

# Lecture 13

# Introduction to

# Convolutional Neural Networks

## Part 3

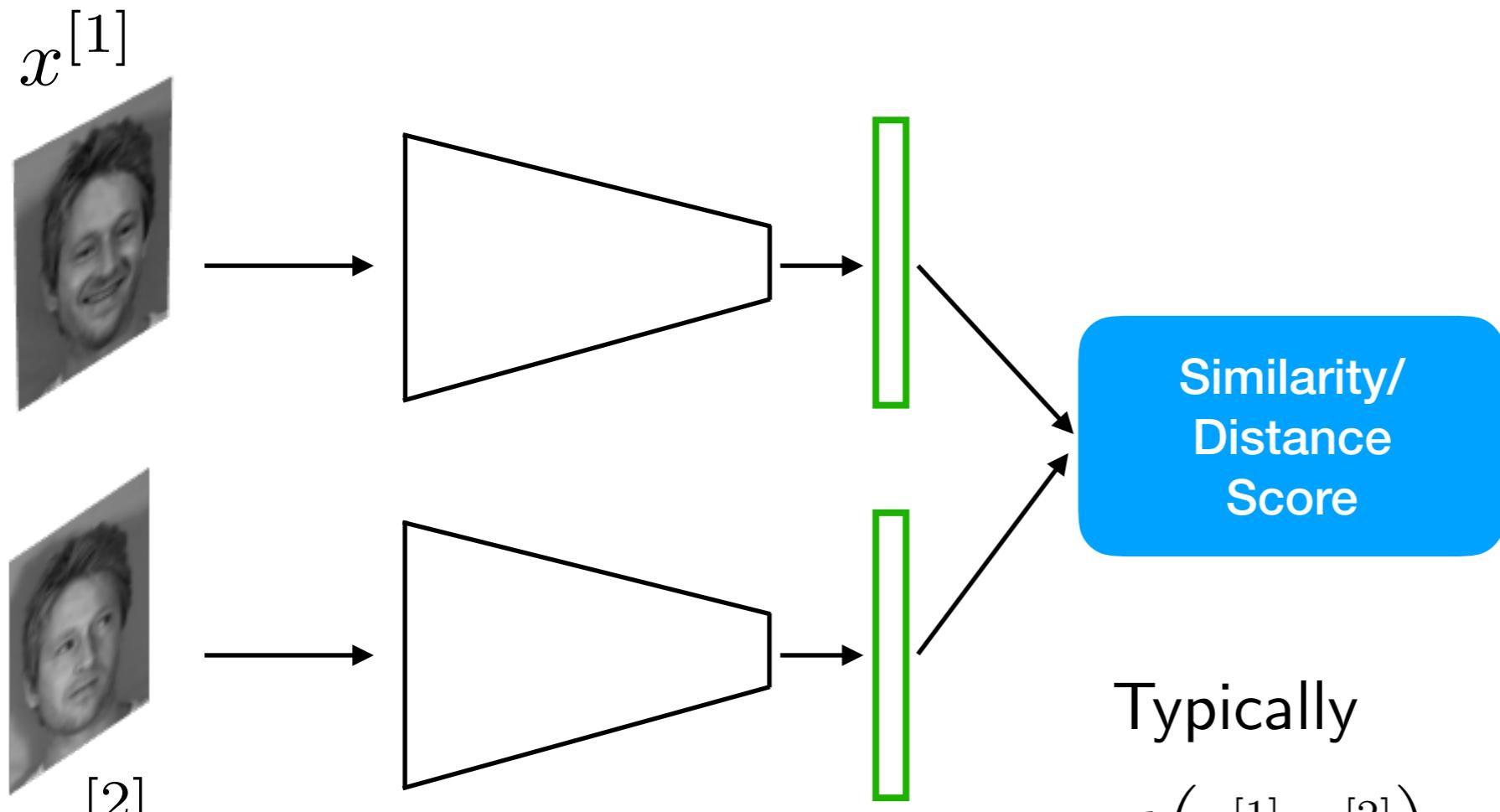
STAT 479: Deep Learning, Spring 2019

Sebastian Raschka

<http://stat.wisc.edu/~sraschka/teaching/stat479-ss2019/>

# Face Recognition and Metric Learning

# Siamese Networks



Typically

$$d(x^{[1]}, x^{[2]}) = \left\| f(x^{[1]}) - f(x^{[2]}) \right\|_2^2$$

or

$$d(x^{[1]}, x^{[2]}) = \left\| f(x^{[1]}) - f(x^{[2]}) \right\|_1$$

# Siamese Networks

Often used for "One-shot learning"

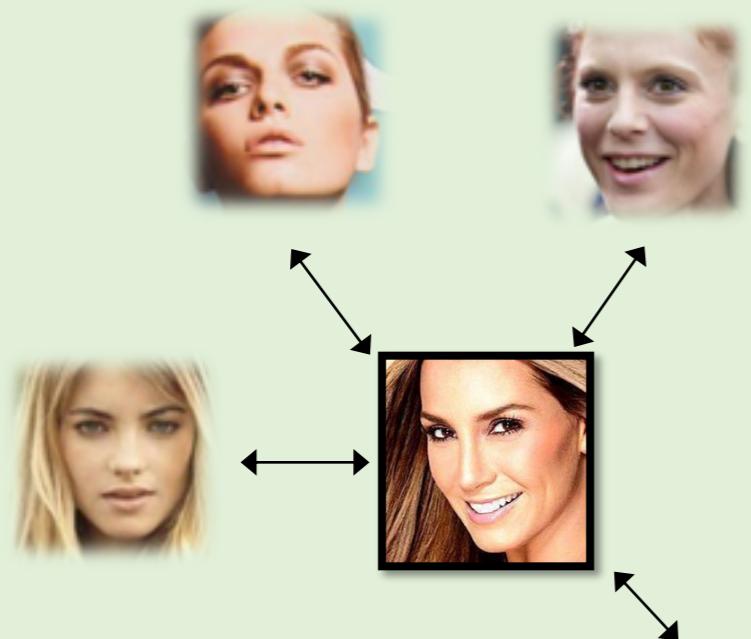
- Suppose you trained a Siamese network for verification tasks
- Now, suppose you have only  $\sim 1$  object per class
- You can compare any new object to any object based on maximum similarity to your given images  
(somewhat related to K-nearest neighbors)

# Face Recognition:

## Face Identification vs Face Verification

### A. Identification

Determine identity of an unknown person  
1-to- $n$  matching



### B. Verification

Verify claimed identity of a person  
1-to-1 matching



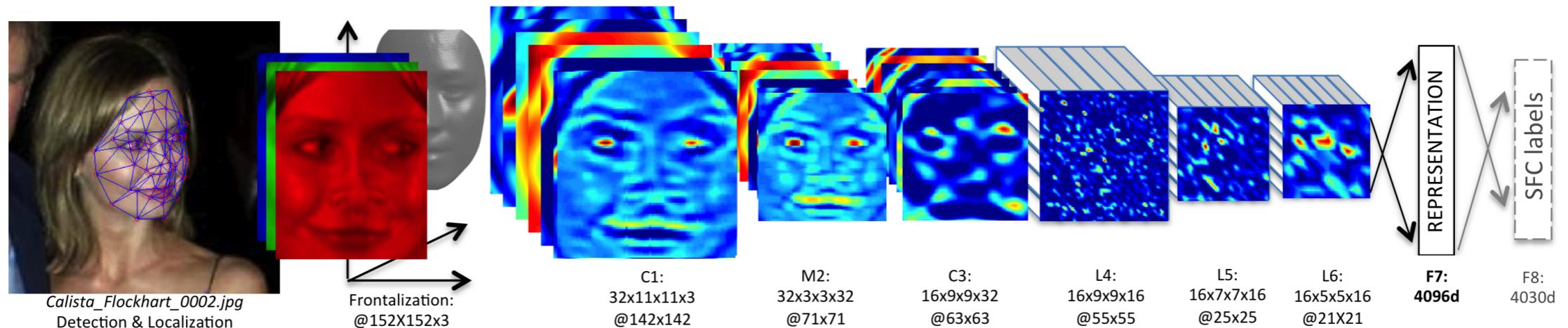
(MUCT dataset)

dataset link: <http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html>

dataset link: <http://www.milbo.org/muct/>

# DeepFace

Taigman, Yaniv, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. "Deepface: [Closing the gap to human-level performance in face verification.](#)" In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1701-1708. 2014.

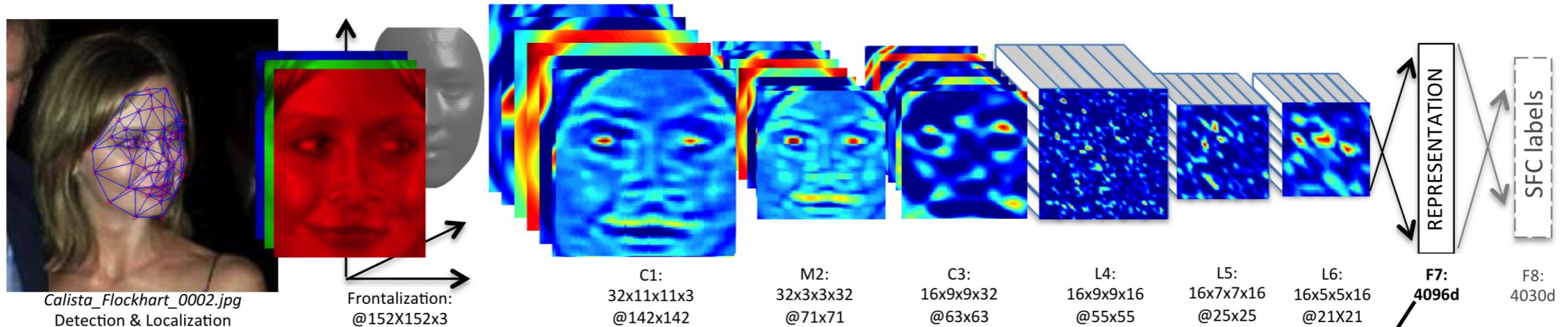


**Figure 2. Outline of the *DeepFace* architecture.** A front-end of a single convolution-pooling-convolution filtering on the rectified input, followed by three locally-connected layers and two fully-connected layers. Colors illustrate feature maps produced at each layer. The net includes more than 120 million parameters, where more than 95% come from the local and fully connected layers.

Hybrid between traditional methods and deep learning

# DeepFace

Taigman, Yaniv, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. "Deepface: [Closing the gap to human-level performance in face verification.](#)" In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1701-1708. 2014.



Given an image  $I$ , the representation  $G(I)$  is then computed using the described feed-forward network. Any feed-forward neural network with  $L$  layers, can be seen as a composition of functions  $g_\phi^l$ . In our case, the representation is:  $G(I) = g_\phi^{F_7}(g_\phi^{L_6}(\dots g_\phi^{C_1}(T(I, \theta_T))\dots))$  with the net's parameters  $\phi = \{C_1, \dots, F_7\}$  and  $\theta_T = \{x_{2d}, \vec{P}, \vec{r}\}$  as described in Section 2.

**Normalizatino** As a final stage we normalize the features to be between zero and one in order to reduce the sensitivity to illumination changes: Each component of the feature vector is divided by its largest value across the training set. This is then followed by  $L_2$ -normalization:  $f(I) := \bar{G}(I)/\|\bar{G}(I)\|_2$  where  $\bar{G}(I)_i = G(I)_i / \max(G_i, \epsilon)$ <sup>3</sup>. Since we employ ReLU activations, our system is not invariant to re-scaling of the image intensities. Without bi-

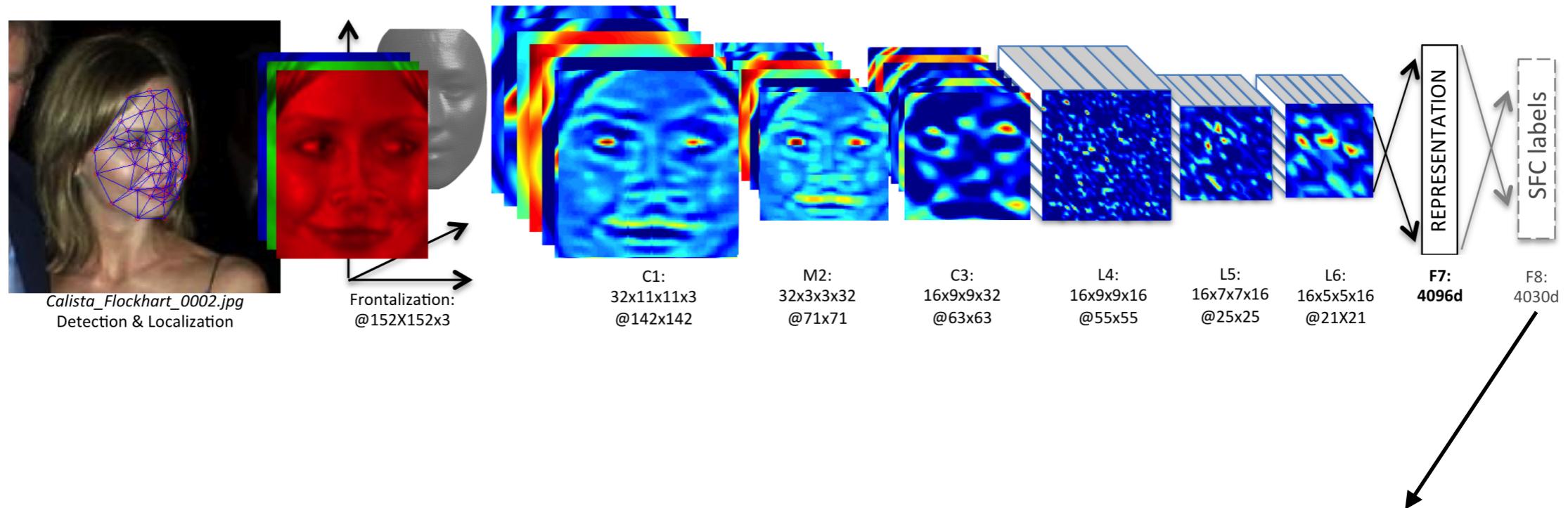
normalized  
feature vectors

<sup>2</sup>See the **supplementary** material for more details.

<sup>3</sup> $\epsilon = 0.05$  in order to avoid division by a small number.

# DeepFace - Face Recognition

Taigman, Yaniv, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. "Deepface: [Closing the gap to human-level performance in face verification.](#)" In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1701-1708. 2014.

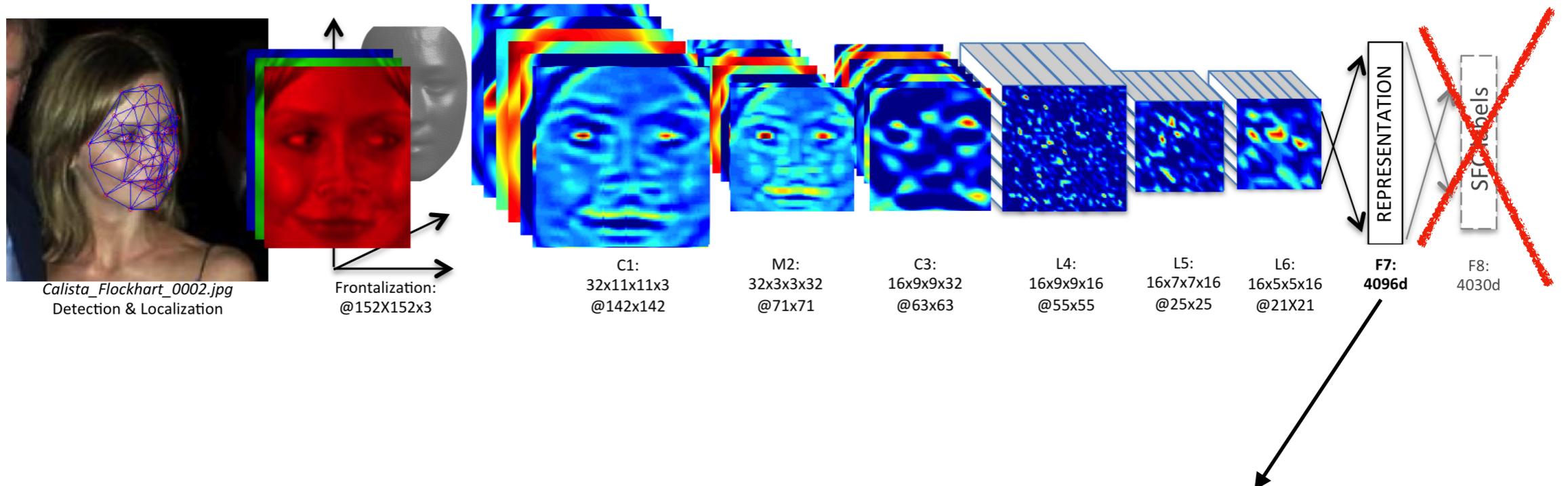


Regular softmax output layer for classifying faces (face IDs)  
optimized via cross-entropy loss.

Note they have 1-4k classes (they achieved a classification accuracy of ~93%).

# DeepFace - Face Verification

Taigman, Yaniv, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. "Deepface: [Closing the gap to human-level performance in face verification.](#)" In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1701-1708. 2014.



Weighted chi-square distance + SVM classifier for binary classification (predict whether two images depict the same person)

$$\chi^2(f_1, f_2) = \sum_i w_i (f_1[i] - f_2[i])^2 / (f_1[i] + f_2[i])$$

You may know this from other stats classes for comparing discrete probability distributions (histograms)

The weight is learned by the SVM.  
(They achieved a classification accuracy is ~97%.)

# FaceNet - Face Verification

Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "[Facenet: A unified embedding for face recognition and clustering.](#)" In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815-823. 2015.



Figure 2. **Model structure.** Our network consists of a batch input layer and a deep CNN followed by  $L_2$  normalization, which results in the face embedding. This is followed by the triplet loss during training.

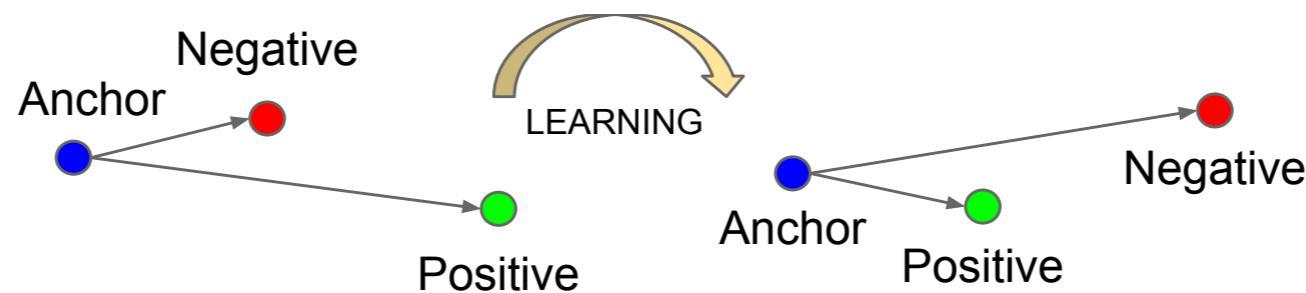


Figure 3. The **Triplet Loss** minimizes the distance between an *anchor* and a *positive*, both of which have the same identity, and maximizes the distance between the *anchor* and a *negative* of a different identity.

# Triplet Loss



Anchor

Positive



Anchor

Negative

Want encodings to be very similar  
(small distance)

Want encodings to be very different  
(large distance)

# Triplet Loss



Anchor



Positive



Anchor



Negative

Want encodings to be very similar  
(small distance)

Want encodings to be very different  
(large distance)

$$d(A, P) \leq d(A, N)$$

$$\|f(A) - f(P)\|_2^2 \leq \|f(A) - f(N)\|_2^2$$

# Triplet Loss



Anchor



Positive



Anchor



Negative

Want encodings to be very similar  
(small distance)

Want encodings to be very different  
(large distance)

$$d(A, P) + \alpha \leq d(A, N)$$

$$\|f(A) - f(P)\|_2^2 + \boxed{\alpha} \leq \|f(A) - f(N)\|_2^2$$

To make it a little harder

# Triplet Loss



Anchor



Positive

Want encodings to be very similar  
(small distance)



Anchor



Negative

Want encodings to be very different  
(large distance)

$$d(A, P) + \alpha \leq d(A, N)$$

$$\|f(A) - f(P)\|_2^2 + \alpha \leq \|f(A) - f(N)\|_2^2$$

Rearrange

$$\|f(A) - f(P)\|_2^2 + \alpha - \|f(A) - f(N)\|_2^2 \leq 0$$

# Triplet Loss



Anchor



Positive

Want encodings to be very similar  
(small distance)



Anchor



Negative

Want encodings to be very different  
(large distance)

Bounded loss function for training:

$$\mathcal{L}(A, P, N) = \max \left( \|f(A) - f(P)\|_2^2 + \alpha - \|f(A) - f(N)\|_2^2, 0 \right)$$

# Triplet Loss



Anchor



Positive

Want encodings to be very similar  
(small distance)



Anchor

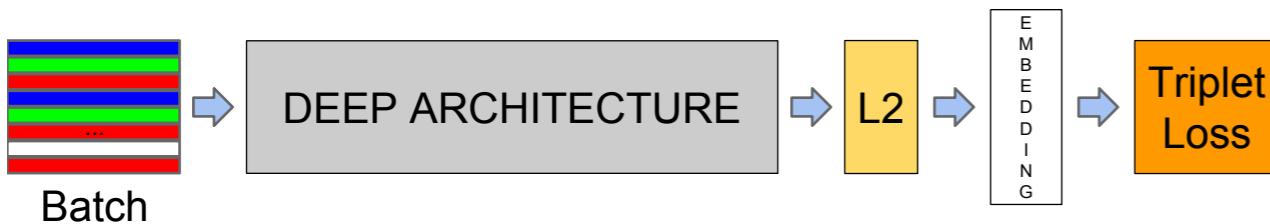


Negative

Want encodings to be very different  
(large distance)

In practice: Selecting good pairs (those that are "hard")  
is crucial during training

$$\mathcal{L}(A, P, N) = \max \left( \|f(A) - f(P)\|_2^2 + \alpha - \|f(A) - f(N)\|_2^2, 0 \right)$$



**Figure 2. Model structure.** Our network consists of a batch input layer and a deep CNN followed by  $L_2$  normalization, which results in the face embedding. This is followed by the triplet loss during training.

Suppose we have 2 L2-normalized vectors:

$$\|\mathbf{x}\|_2 = \|\mathbf{y}\|_2 = 1$$

The squared L2 distance is then proportional to the cosine similarity

$$\begin{aligned}
\|\mathbf{x} - \mathbf{y}\|_2^2 &= (\mathbf{x} - \mathbf{y})^\top (\mathbf{x} - \mathbf{y}) \\
&= \mathbf{x}^\top \mathbf{x} - 2\mathbf{x}^\top \mathbf{y} + \mathbf{y}^\top \mathbf{y} \\
&= 2 - 2\mathbf{x}^\top \mathbf{y} \\
&= 2 - 2 \cos(\mathbf{x}, \mathbf{y}) \quad \text{where } \cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^\top \mathbf{y}}{\|\mathbf{x}\| \cdot \|\mathbf{y}\|} \in [-1, 1]
\end{aligned}$$

# Optional: Recent Triplet Loss Variants

(not required), only for those who are interested

- Cosine Similarity-based triplet loss:

Li, Chao, Xiaokong Ma, Bing Jiang, Xiangang Li, Xuewei Zhang, Xiao Liu, Ying Cao, Ajay Kannan, and Zhenyao Zhu. "[Deep speaker: an end-to-end neural speaker embedding system](#)." *arXiv preprint arXiv:1705.02304* (2017).

- Angular Loss:

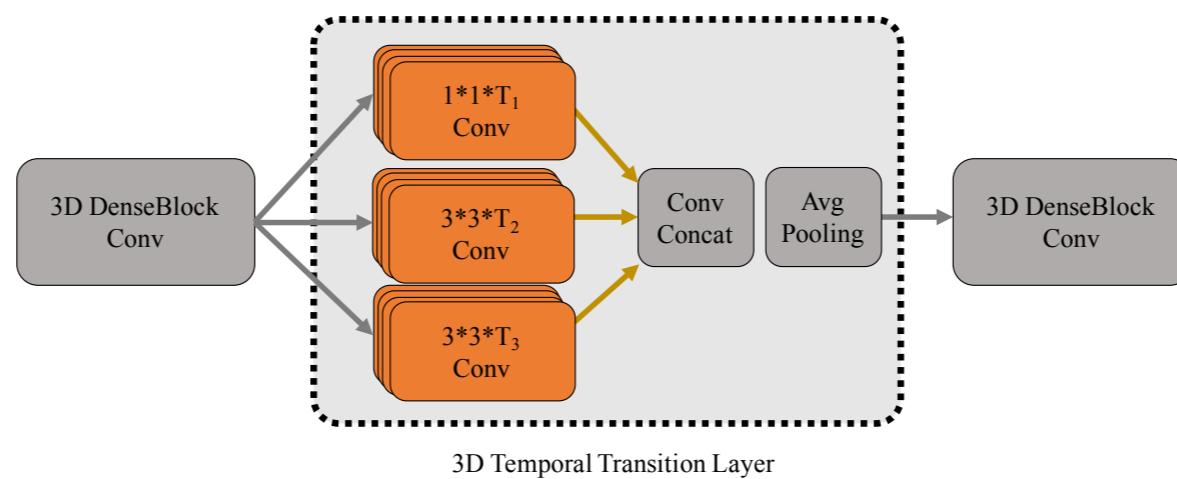
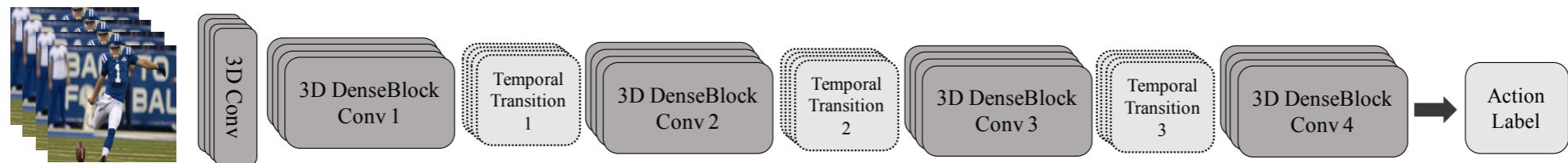
Wang, Jian, Feng Zhou, Shilei Wen, Xiao Liu, and Yuanqing Lin. "[Deep metric learning with angular loss](#)." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2593-2601. 2017.

- Large margin cosine loss:

Wang, Hao, Yitong Wang, Zheng` Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. "[Cosface: Large margin cosine loss for deep face recognition](#)." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5265-5274. 2018.

# Additional Concepts to Wrap Up the Intro to Convolutional Neural Networks

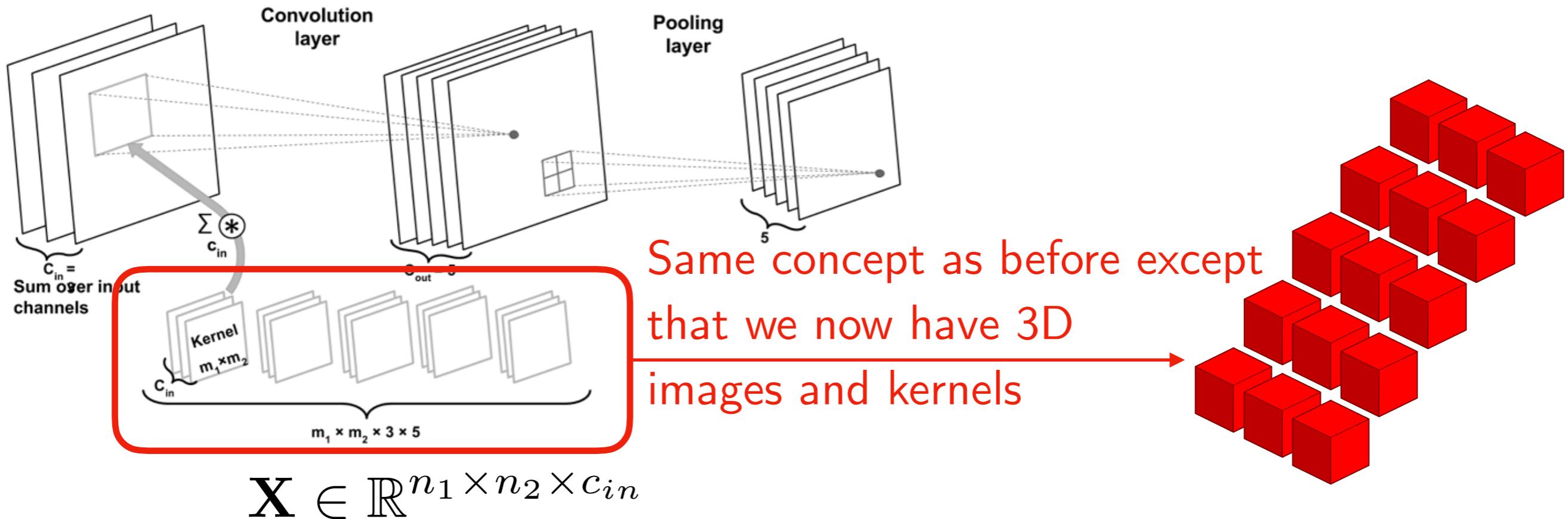
# ConvNets and 3D Inputs



Diba, Ali, Mohsen Fayyaz, Vivek Sharma, Amir Hossein Karami, Mohammad Mahdi Arzani, Rahman Yousefzadeh, and Luc Van Gool. "[Temporal 3d convnets: New architecture and transfer learning for video classification](#)." *arXiv preprint arXiv: 1711.08200* (2017).

Also very popular for Medical Imaging (MRI, CT scans ...)

# ConvNets and 3D Inputs



# ConvNets and 3D Inputs

Usage is similar to Conv2d, except that we now have 3 dimensional kernels

Conv3d

---

**CLASS** `torch.nn.Conv3d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True)`

[SOURCE]

Applies a 3D convolution over an input signal composed of several input planes.

<https://pytorch.org/docs/stable/nn.html?highlight=conv3d#torch.nn.functional.conv3d>

```
[1]: import torch
      import torch.nn as nn

[2]: m = nn.Conv3d(16, 33, 3, stride=2)
      m = nn.Conv3d(16, 33, (3, 5, 2), stride=(2, 1, 1), padding=(4, 2, 0))
      input = torch.randn(20, 16, 10, 50, 100)
      output = m(input)

[3]: input.size()

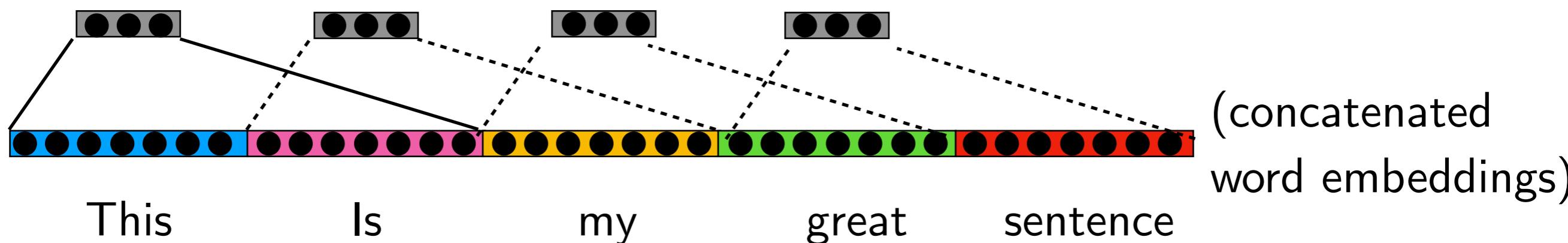
[3]: torch.Size([20, 16, 10, 50, 100])

[4]: output.size()

[4]: torch.Size([20, 33, 8, 50, 99])
```

# ConvNets for Text with 1D Convolutions

We can think of text as image with width 1



<https://pytorch.org/docs/stable/nn.html#conv1d>

Conv1d

**CLASS** `torch.nn.Conv1d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True)`

[SOURCE]

Applies a 1D convolution over an input signal composed of several input planes.

# CNNs for Text (with 2D Convolutions)

Good results have also been achieved by representing a sentence as a matrix of word vectors and applying 2D convolutions (where each filter uses a different kernel size)

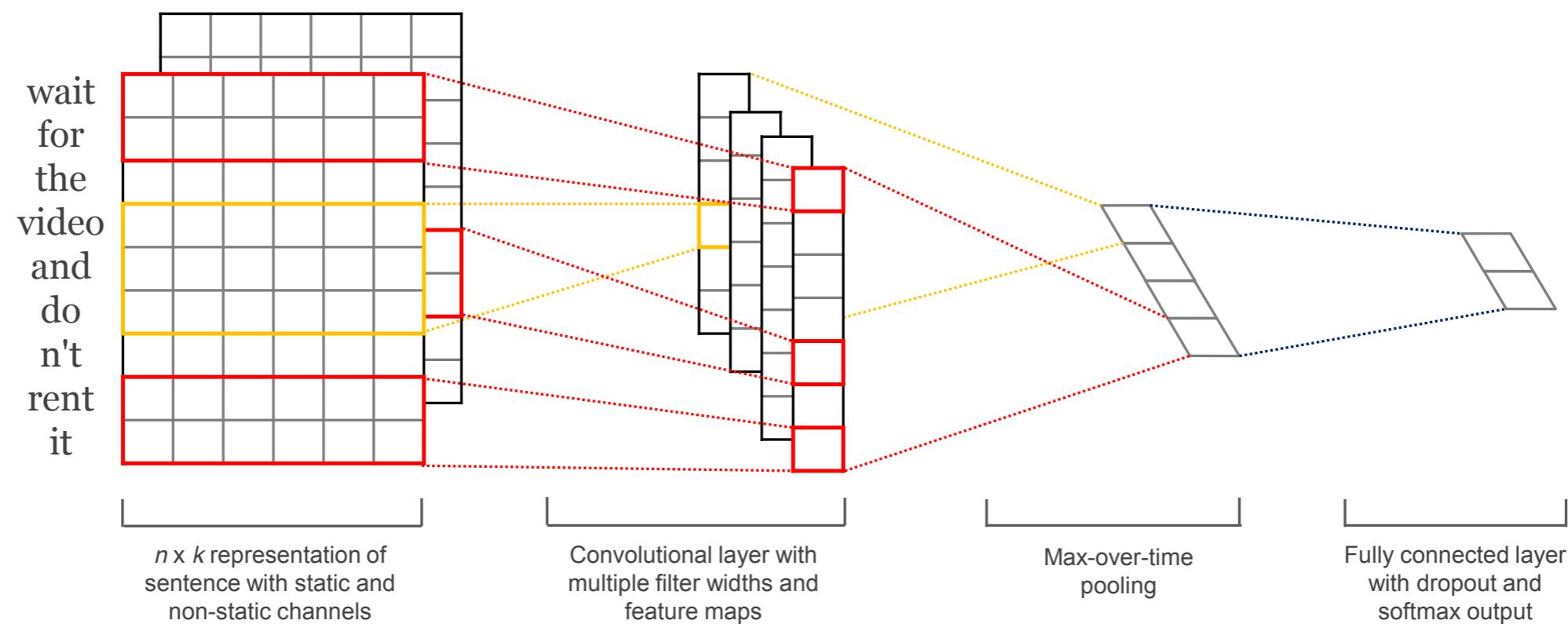


Figure 1: Model architecture with two channels for an example sentence.

Kim, Y. (2014). [Convolutional neural networks for sentence classification](#). *arXiv preprint arXiv:1408.5882*.

# Pre-Trained Models for Text

<https://modelzoo.co/model/pytorch-nlp>