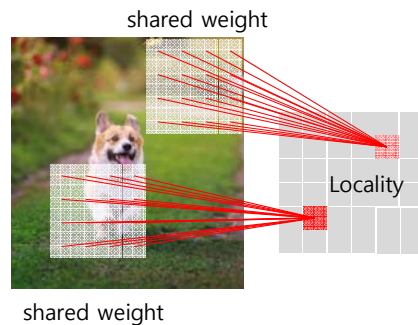


Fully connected layer

Classification

ANN

- No spatial information
- A huge number of parameters



shared weight

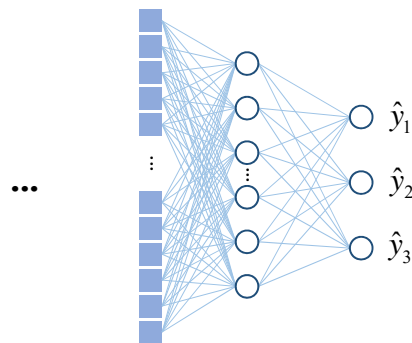
Convolution and pooling layer

Fully connected

Classification

CNN

- Preserve spatial information
- Weight sharing, local connectivity, sparse interactions



Limitations of Most Machine Learning Models

- 귀납적 접근법의 한계
 - 데이터를 기반으로 학습하기에 이론적 근거가 부족
 - 추정/분류 근거는 블랙박스로써 이해됨.
 - 많은 경우, 시행 착오 방식의 개발 절차 요구
- 부족한 설득력/신뢰도
 - 추정/분류 결과에 대한 근거 제시 능력 부족
 - 논리적 추론 능력 부재
 - 데이터의 품질, 양에 따라 달라지는 성능 변화

대안: 설명 가능한 인공지능 (XAI: Explainable Artificial Intelligence)

Why Explainable AI?

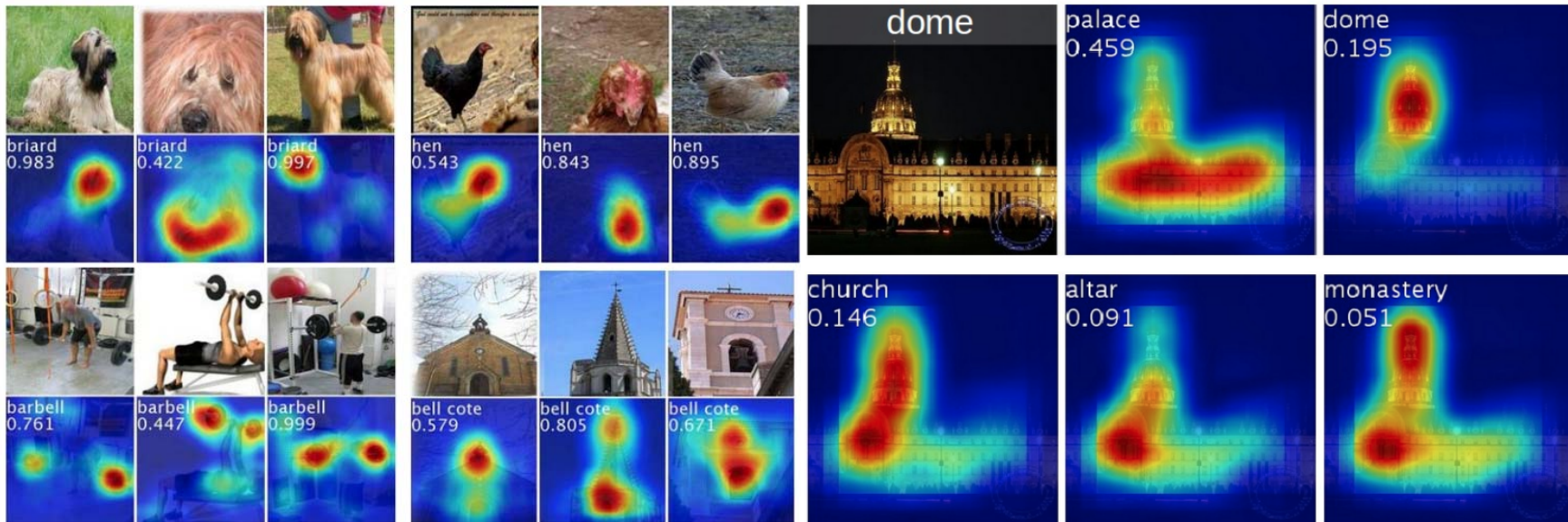
- 설명 가능한 인공지능을 이용한 AI 모델 해석
 - 인공지능 모델의 예측/분류 등의 출력 값을 인간이 이해하고 신뢰할 수 있도록 해주는 일련의 방법론
 - 설명 가능한 인공지능을 이용해 모델의 예상된 영향 및 잠재적 편향 기술을 가능케 함으로써 AI의 해석 가능성을 높일 수 있음.
- AI 성능에 따른 해석 결과 활용 방향
 - AI < 인간: AI 모델의 개선 방향 제시
 - AI = 인간: AI 모델이 학습하는 원리 규명
 - AI > 인간: AI로부터 새로운 지식 습득



- AI < 인간
 - 아키텍처 개선
 - 훈련 데이터 개선 등
- AI = 인간
 - 모델의 학습 원리 규명
 - 학습 원리 비교
- AI > 인간
 - AI의 학습 원리 분석
 - 데이터 분석 관련 지식 습득

XAI by Visualization

- 분류 혹은 예측 결과에 대한 근거의 시각화
 - 인간은 사진을 보고 특정 결과로 분류할 때, 이미지 전체가 아닌 일부를 통해 결론을 내림.
 - 현재의 xAI는 이미지 분류에 주요하게 작용한 픽셀을 활성화하는 기술
 - ex) Class Activation Map (CAM), Grad-CAM, Layer-wise relevance propagation (LRP), etc.



Class Activation Map (CAM)

- Zhou et al, “Learning deep features for discriminative localization”, CVPR, 2016.

Learning Deep Features for Discriminative Localization

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, Antonio Torralba
 Computer Science and Artificial Intelligence Laboratory, MIT
 {bzhou, khosla, agata, oliva, torralba}@csail.mit.edu

Abstract

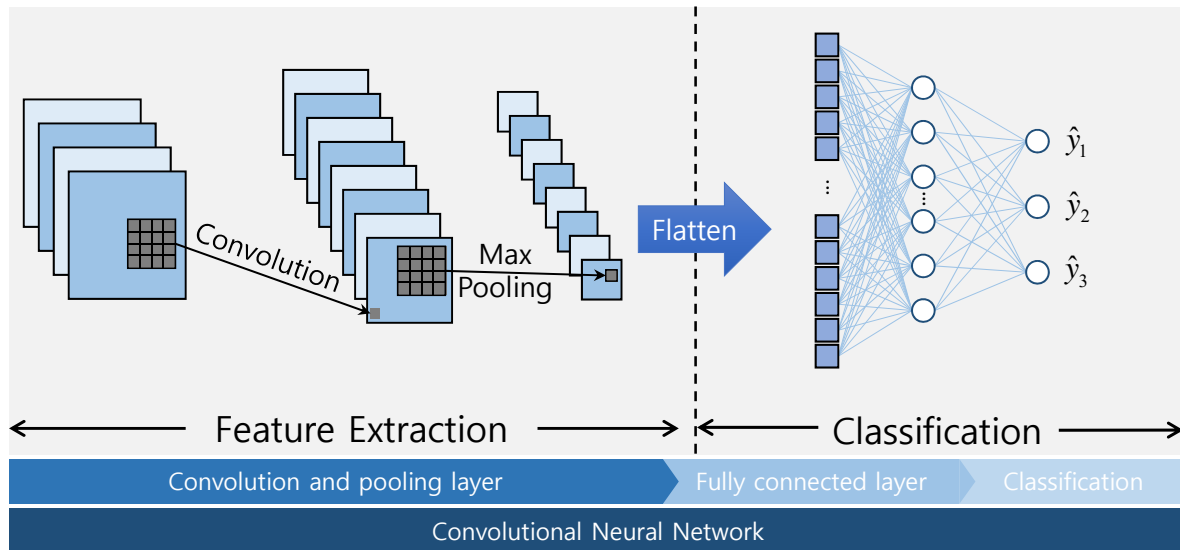
In this work, we revisit the global average pooling layer proposed in [13], and shed light on how it explicitly enables the convolutional neural network to have remarkable localization ability despite being trained on image-level labels. While this technique was previously proposed as a means for regularizing training, we find that it actually builds a generic localizable deep representation that can be applied to a variety of tasks. Despite the apparent simplicity of global average pooling, we are able to achieve 37.1% top-5 error for object localization on ILSVRC 2014, which is remarkably close to the 34.2% top-5 error achieved by a fully supervised CNN approach. We demonstrate that our network is able to localize the discriminative image regions on a variety of tasks despite not being trained for them.



Figure 1. A simple modification of the global average pooling layer combined with our class activation mapping (CAM) technique allows the classification-trained CNN to both classify the image and localize class-specific image regions in a single forward-pass e.g., the toothbrush for *brushing teeth* and the chainsaw for *cutting trees*.

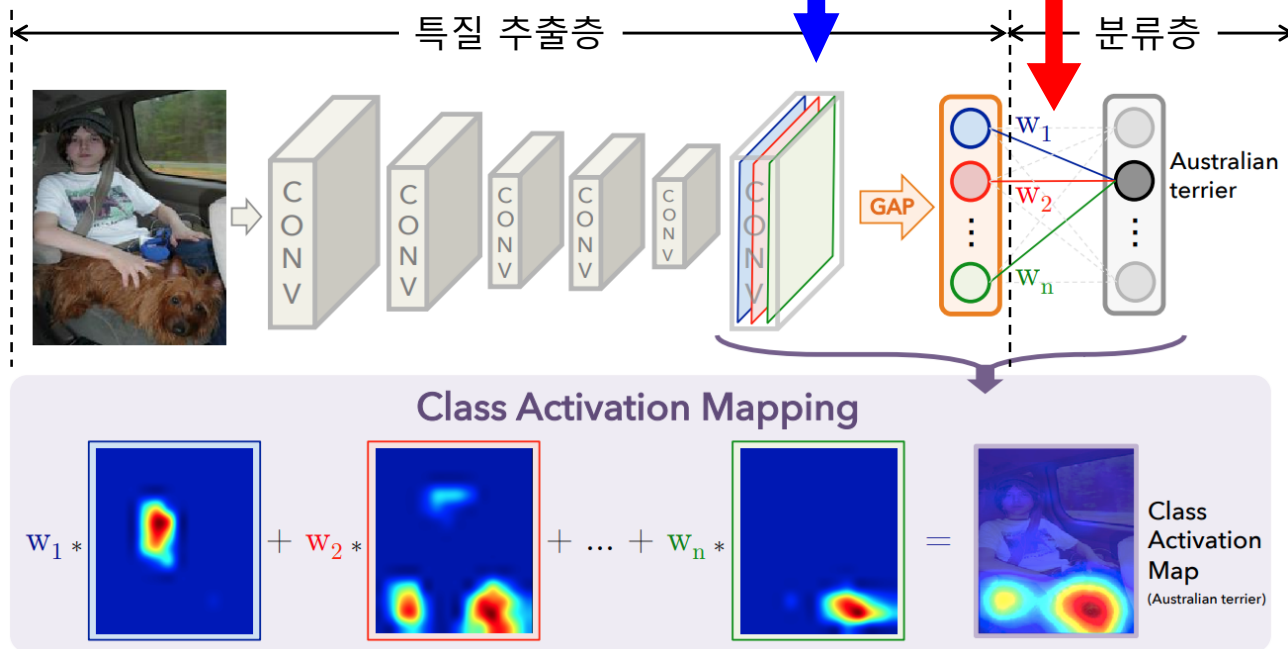
Trained CNN Model

- Feature Extraction + Classification
 - 특성인자 추출: 입력 값과 학습된 필터 값과 합성곱 연산을 통해 특성인자 맵 도출
 - 분류기: 마지막 Conv 블록의 특성인자 맵 정보를 활용하여 클래스 해당 확률 값 도출
- Using a trained CNN model, a new input is provided to predict the corresponding label.



Implementation of CAM on Trained CNN Model

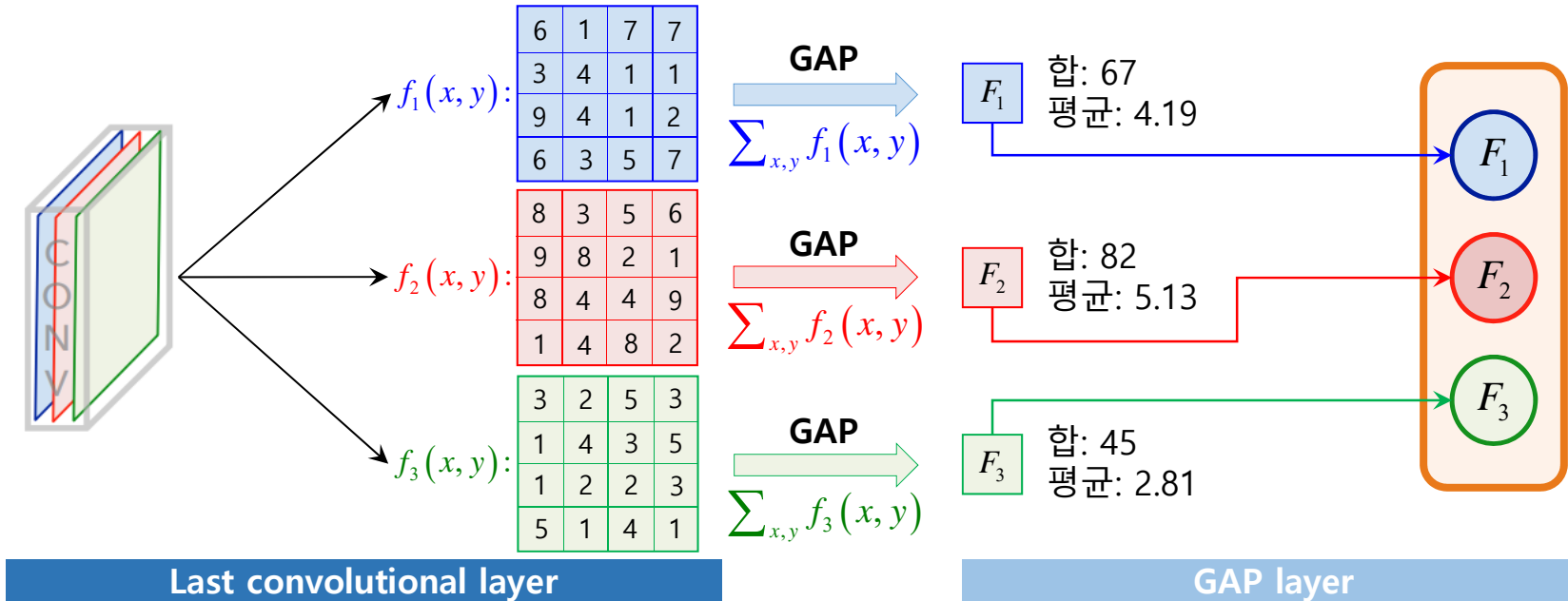
- 마지막 층의 특성인자 맵(Feature Map)
- 분류를 위한 가중치 값(Weight)



Key Component of CAM: Feature Map

• 연산

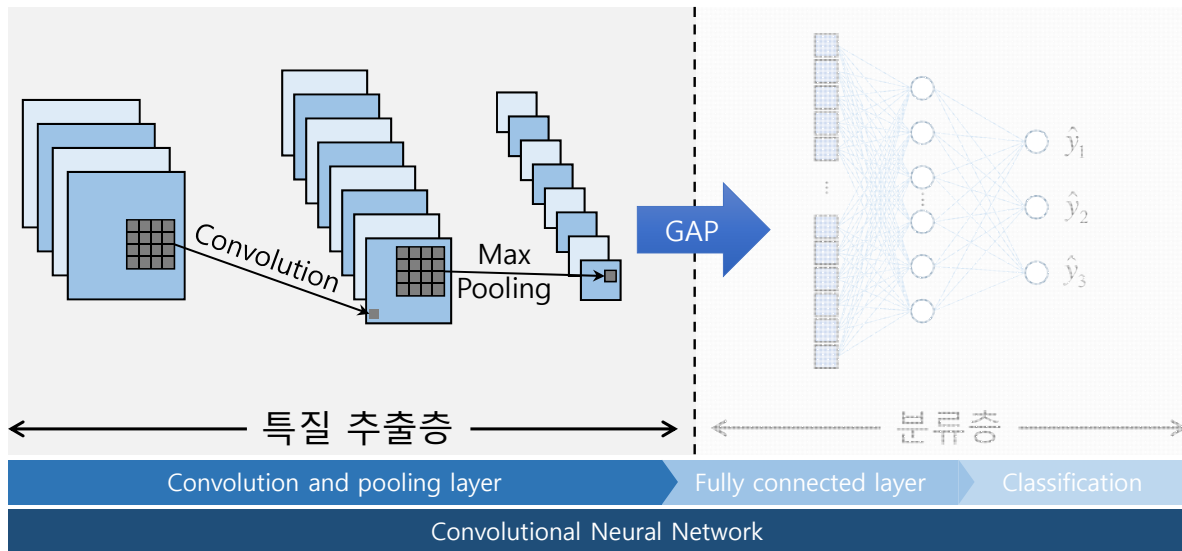
- 입력 값과 학습을 완료한 필터 값에 대해 합성곱 연산을 수행하고 그 결과를 하나의 숫자로 압축
- 한 채널의 모든 x, y 에 존재하는 값에 합(Summation) 또는 평균(Average)을 취함으로써 수행
- 이는 특질의 대푯값을 추출하는 과정으로 볼 수 있음.



Key Component of CAM: Feature Map

- 물리적 의미

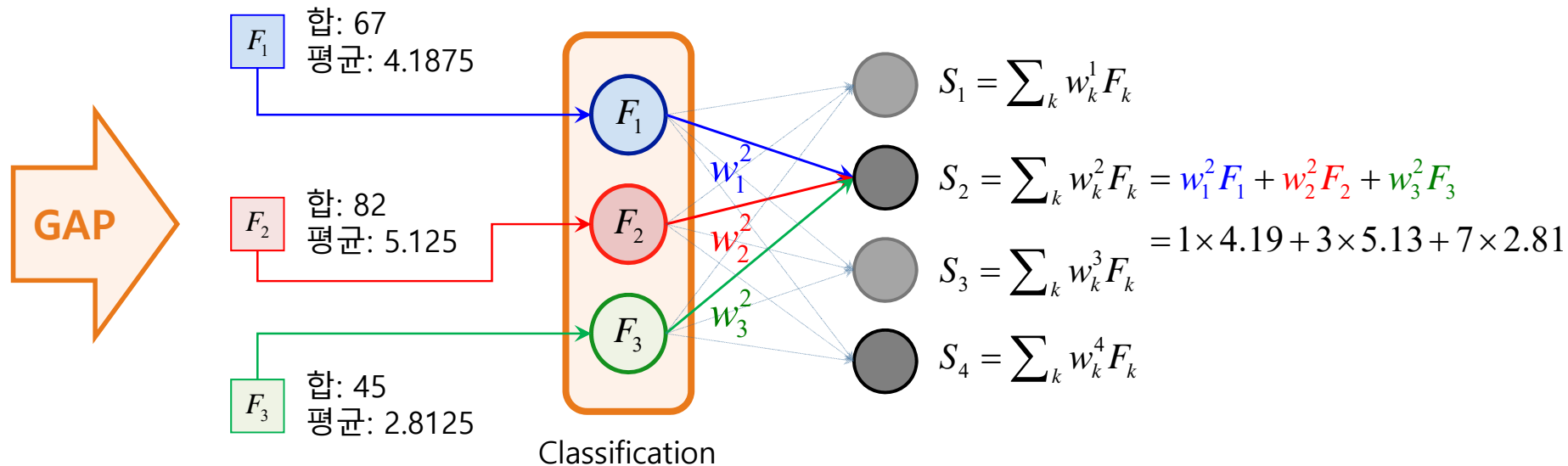
- 마지막 합성곱 층에서는 가장 많은 필터를 통과한 연산 결과를 보유
- 따라서 마지막 특성인자 맵은 고차원(상위) 특성인자들이 분포
- CAM에서는 마지막 특성인자 맵에 GAP를 취함으로써 분류하기 쉬운 특질들을 가공



Key Component of CAM: Weight

- 연산

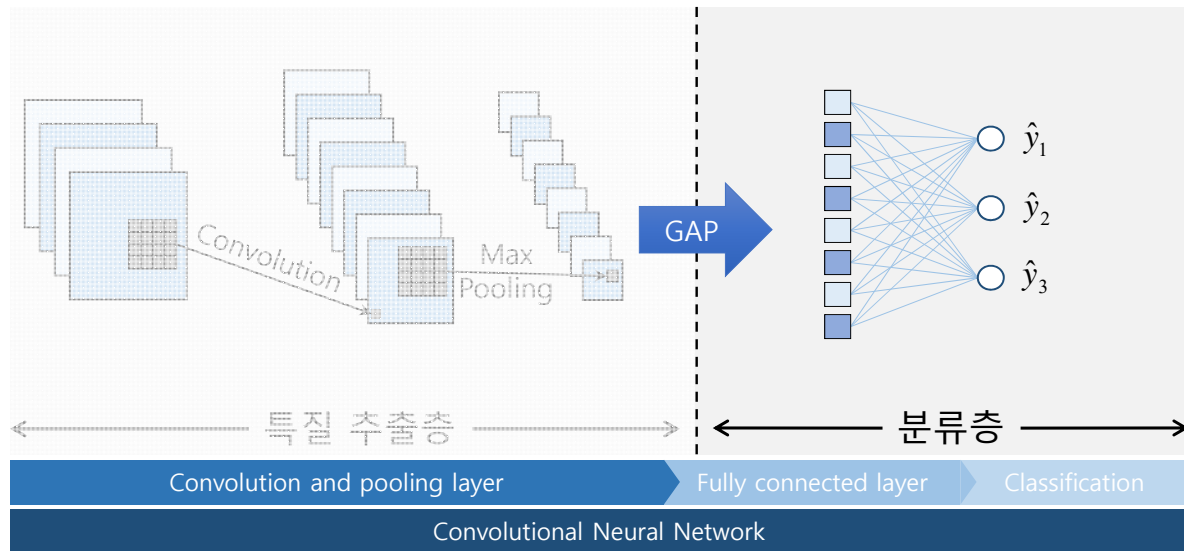
- 분류 결과와 Fully-connected layer를 구성 (Activation function: Softmax function)
- 분류를 위한 가중치 값은 훈련(Training)과정을 통해 자동으로 결정됨.



Key Component of CAM: Weight

- 물리적 의미

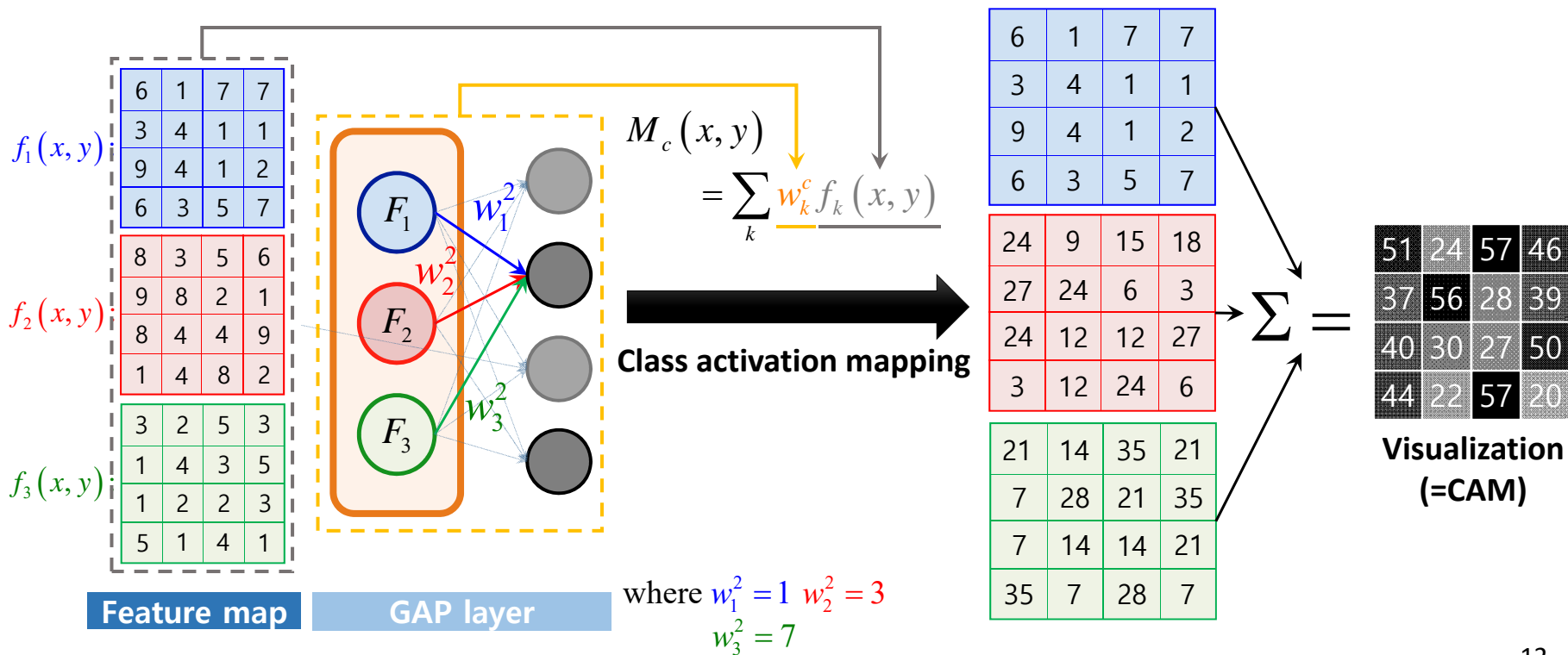
- GAP 연산을 통해 추출된 대푯값들은 최종 추출된 특징 정보를 보유
- 마지막 분류기와 Fully-connected layer를 구성함으로써 각 특징 대푯값들이 분류 결과에 얼마나 영향력을 끼치는지 가중치(Weight)를 통해 확인 가능
- 여기서의 가중치는 특징 대푯값의 중요도를 평가한 결과로 추론



Class Activation Mapping

• 연산

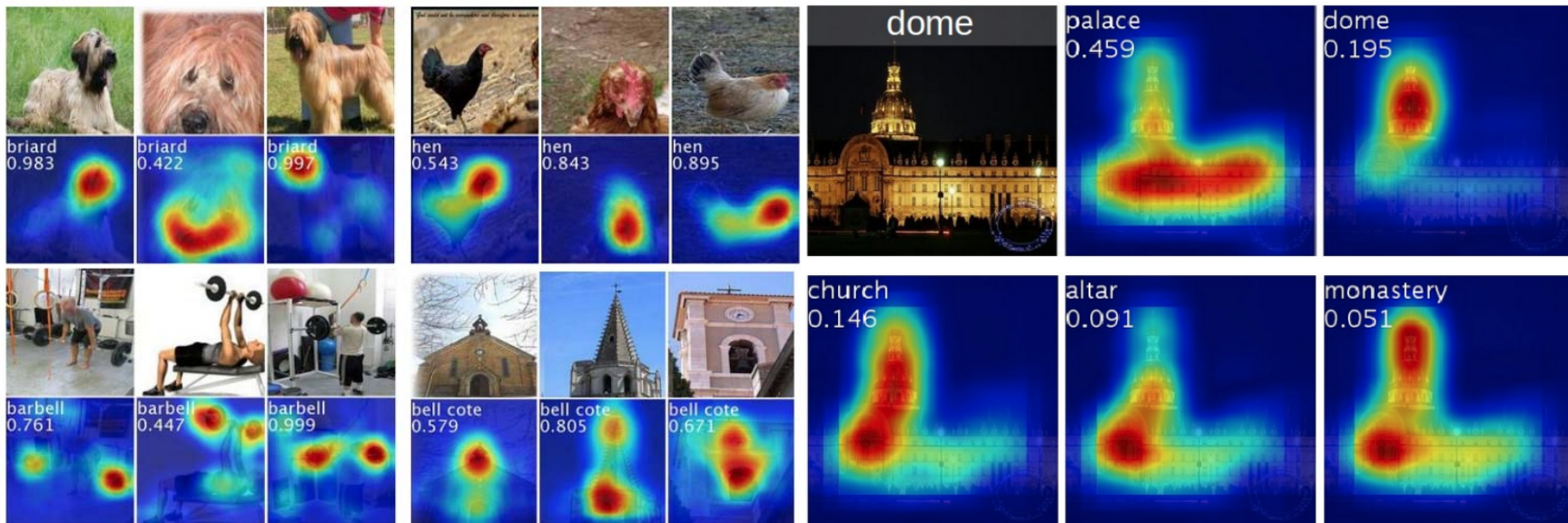
– $CAM = M_c(x, y) = \Sigma(\text{마지막 합성곱 연산 결과}) \times (\text{분류를 위한 가중치 값})$



Class Activation Mapping

물리적 의미

- 입력된 이미지가 특정 클래스로 분류되었을 때, 그 분류 결과에 대한 픽셀의 중요도 혹은 참여도를 의미
- 논문 실험 결과, 인간이 이미지를 판별할 때와 같이 이미지 전체가 아닌 클래스에 해당하는 객체 (Object)에 높은 중요도를 부여하는 것을 확인



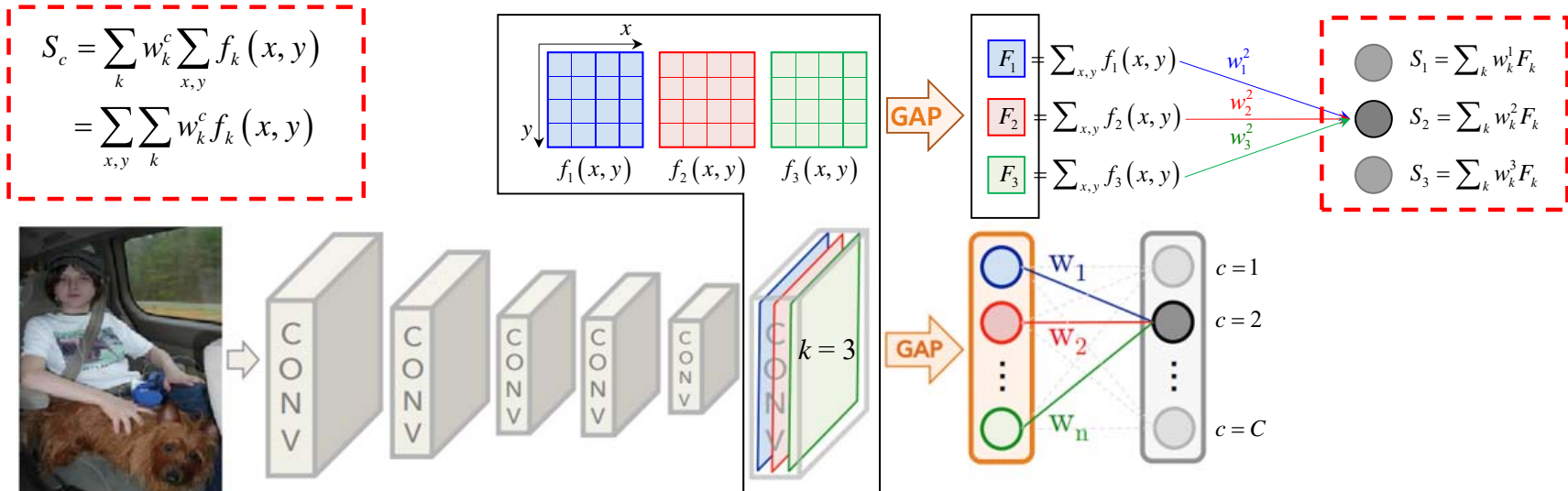
When an Input is Fed to Trained CNN Model...

$f_k(x, y)$: the activation of unit k in the last convolutional layer at spatial location (x, y)

F_k : the result of performing global average pooling ($= \sum_{x,y} f_k(x, y)$)

w_k^c : the weight corresponding to class c for unit k

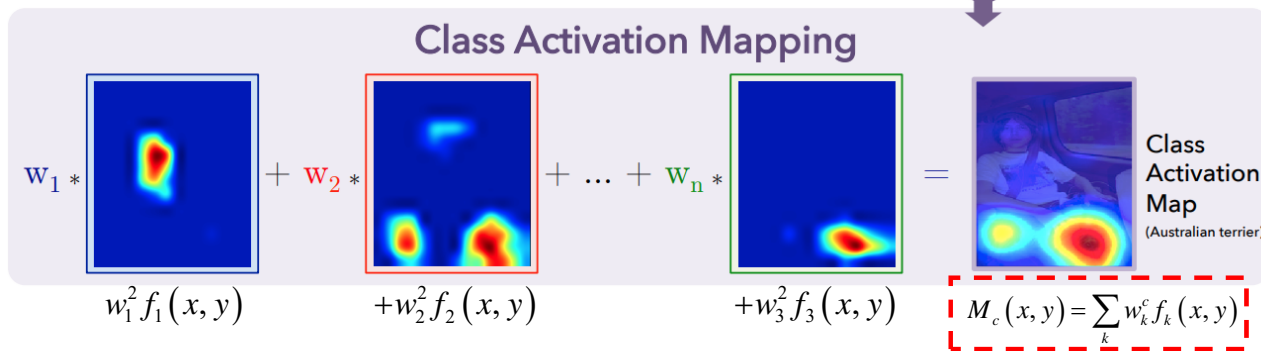
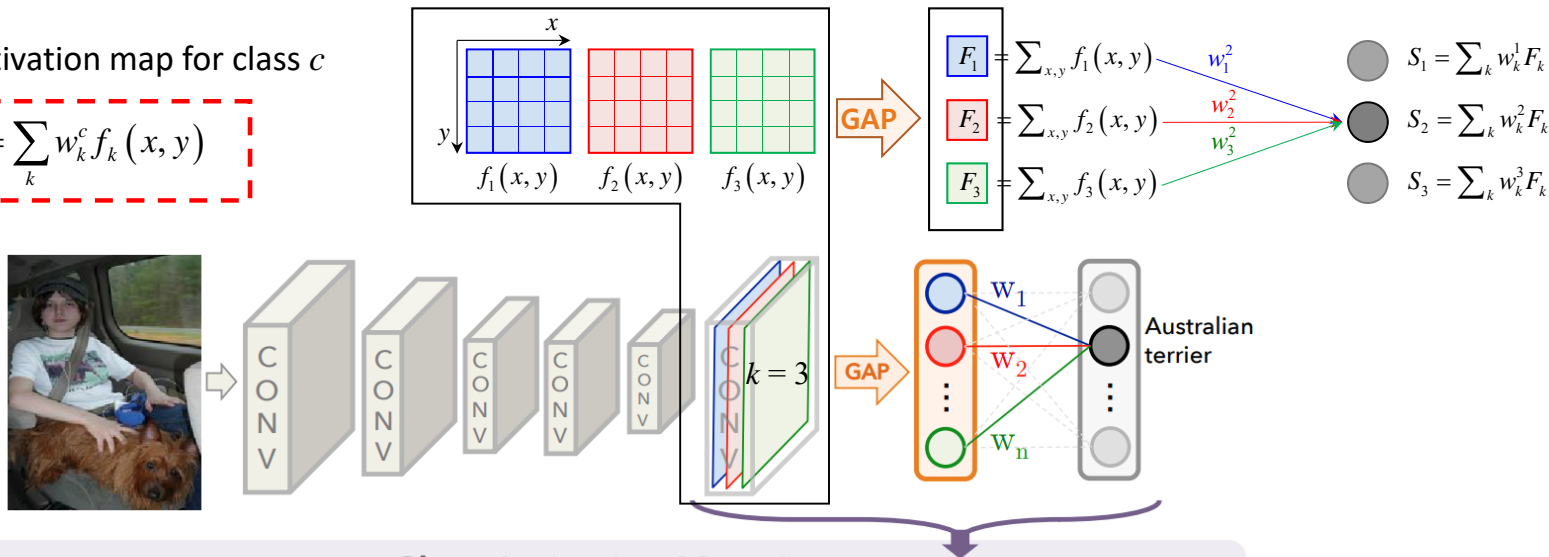
S_c : the input to the softmax ($= \sum_k w_k^c F_k$)



Visualization by CAM is Conducted as Follows!

M_c : the class activation map for class c

$$M_c(x, y) = \sum_k w_k^c f_k(x, y)$$



Demo

CAM 구현 절차

- Step 1: GAP – FC layer로 종료되는 아키텍처 구축
 - CAM 연산에 필요한 두가지 구성요소를 포함한 아키텍처 구축
 - 구성요소 1: 마지막 합성곱 층 뒤 GAP layer
 - 구성요소 2: 분류를 위한 FC layer에 이용되는 가중치 값
- Step 2: 인공지능 신경망 훈련
 - 합성곱 연산에 필요한 필터값 결정
 - CAM 연산에 필요한 가중치 값 결정
- Step 3: CAM 데이터 시각화
 - $f_k(x, y)$: 마지막 합성곱 연산이 수행된 후 k 번째 채널의 x 행, y 열의 픽셀 값
 - F_k : $f_k(x, y)$ 에 GAP 연산이 수행된 후의 값 $(= \sum_{x,y} f_k(x, y))$
 - w_k^c : 클래스 c 를 연산하기 위해 F_k 에 곱해지는 가중치 값
 - S_c : Softmax 함수에 입력되는 값 $(= \sum_k w_k^c F_k)$
 - $CAM = M_c(x, y) = \Sigma(\text{마지막 합성곱 연산 결과}) \times (\text{분류를 위한 가중치 값})$

MNIST Example

- Step 0: Preparation

- 라이브러리 호출

```
1 import tensorflow as tf
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import cv2
```

- MNIST 데이터 로드

```
1 mnist = tf.keras.datasets.mnist
2
3 (train_x, train_y), (test_x, test_y) = mnist.load_data()
4
5 train_x, test_x = train_x/255.0, test_x/255.0
6
7 train_x = train_x.reshape((train_x.shape[0], 28, 28, 1))
8 test_x = test_x.reshape((test_x.shape[0], 28, 28, 1))
9
10 n_train = train_x.shape[0]
11 n_test = test_x.shape[0]
12
13 print ("The number of training images : {}, shape : {}".format(n_train, train_x.shape))
14 print ("The number of testing images : {}, shape : {}".format(n_test, test_x.shape))
```

executed in 260ms, finished 16:10:50 2021-12-16

MNIST Example

- Step 1: GAP – FC layer로 종료되는 아키텍처 구축

```

1 model = tf.keras.models.Sequential([
2     tf.keras.layers.Conv2D(filters = 32,
3                             kernel_size = (3, 3),
4                             activation = 'relu',
5                             padding = 'SAME',
6                             input_shape = (28, 28, 1)),
7     tf.keras.layers.MaxPool2D((2, 2)),
8
9     tf.keras.layers.Conv2D(filters = 64,
10                            kernel_size = (3, 3),
11                            activation = 'relu',
12                            padding = 'SAME',
13                            input_shape = (14, 14, 32)),
14
15     tf.keras.layers.GlobalAveragePooling2D(),
16
17     tf.keras.layers.Dense(units = 10, activation = 'softmax')
18 ])
  
```

Annotations for the code above:

- Lines 2-6: 1st convolutional layer
- Line 7: 1st max-pooling layer
- Lines 9-13: 2nd convolutional layer
- Line 15: Global average pooling layer
- Line 17: Classification with fully-connected layer

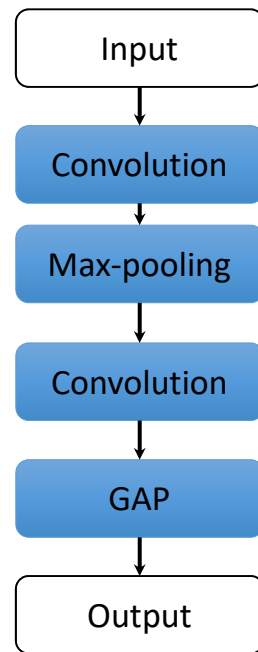
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 32)	320
max_pooling2d (MaxPooling2D)	(None, 14, 14, 32)	0
conv2d_1 (Conv2D)	(None, 14, 14, 64)	18496
global_average_pooling2d (GlobalAveragePooling2D)	(None, 64)	0
dense (Dense)	(None, 10)	650
Total params: 19,466		
Trainable params: 19,466		
Non-trainable params: 0		

executed in 504ms, finished 16:10:51 2021-12-16

```

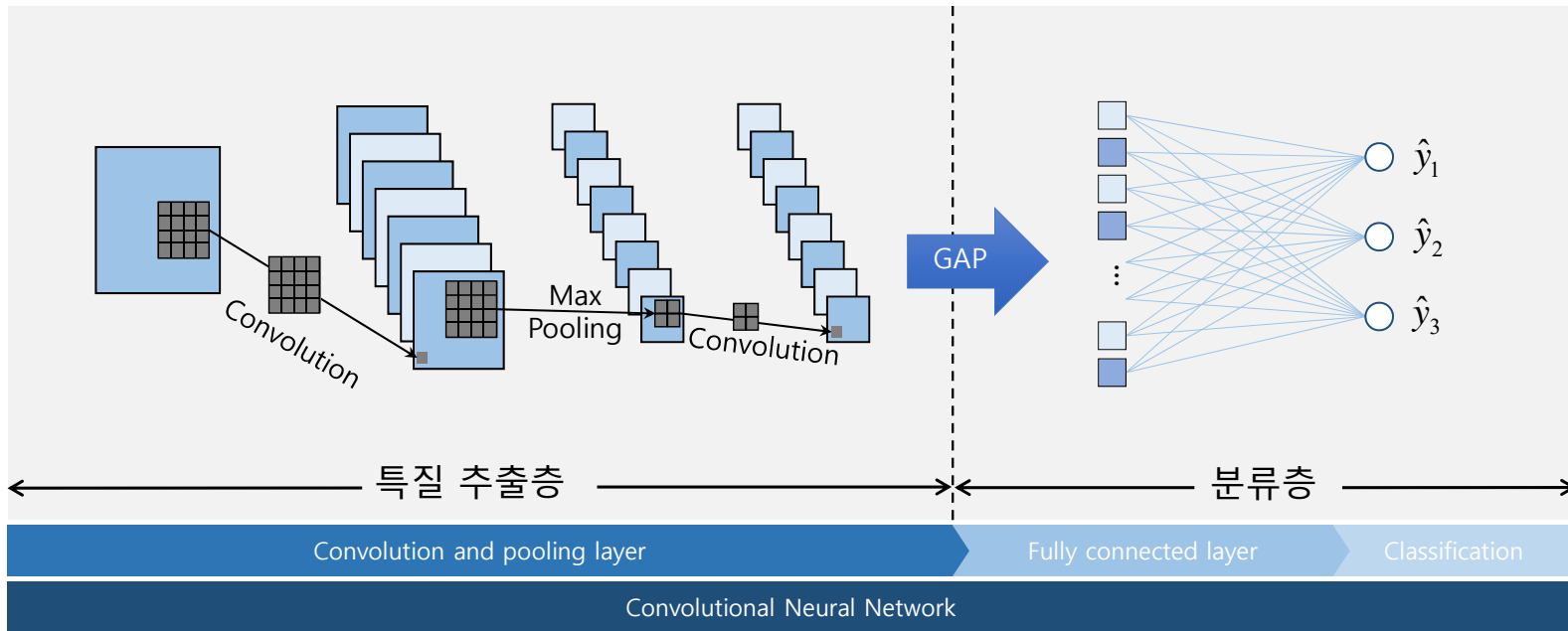
1 model.compile(optimizer = 'adam',
2               loss = 'sparse_categorical_crossentropy',
3               metrics = 'accuracy')
  
```

executed in 7ms, finished 16:10:51 2021-12-16



MNIST Example

- Step 1: GAP – FC layer로 종료되는 아키텍처 구축
 - Conv1 – Max. pool – Conv2 – GAP – Classification



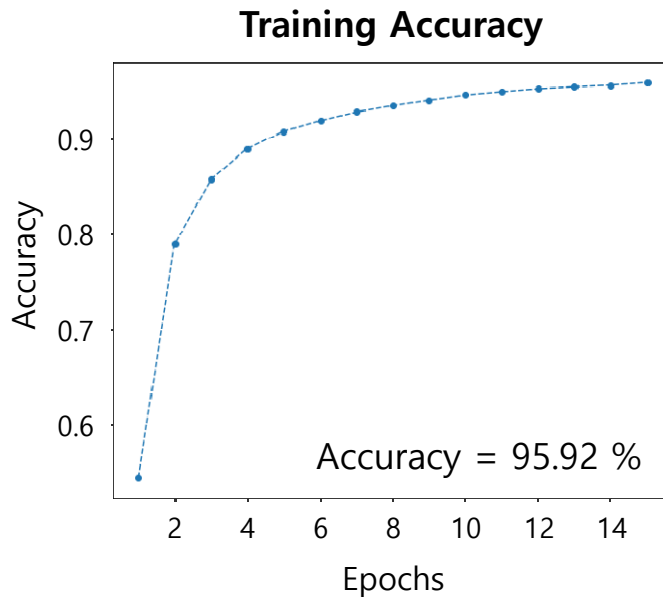
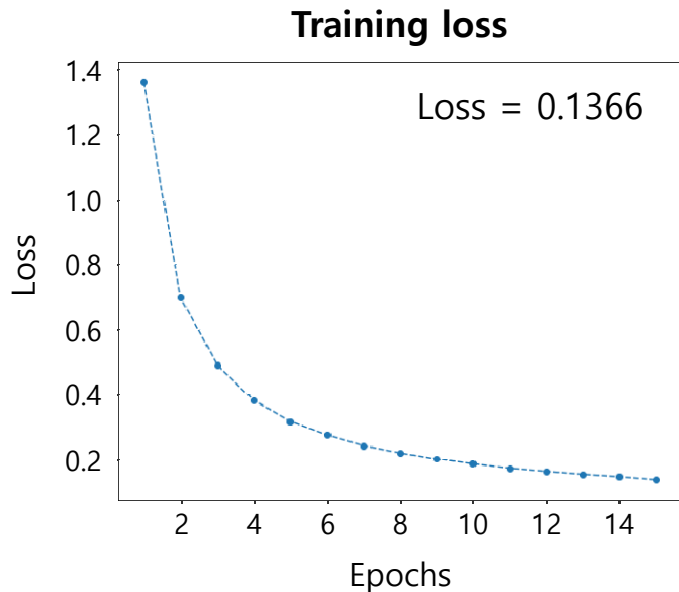
MNIST Example

- Step 2: 인공지능 모델 훈련

```
1 model.fit(train_x, train_y, epochs = 15) → Train
```

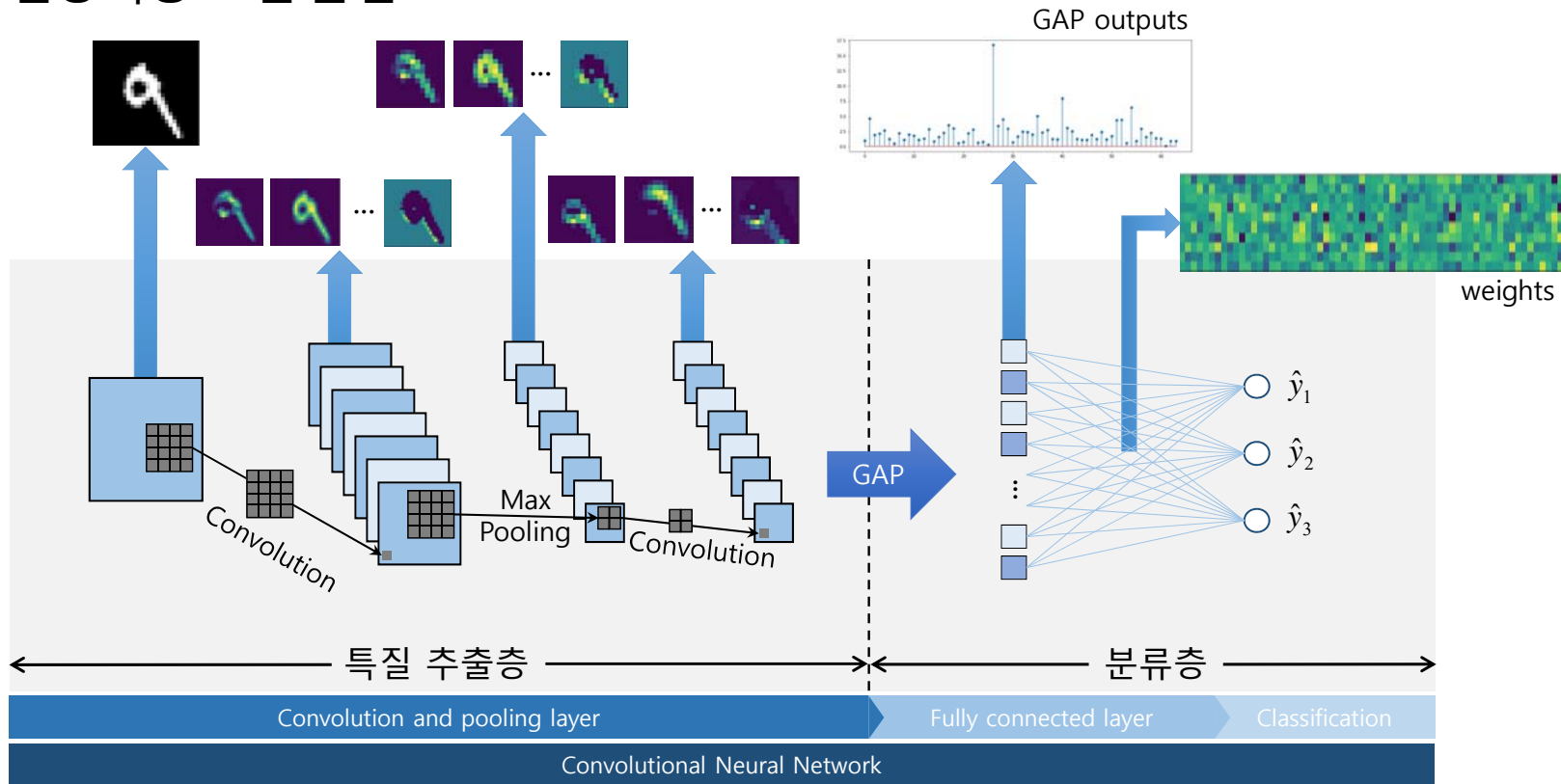
executed in 1m 33.5s, finished 16:12:25 2021-12-16

- 결과



MNIST Example

- Step 2: 인공지능 모델 훈련



MNIST Example

• Step 3: CAM 시각화

– CAM 연산

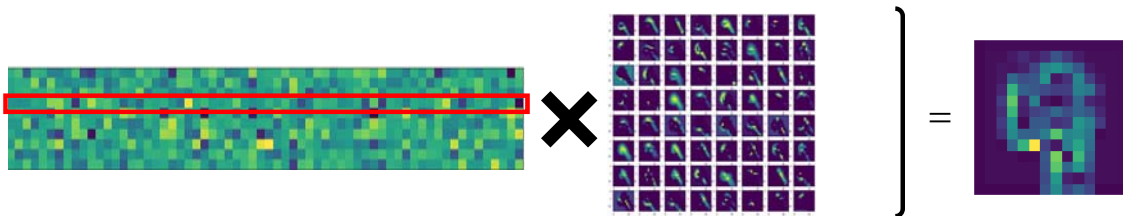
```

1 # get max pooling layer and fully connected layer
2 conv_layer = model.get_layer(index = 2) → 마지막 합성곱 연산 결과 추출:  $f_k(x, y)$ 
3 fc_layer = model.layers[4].get_weights()[0] → FC layer 가중치 추출:  $w_k^c$ 
4
5 # Class activation map
6 my_map = tf.matmul(conv_layer.output, fc_layer) → CAM 데이터 계산:  $M_c(x, y) = \sum_k w_k^c f_k(x, y)$ 
7 CAM = tf.keras.Model(inputs = model.inputs, outputs = my_map)

```

executed in 19ms, finished 16:12:26 2021-12-16

- $f_k(x, y)$: 마지막 합성곱 연산이 수행된 후 k 번째 채널의 x 행, y 열의 픽셀 값
- F_k : $f_k(x, y)$ 에 GAP 연산이 수행된 후의 값 $(= \sum_{x,y} f_k(x, y))$
- w_k^c : 클래스 c 를 연산하기 위해 F_k 에 곱해지는 가중치 값
- S_c : Softmax 함수에 입력되는 값 $(= \sum_k w_k^c F_k)$
- $\text{CAM} = M_c(x, y) = \sum_k w_k^c f_k(x, y)$

$$M_4(x, y) = \sum_k w_k^4 f_k(x, y) = \sum \left[\text{Feature Map} \times \text{Weights Matrix} \right] = \text{CAM Map}$$


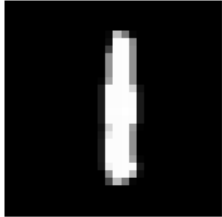
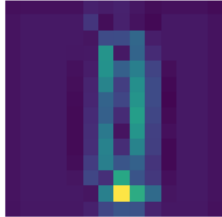
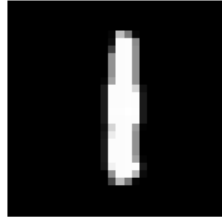
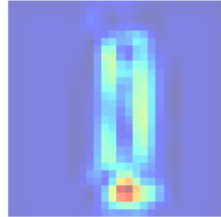
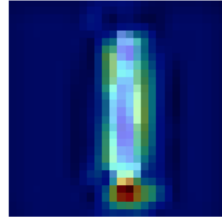

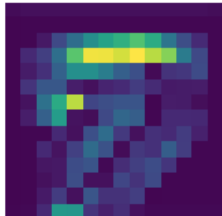

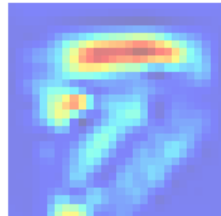
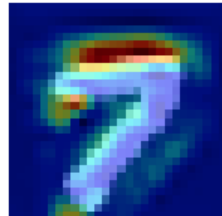

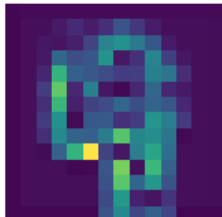

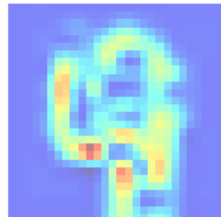
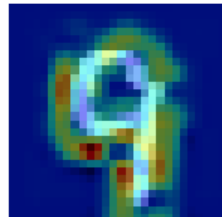
MNIST Example

- Step 3: CAM 시각화
 - CAM 데이터 시각화

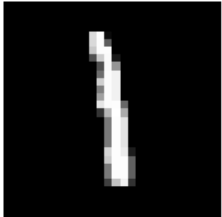
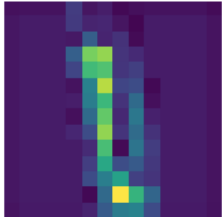
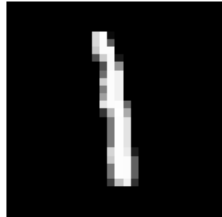
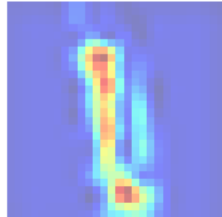
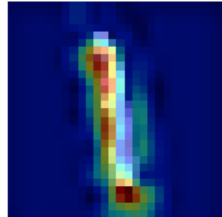

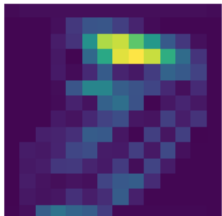

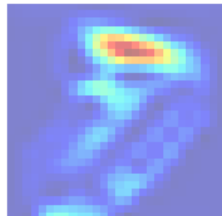
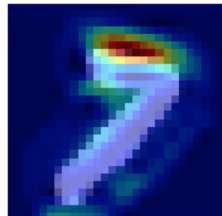

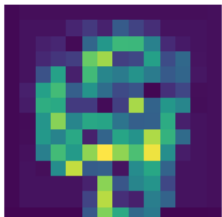

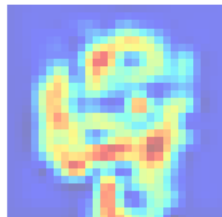
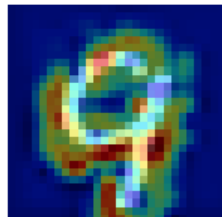
```
1 test_idx = [1005]
2 test_image = test_x[test_idx]
3
4 pred = np.argmax(model.predict(test_image), axis = 1)
5 predCAM = CAM.predict(test_image)
6
7 attention = predCAM[:, :, :, pred]
8 attention = np.abs(np.reshape(attention, (14, 14))) —————> Raw CAM 데이터
9
10 resized_attention = cv2.resize(attention,
11                               (28, 28), —————> 원본 입력 이미지 크기의 CAM 데이터
12                               interpolation = cv2.INTER_CUBIC)
13
14 resized_test_x = cv2.resize(test_image.reshape(28, 28),
15                             (28, 28), —————> 원본 이미지
16                             interpolation = cv2.INTER_CUBIC)
```

executed in 253ms, finished 16:13:47 2021-12-16

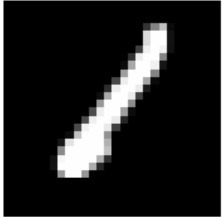
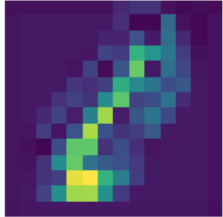
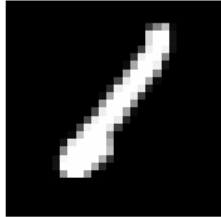
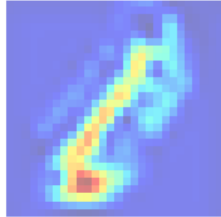
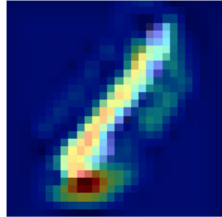
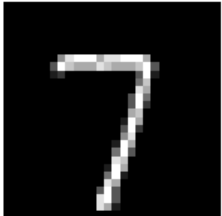
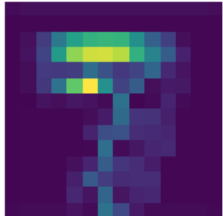
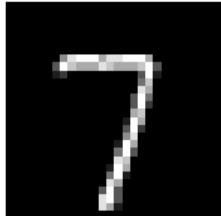
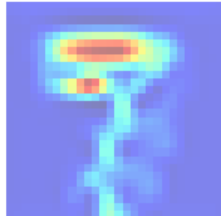
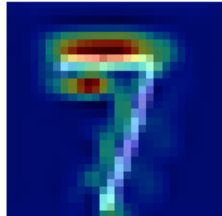
MNIST: Results of Example

	Original input	Original CAM	Resized input	Resized CAM	Resized input + CAM
Class '1'					
Class '7'					
Class '9'					

MNIST: Results of Example

	Original input	Original CAM	Resized input	Resized CAM	Resized input + CAM
Class '1'					
Class '7'					
Class '9'					

MNIST: Results of Example

	Original input	Original CAM	Resized input	Resized CAM	Resized input + CAM
Class '1'					
Class '7'					
Class '9'	