# Face recognition

# Face verification vs. face recognition

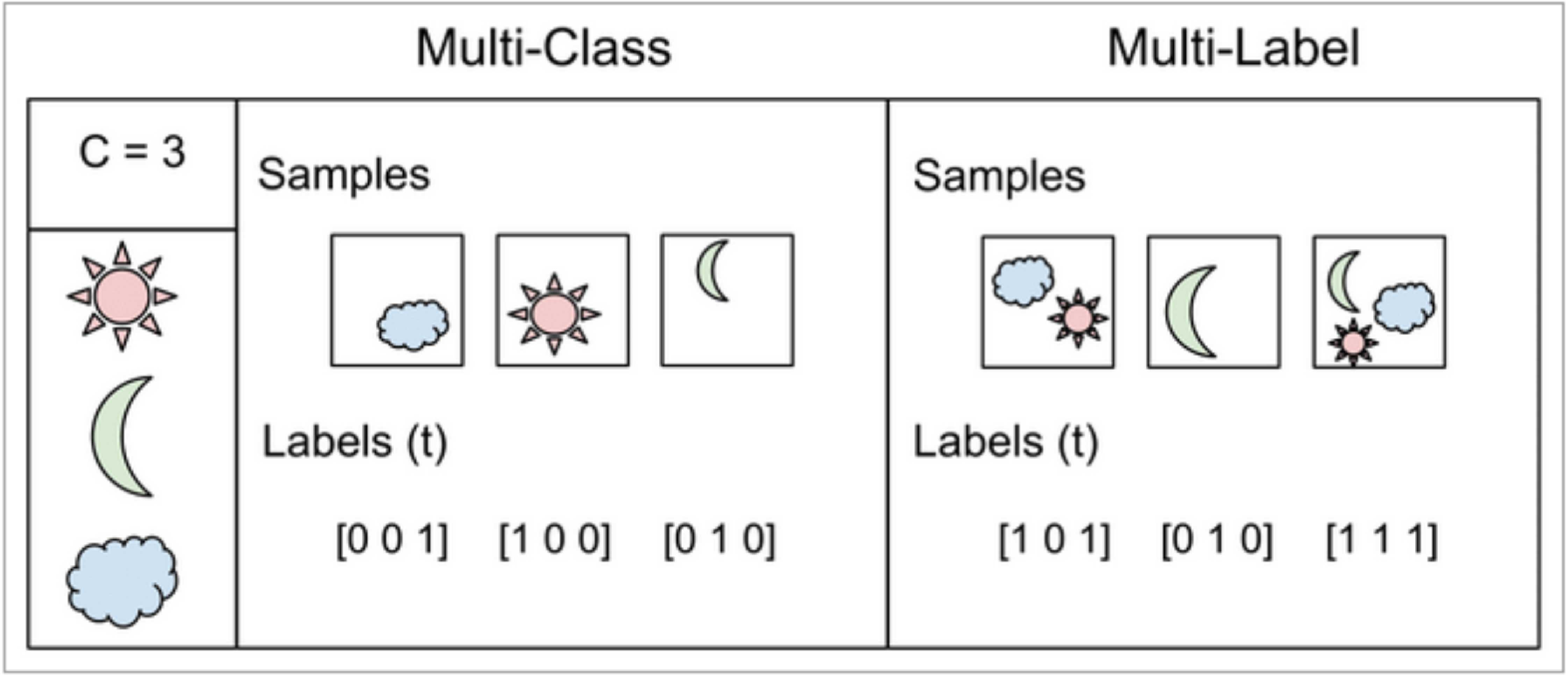
## Verification (Binary Classification)

- Input image, name/ID
- Output whether the input image is that of the claimed person

## Recognition (Multi Class Classification)

- Has a database of K persons
- Get an input image
- ID Output ID if the image is any of the K persons (or "not recognized")

# Multi Class Classification vs Multi Label Classification




- ref: scikit-learn

# One-shot learning: intro

## 하나의 example만으로 classifier를 학습하는 연구.

few-shot learning으로 추상되며,
보통 meta learning 방법론을 참조한다.

강의에서는 데이터들 간의 **유사도 함수를 학습**하는 형태의 방법론을 보였다.

# One-shot learning: Meta Learning

**Meta refers to a level above.**
목표과제(labeled data)와는 간접적일 수 있는 목표함수(meta)를 학습하는 방법론
(한국웹에서는 보통 학습/해결 방법을 학습하는 방법 이라고 소개한다.)
일반적인 ML에서는 모델의 결과와, 데이터의 레이블에 기반한
loss(cost) function을 만들어 최소화 시키는 방법으로 학습하는데

$$f_{loss}(y, \hat{y})$$

Meta leaning 에서는 보통 문제를 해결하기 위한 다른 함수(e.g. 유사도)를 정의하고
해당 **함수의 기능을 최대화**하는 방식으로 학습한 후
해당 함수를 이용하며 문제를 해결한다.

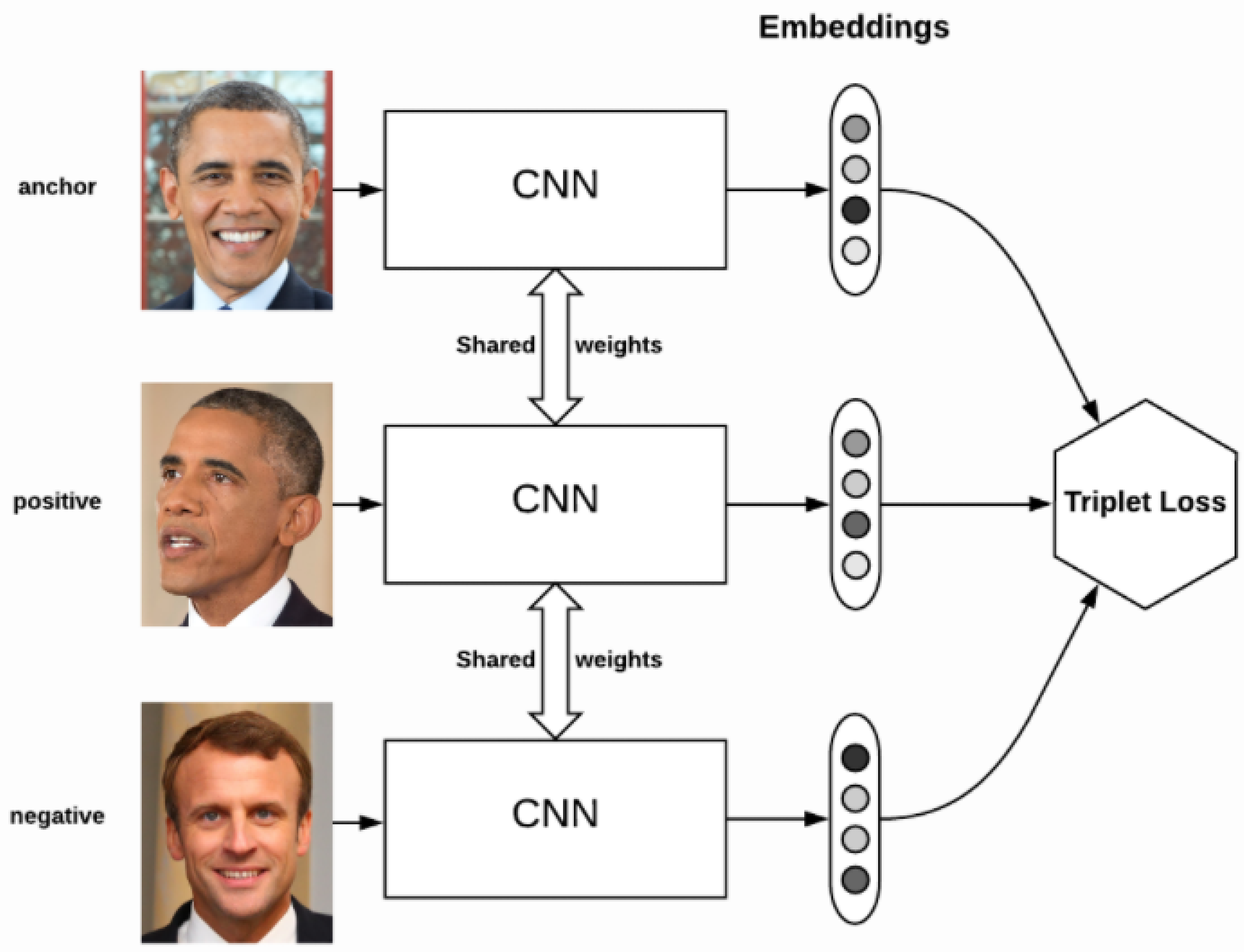
If $x^{(i)}, x^{(j)}$ are the same person, $\|f(x^{(i)}) - f(x^{(j)})\|^2$ is small.

If $x^{(i)}, x^{(j)}$ are different persons, $\|f(x^{(i)}) - f(x^{(j)})\|^2$ is large.

# One-shot learning: refs

- refs
  - meta learning 영문 wiki
  - meta learning 영문 blog
  - Supervised vs one-shot 영문
  - few-shot 영문
  - few-shot 한글(카카오)

# Triplets loss: intro



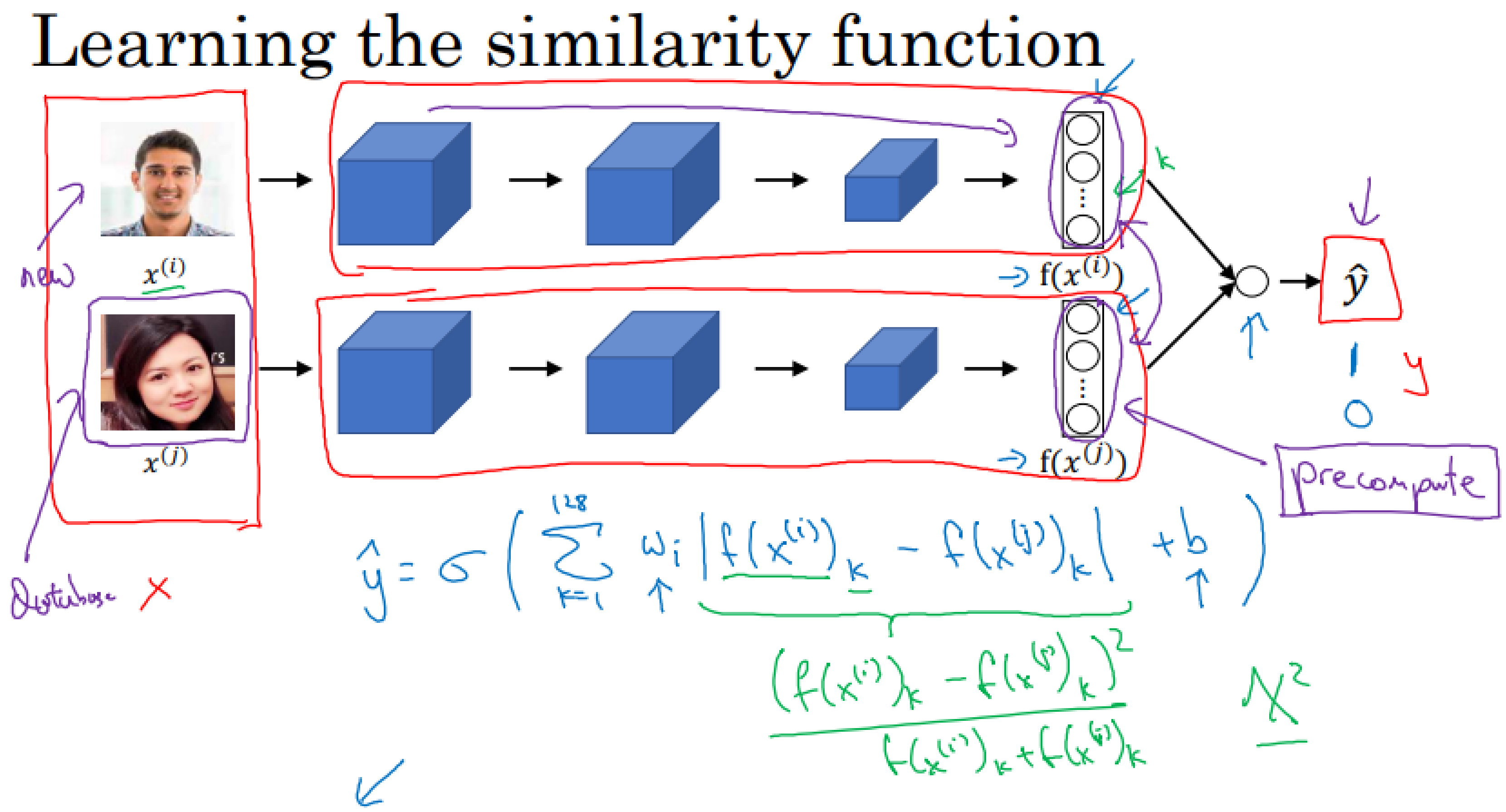

*Triplet loss on two positive faces (Obama) and one negative face (Macron)*

# Triplets loss: code

```python
def batch_all_triplet_loss(labels, embeddings, margin, squared=False):
    """Build the triplet loss over a batch of embeddings.

    We generate all the valid triplets and average the loss over the positive ones.

    Args:
        labels: labels of the batch, of size (batch_size,)
        embeddings: tensor of shape (batch_size, embed_dim)
        margin: margin for triplet loss
        squared: Boolean. If true, output is the pairwise squared euclidean distance matrix.
                 If false, output is the pairwise euclidean distance matrix.

    Returns:
        triplet_loss: scalar tensor containing the triplet loss
    """
    # Get the pairwise distance matrix
    pairwise_dist = _pairwise_distances(embeddings, squared=squared)

    anchor_positive_dist = tf.expand_dims(pairwise_dist, 2)
    anchor_negative_dist = tf.expand_dims(pairwise_dist, 1)

    # Compute a 3D tensor of size (batch_size, batch_size, batch_size)
    # triplet_loss[i, j, k] will contain the triplet loss of anchor=i, positive=j, negative=k
    # Uses broadcasting where the 1st argument has shape (batch_size, batch_size, 1)
    # and the 2nd (batch_size, 1, batch_size)
    triplet_loss = anchor_positive_dist - anchor_negative_dist + margin
```

- ref: https://omoindrot.github.io/triplet-loss

# Face Verification in the lecture




[Taigman et. al., 2014. DeepFace closing the gap to human level performance]

Andrew Ng

# Face Verification in the lecture

## Face verification supervised learning

| $x$ | $y$ | |
|---|---|---|
| (pair 1) | 1 | "Same" |
| (pair 2) | 0 | "different" |
| (pair 3) | 0 | |
| (pair 4) | 1 | |

[Taigman et. al., 2014. DeepFace closing the gap to human level performance]

Andrew Ng

# Neural Style Transfer

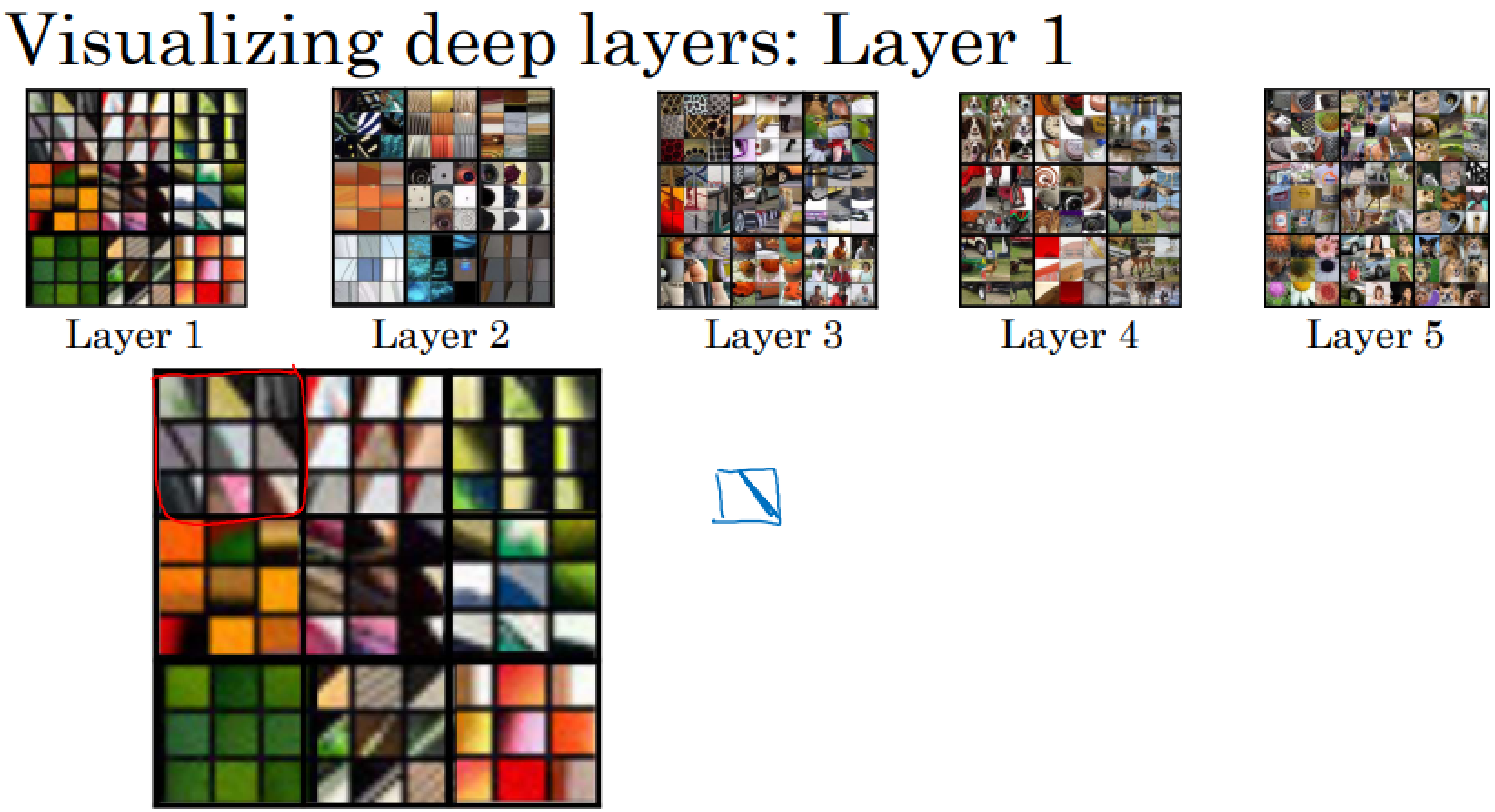
- Visualize CNN
- Neural Style Transfer

# Visualize CNN

# Visualize CNN: layer 1

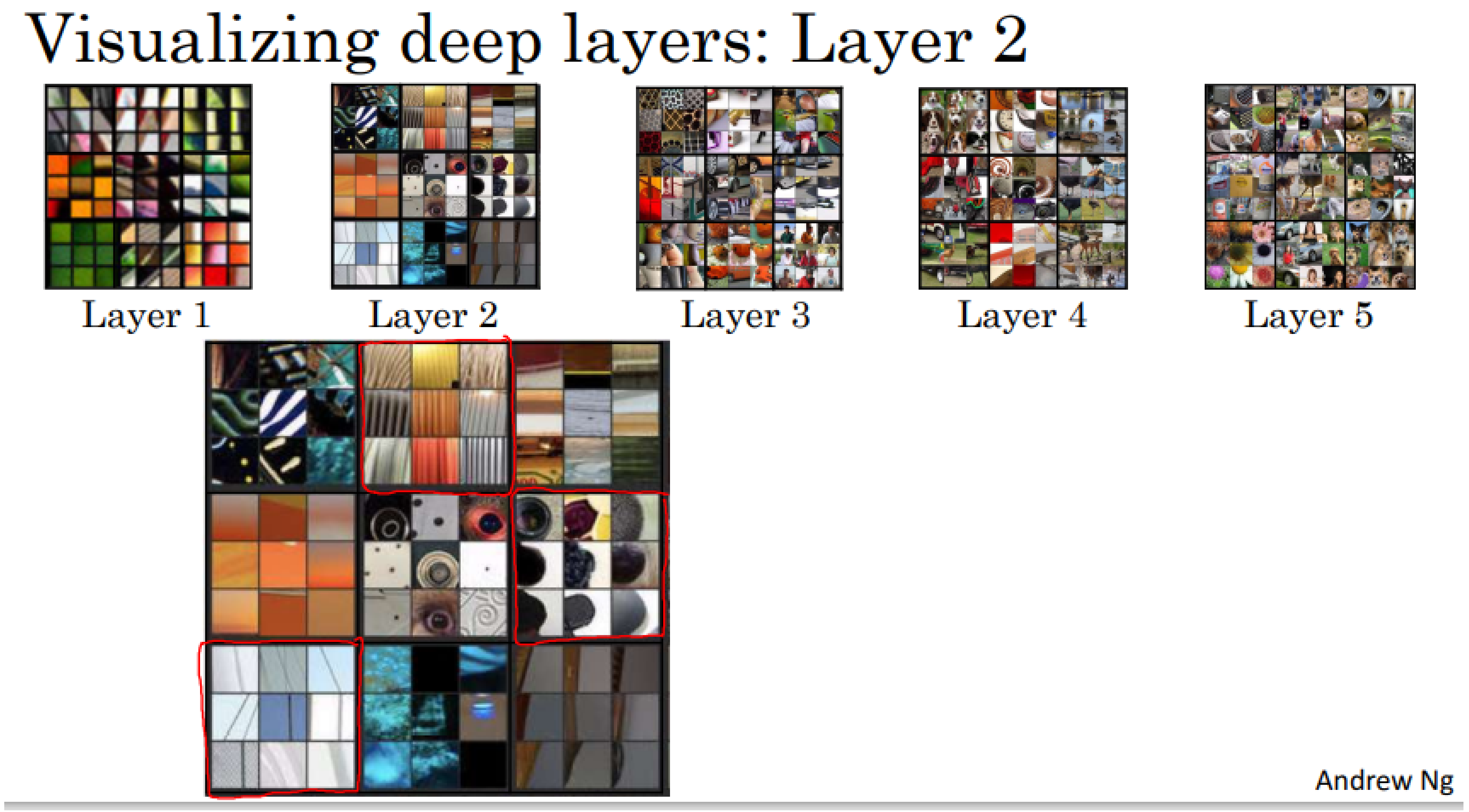
각 layer에서 unit의 최대로 activate 되게 하는 fetch를 보인 것 입니다.

- fetch는 dataset에서 찾습니다.




Andrew Ng

# Visualize CNN: layer 2

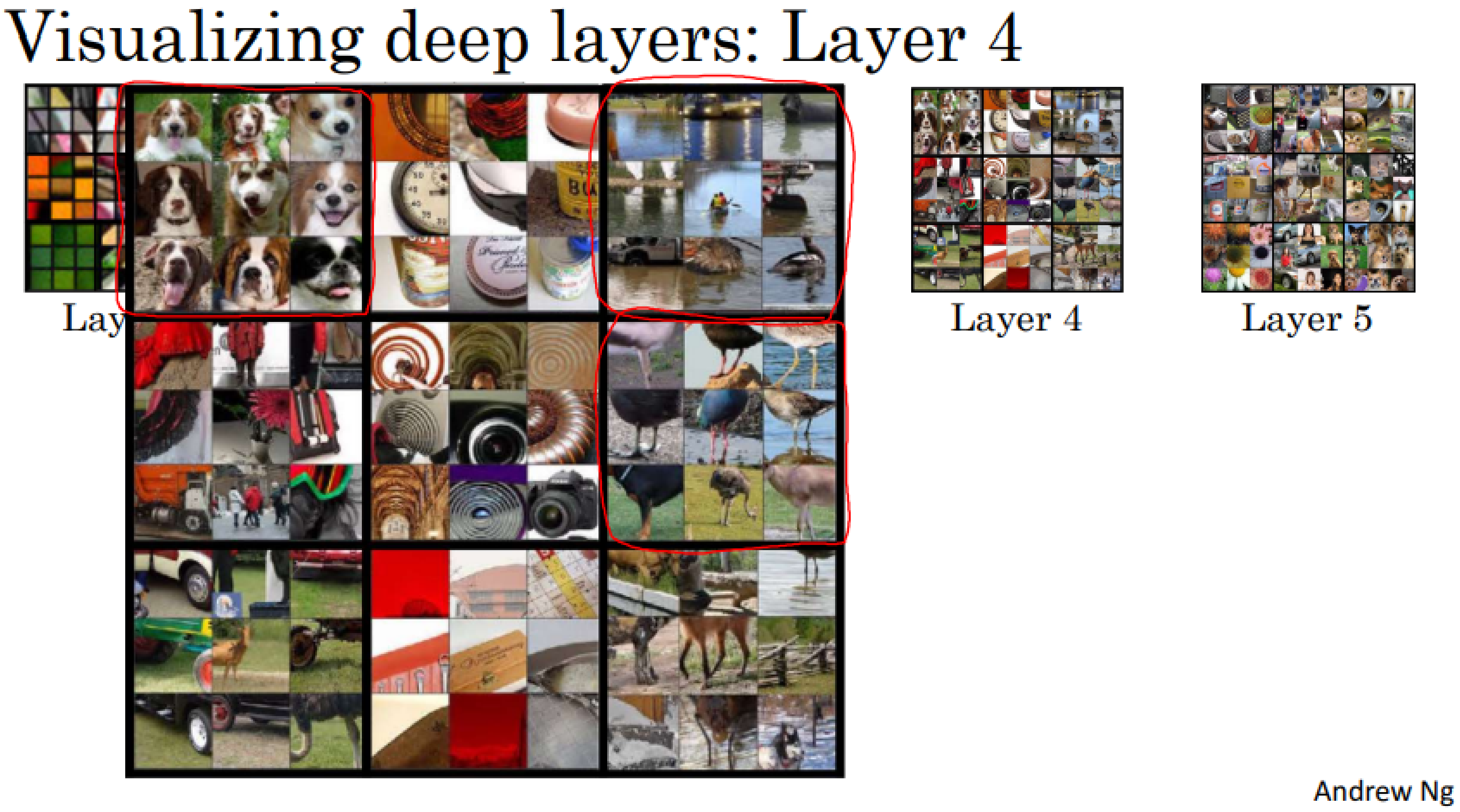


# Visualize CNN: layer 3

Visualizing deep layers: Layer 3

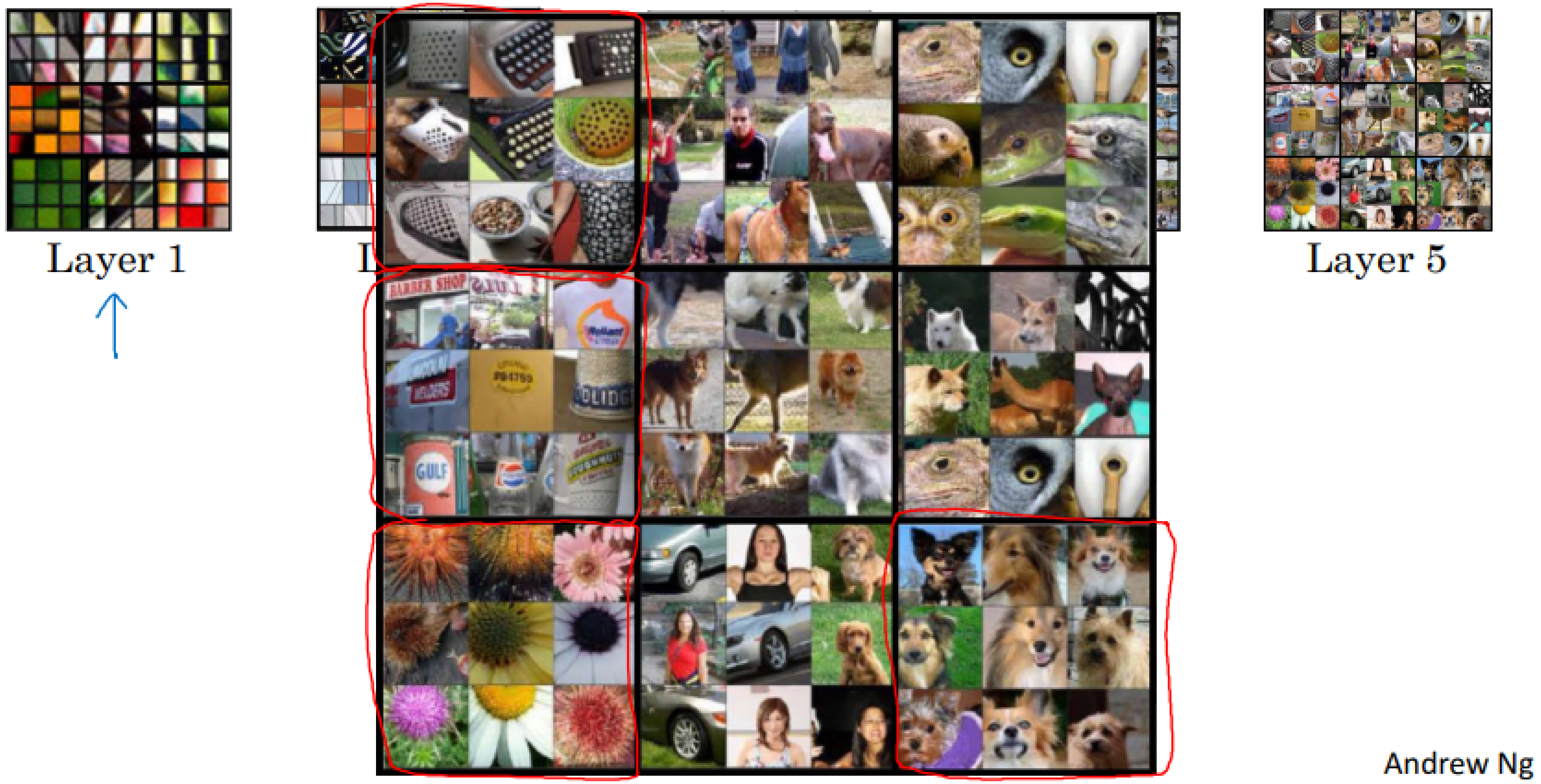

Andrew Ng

# Visualize CNN: layer 4



# Visualize CNN: layer 5

## Visualizing deep layers: Layer 5



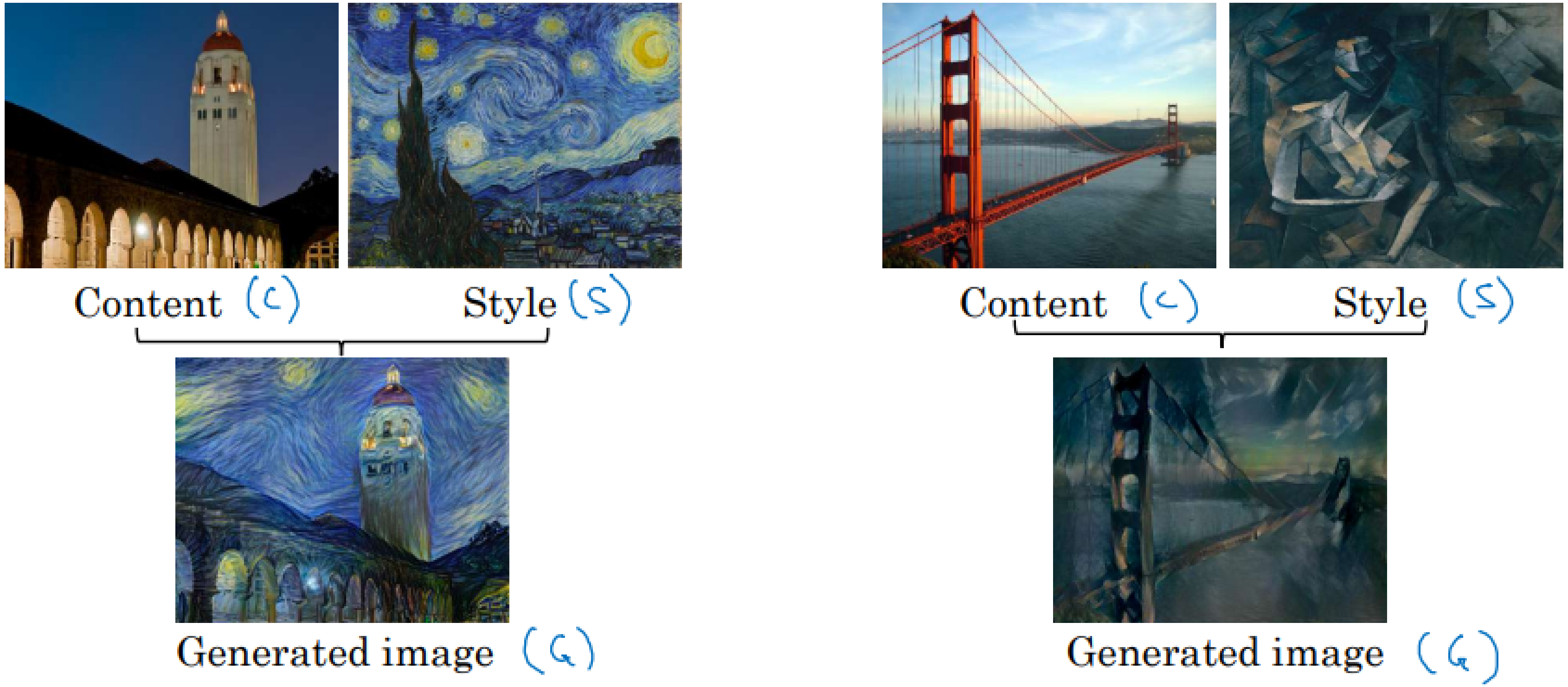
# Visualize CNN: other method

- Filters의 weight를 그대로 출력한다.
- Image를 the filters 에 그대로 적용하여 출력한다.

- https://www.youtube.com/watch?v=ho6JXE3EbZ8

# Neural Style Transfer: cost function

## Neural style transfer




[Images generated by Justin Johnson]

Andrew Ng

$$J(G) = \alpha\, J_{content}(C, G) + \beta\, J_{style}(S, G)$$

# Neural Style Transfer: generate

## Find the generated image G

1. Initiate G randomly

   G: 100×100×3

   RGB

2. Use gradient descent to minimize $J(G)$

$$G := G - \frac{\partial}{\partial G} J(G)$$

[Gatys et al., 2015. A neural algorithm of artistic style]

Andrew Ng

$$J(S, C, G)$$

에서 S,C 를 고정하고, gradient descent로 G를 업데이트

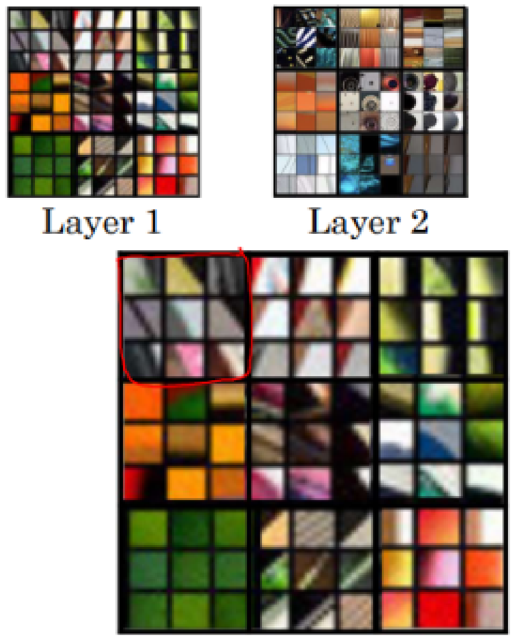
# Neural Style Transfer: content cost function

## Content cost function

$$J(G) = \alpha\, J_{content}(C, G) + \beta\, J_{style}(S, G)$$

- Say you use hidden layer $l$ to compute content cost.
- Use pre-trained ConvNet. (E.g., VGG network)
- Let $a^{[l](C)}$ and $a^{[l](G)}$ be the activation of layer $l$ on the images
- If $a^{[l](C)}$ and $a^{[l](G)}$ are similar, both images have similar content

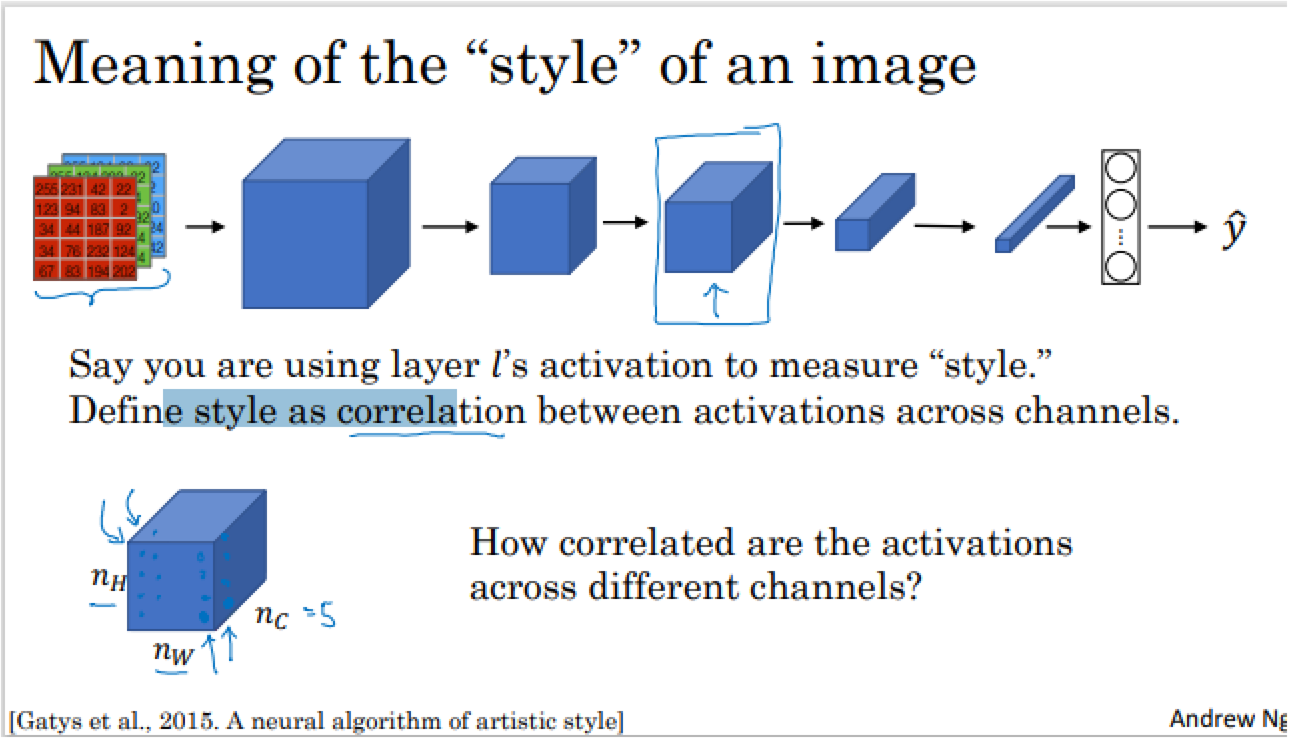
$$J_{content}(C, G) = \frac{1}{2} \underbrace{\| a^{[l](C)} - a^{[l](G)} \|^2}$$

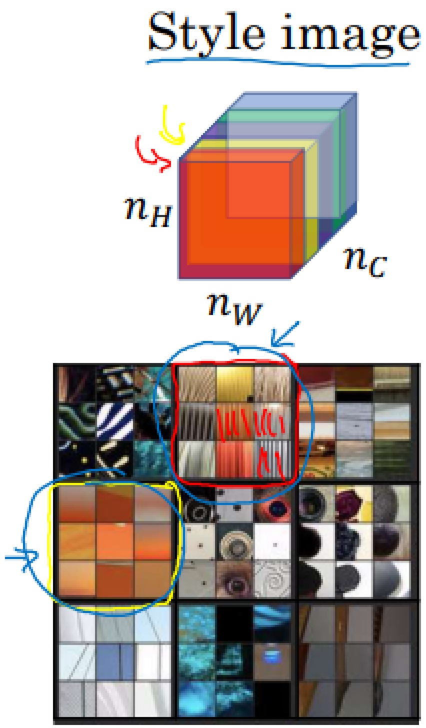



[Gatys et al., 2015. A neural algorithm of artistic style]

Andrew Ng

# Neural Style Transfer: style cost function



## Activation들의 조합/곱은(k, k`)

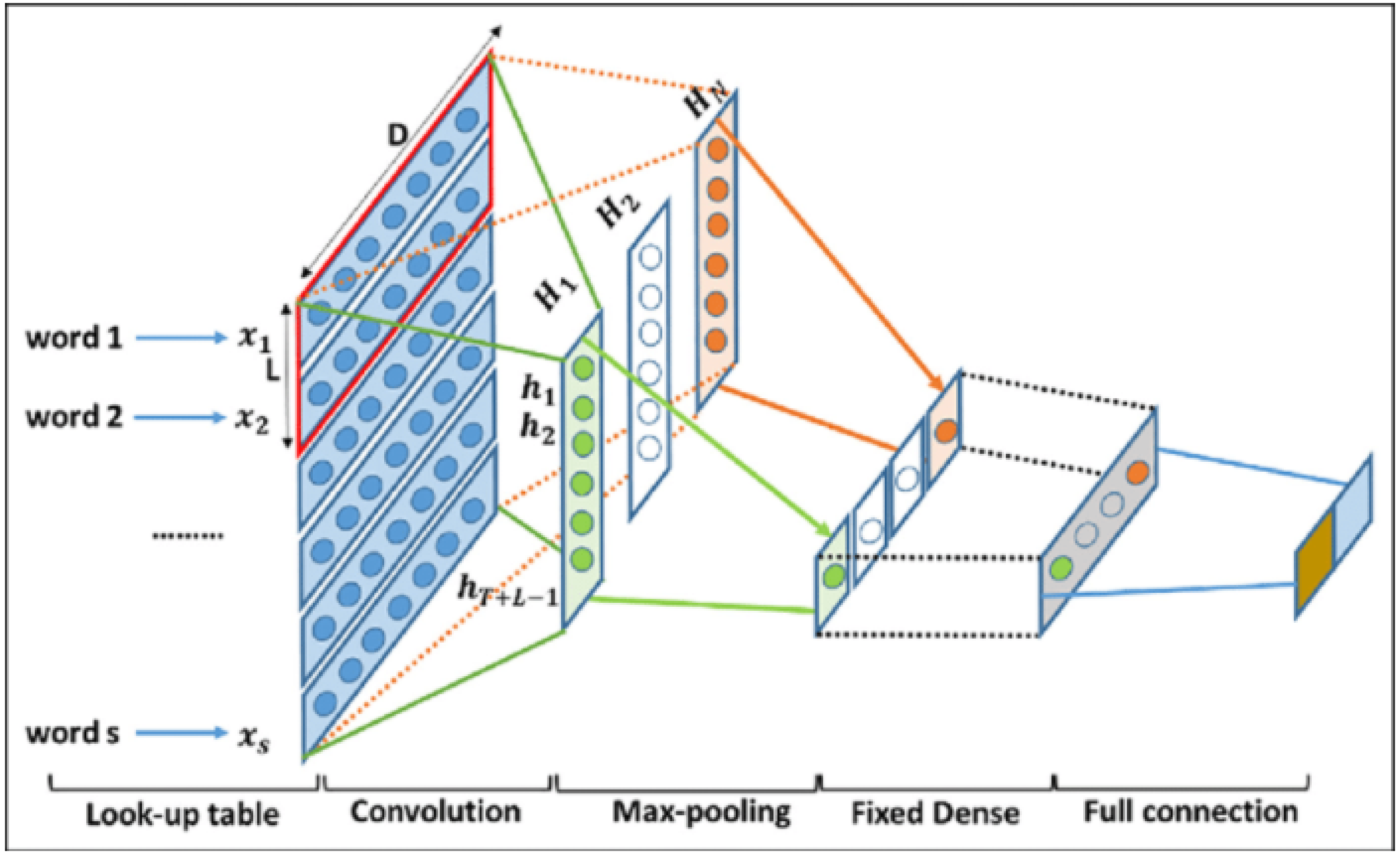
- 그림의 스트로크와 같은
  특징(색상, 표현, 패턴 또는 이들의 조합)을 대표 할 것이다.
- Style cost는 generated image와 style image 간에 이 특징들의 차이를 나타낸다.
- Style cost를 최소화 하는 것은
  igenerated image와 style image 간의 특징 차이를 최소화 하는 것!

The style cost function should be:

$$J_{style}^{[l]}(S,G) = \frac{1}{(2n_H^{[l]}n_W^{[l]}n_C^{[l]})^2} \sum_k \sum_{k'} \left(G_{kk'}^{[l](S)} - G_{kk'}^{[l](G)}\right)^2$$

# 1D and 3D Generalizations

## Conv 1d




Standard CNN on text classification.

- ref https://arxiv.org/abs/1408.5882

# Conv 3d

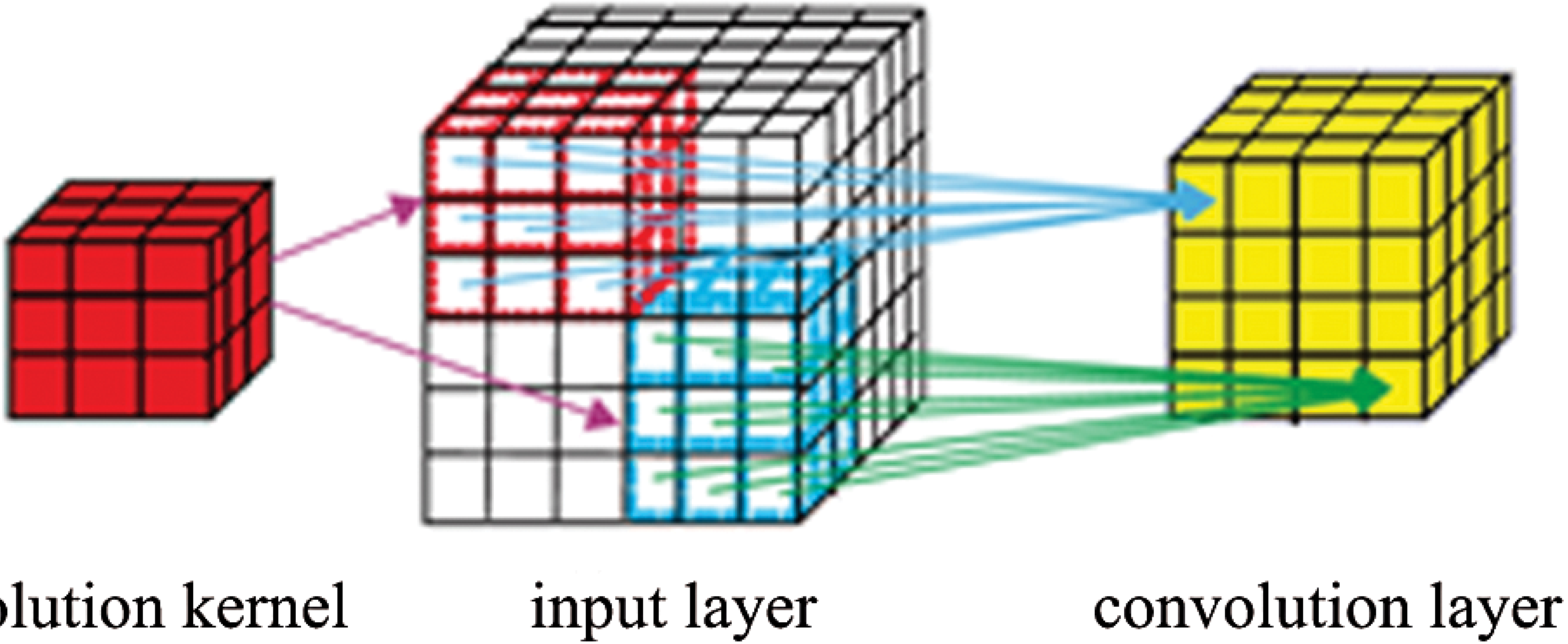



**Figure 3:** Illustration of the 3D convolutional operation

## Conv 3d: rels

- Learning Spatiotemporal Features with 3D Convolutional Networks
- https://github.com/karolzak/conv3d-video-action-recognition
- conv3d for image classification