



Class Activation Map (CAM)

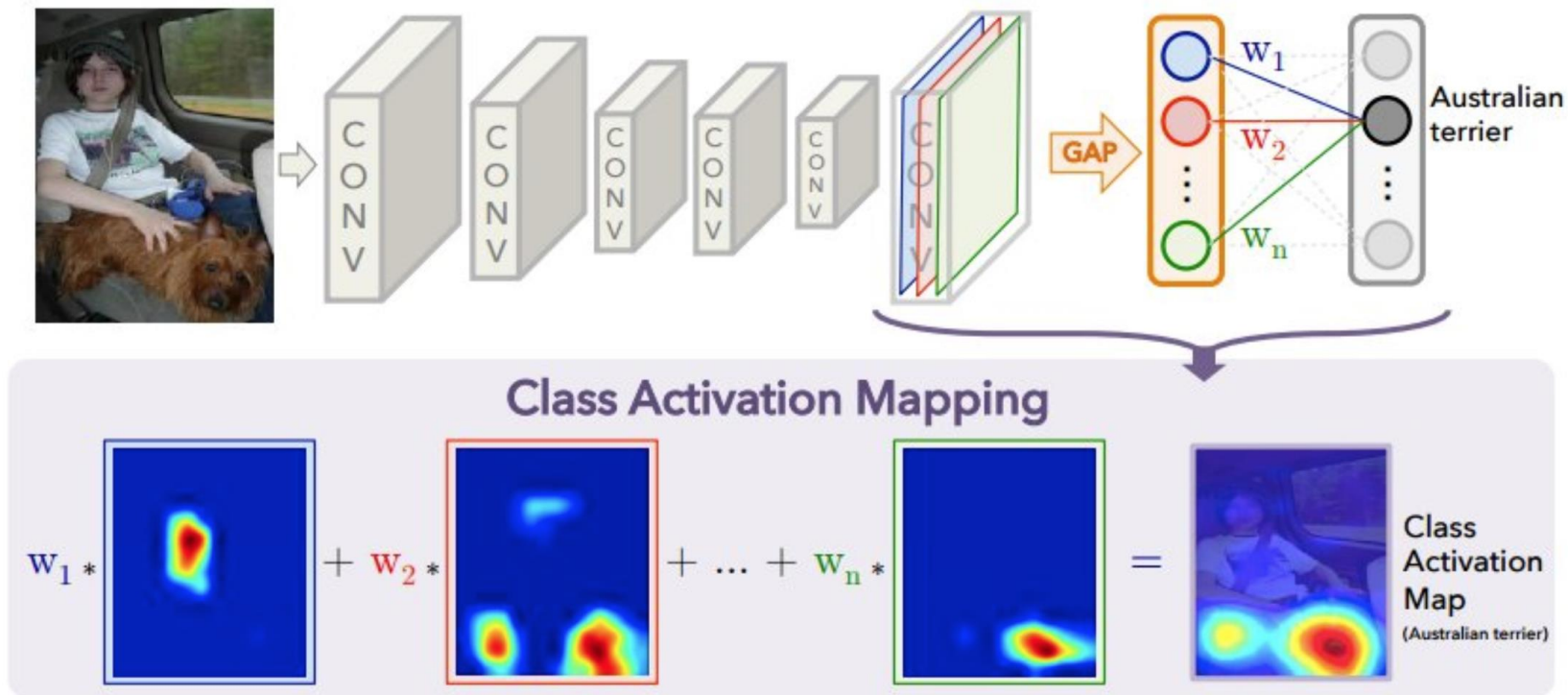
Industrial AI Lab.
Prof. Seungchul Lee

Issues on CNN (or Deep Learning)

- Deep learning performs well comparing with any other existing algorithms
- But works as a black box
 - A classification result is simply returned without knowing how the classification results are derived → little interpretability
- When we visually identify images, we do not look at the whole image
- Instead, we intuitively focus on the most important parts of the image
- When CNN weights are optimized, the more important parts are given higher weights
- Class activation map (CAM)
 - We can determine which parts of the image the model is focusing on, based on the learned weights
 - Highlighting the importance of the image region to the prediction

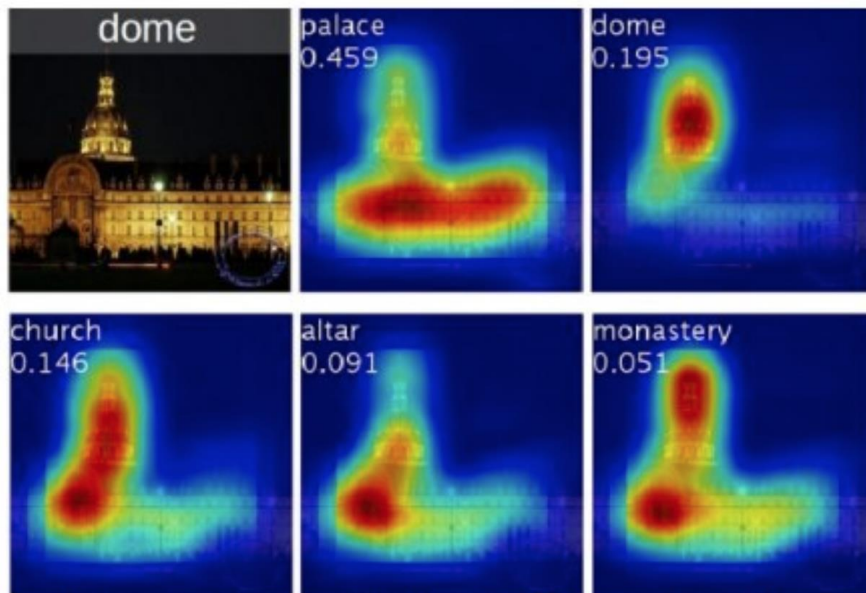
Visualizing Convolutional Neural Networks

- Class Activation Maps (CAMs)
- A class activation map (CAM) for a given class highlights the image regions used by the CNN to identify that class



Visualizing Convolutional Neural Networks

- Class Activation Maps (CAMs)
- A class activation map (CAM) for a given class highlights the image regions used by the CNN to identify that class

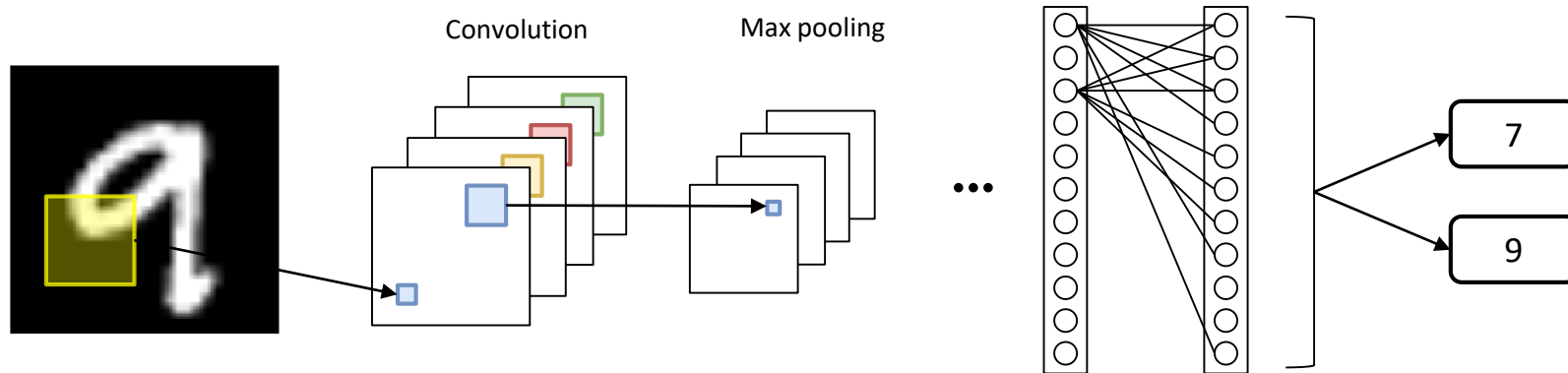


Class activation maps of top 5 predictions



Class activation maps for one object class

Fully Connected Layer



Convolution and pooling layers

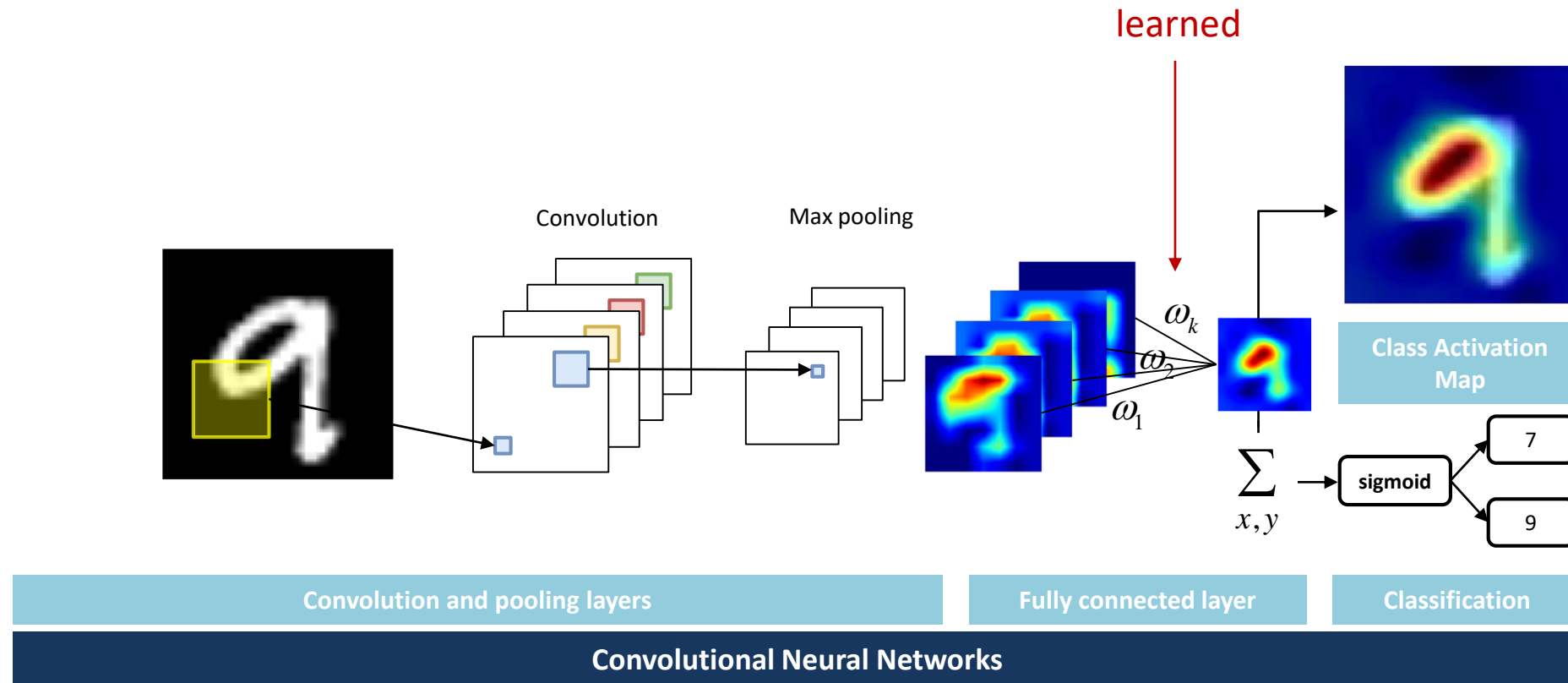
Fully connected layer

Classification

Convolutional Neural Networks

Global Average Pooling

- Class Activation Map (or Attention)

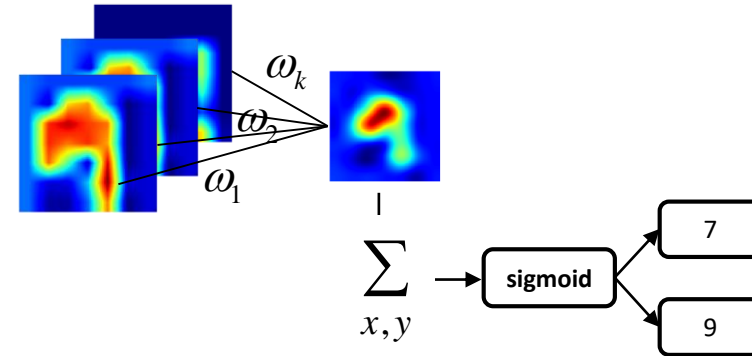
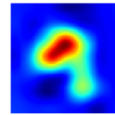


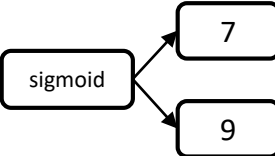
Global Average Pooling Implementation

```
## global average pooling
SUM = tf.zeros([1,7,7,1])
for i in range(int(weights['w'].shape[0])):
    SUM = tf.add(weights['w'][i]*tf.reshape(maxp2[:, :, :, i], (-1,7,7,1)), SUM)

attention = tf.reduce_sum(SUM, axis = (3))
output = tf.reduce_sum(attention, axis = (1,2))
output = tf.nn.sigmoid(output)
output = tf.stack(((1-output), output),1)
```

$$S_c = \sum_k \omega_k^c \sum_{x,y} f_k(x,y) = \sum_{x,y} \underbrace{\sum_k \omega_k^c f_k(x,y)}$$



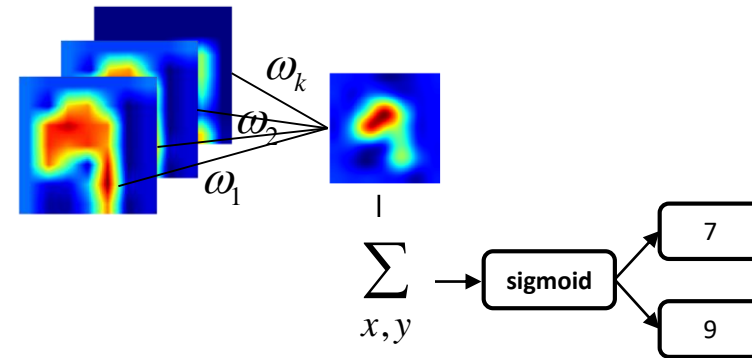
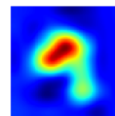
$$P_c = \frac{\exp(S_c)}{\sum_c \exp(S_c)}$$


Global Average Pooling Implementation (Better Way)

```
## global average pooling  
avg = tf.reduce_mean(maxp2, axis = (1,2))  
output = tf.matmul(avg, weights['output'])
```

```
maps, pred = net(x, weights, biases)  
loss = tf.nn.softmax_cross_entropy_with_logits(labels = y, logits = pred)  
loss = tf.reduce_mean(loss)
```

$$S_c = \sum_k \omega_k^c \sum_{x,y} f_k(x,y) = \sum_{x,y} \underbrace{\sum_k \omega_k^c f_k(x,y)}$$



$$P_c = \frac{\exp(S_c)}{\sum_c \exp(S_c)}$$

Example: MNIST

