Extreme Low Resolution Activity Recognition with Multi-Siamese Embedding Learning

(AAAI 2018 accepted)

Kiyoon Kim[1,2]

[1]EgoVid Inc., Ulsan, South Korea; [2]Ulsan National Institute of Science and Technology, Ulsan, South Korea

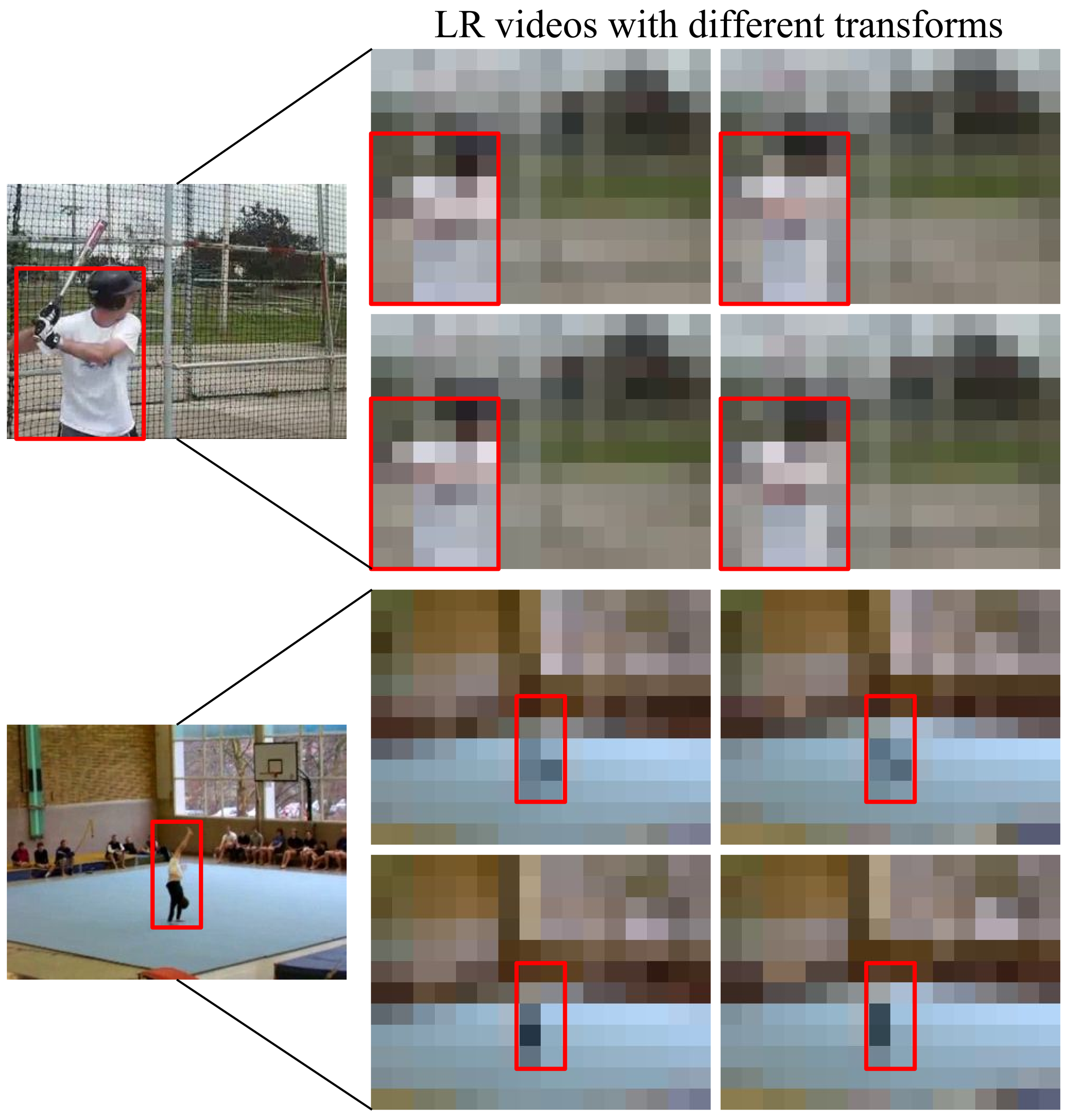
Introduction

- Usage of computer vision (camera) is increasing fast with various applications such as autonomous vehicle, drone, robots, wearable devices, smart home, and so on.
- Will lead to serious *privacy concern*. Once high-resolution data is on CPU or GPU memory, hackers may snatch the data.

Objective

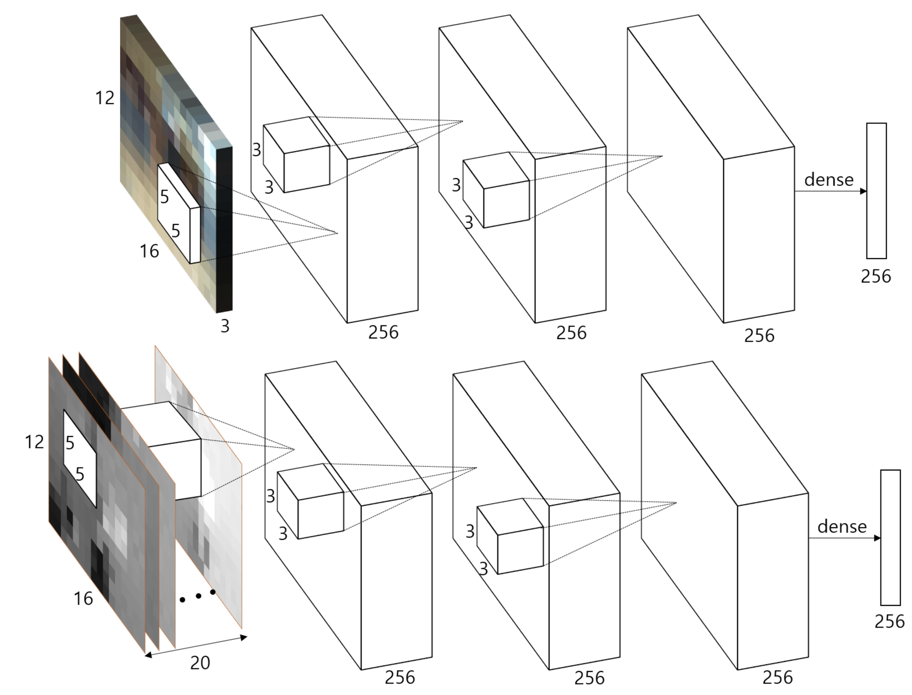
- Activity recognition with anonymized video data (e.g., 16x12).
- Assume that high-resolution training data are available from public sources. (i.e., YouTube)
- Take advantage of the fact that a single high-resolution video can generate multiple low-resolution videos from slightly *different transforms*.

LR videos with different transforms

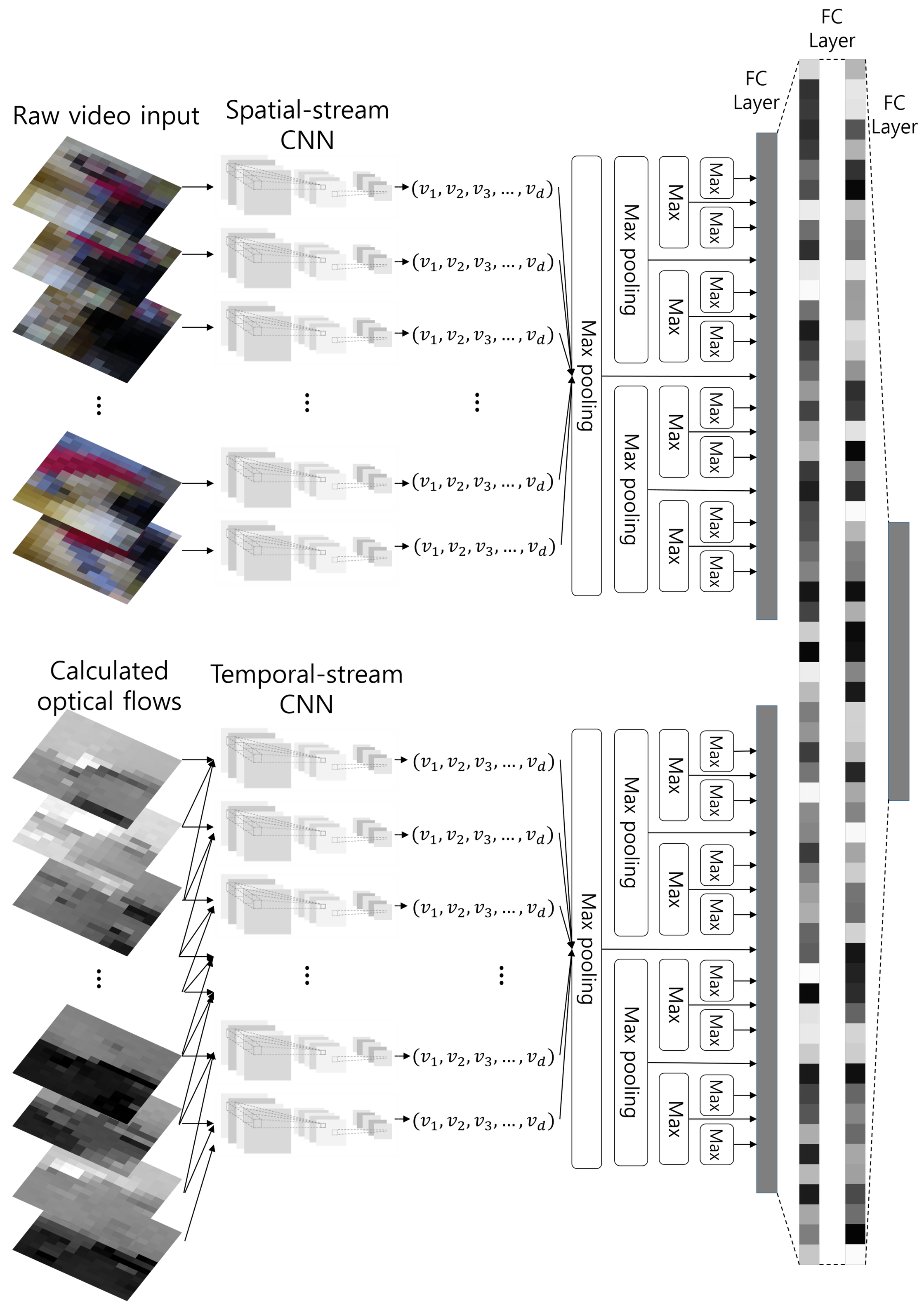


Two-stream CNN

- **Spatial stream**: takes RGB pixel values of each frame (e.g., 16x12x3)
- **Temporal stream**: takes 10-frame concatenated optical flow values (e.g., 16x12x20)



- **Temporal pyramid**: applies two-stream models above for each frame, and takes temporal max pooling with different intervals



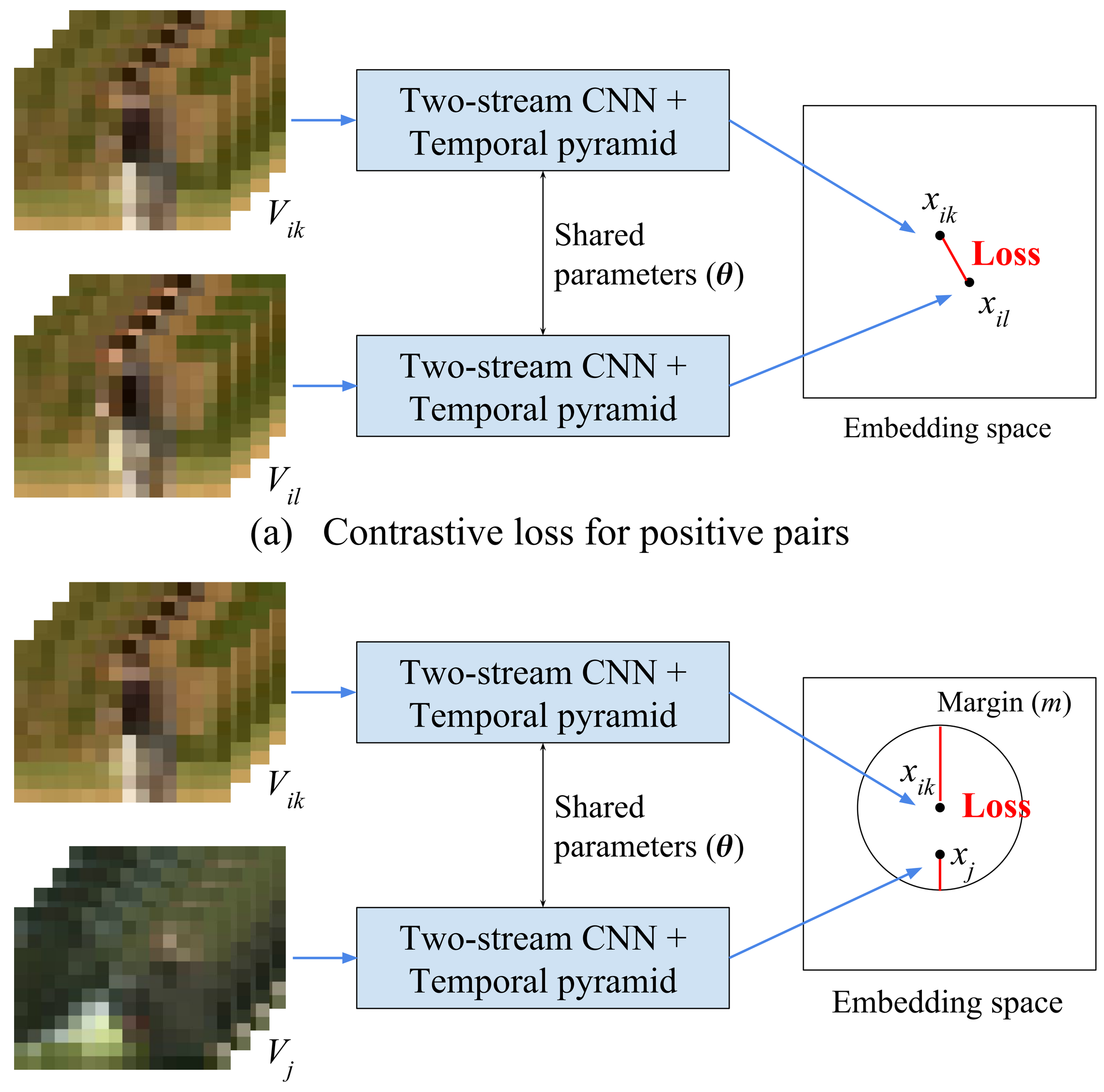
Multi-Siamese Constrastive Loss

- **Siamese CNN**: The training tries to minimize the embedding distance between a positive pair while maximizing the distance between a negative pair.

$$L_{siam}(\theta) = \sum_{(i,j)}^{B} y'_{(i,j)} ||x_i - x_j||_2^2 + (1 - y'_{(i,j)}) \max(0, m - ||x_i - x_j||_2)^2$$

* m: margin, B: the batch of LR examples being used, i and j: the indices of pairs in the batch.

$$L(\theta) = \lambda_1 L_{siam}(\theta) + \lambda_2 L_{class}(\theta)$$



(a) Contrastive loss for positive pairs

(b) Contrastive loss for negative pairs

- **Multi-Siamese CNN**: 2 · n branches sharing the parameters for the embedding and the classifier learning.

$$L_{multi}(\theta) = \sum_{i \in B}\Big[\sum_{(k,l)\in B_1} ||x_{ik} - x_{il}||_2^2 + \max(0, n^2 \cdot m^2 - (\sum_{k}\sum_{j \in B_2} ||x_{ik} - x_j||_2^2))\Big]$$

$$L(\theta) = \lambda_1 L_{multi}(\theta) + \lambda_2 \sum L_{class}(\theta)$$

Experiment Results

Table 1: Classification accuracies (%) measured with the 16x12 HMDB dataset [Kuehne et al., 2011]. Reporting the mean and standard deviation of each method.

Approach	One-Stream	Two-Stream
Baseline CNN	25.08 ± 0.40	31.50 ± 0.30
Data augmentation	25.17 ± 0.24	35.34 ± 0.41
Our multi-Siamese	26.21 ± 0.27	**37.70** ± 0.17

Table 2: The average performance of classification accuracies (%) measured with the 16x12 DogCentric dataset [Iwashita et al., 2014].

Approach	One-Stream	Two-Stream
Baseline CNN	53.05	61.25
Data augmentation	57.61	68.09
Our multi-Siamese	59.08	**69.43**

Table 3: Comparing our approach with previous state-of-the-arts on the **16x12** HMDB dataset.

Approach	Accuracy
3-layer CNN [Ryoo et al., 2017]	20.81 %
ResNet-32 [He et al., 2016]	22.37 %
PoT [Ryoo et al., 2015]	26.57 %
ISR [Ryoo et al., 2017]	28.68 %
Two-stream [Chen et al., 2017]	29.2 %
Our two-stream CNN with pyramid	31.50 %
Ours	**37.70** %

Table 4: Comparing our approach with previous state-of-the-art results reported on the **16x12** DogCentric activity dataset.

Approach	Accuracy
Iwashita et al. [Iwashita et al., 2014]	46.2 %
ITF [Wang and Schmid, 2013]	10.0 %
PoT [Ryoo et al., 2015]	64.6 %
ISR [Ryoo et al., 2017]	67.36 %
Our two-stream CNN with pyramid	61.25 %
Ours	**69.43** %

Our approach runs in real-time (∼50 fps) on a Nvidia Jetson TX2 mobile GPU card with our Python code using the TensorFlow library.