

# **Deep Reinforcement Template Matching**

**Onkar Krishna**

**Go Irie**

**Xiaomeng Wu**

**Takahito Kawanishi**

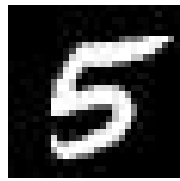
**Kunio Kashino**

**NTT Corporation**

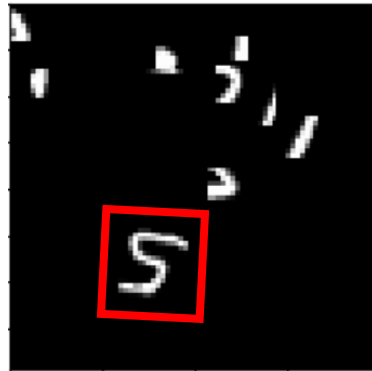
**July 30<sup>th</sup>, 2019**

# Our Task: Template Matching

- To find a part of a reference image that matches to a query image.



Query



Reference



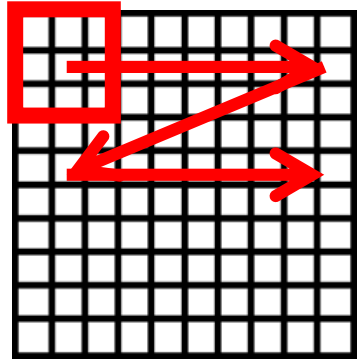
Query



Reference

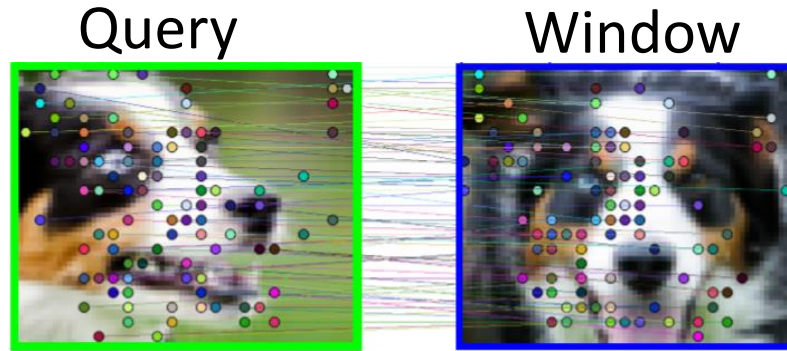
- Applications: object tracking, image re-targeting, image denoising, etc.
- A desired algorithm should be *fast* and *robust* to noises such as background clutter, illumination change, and geometric transformations.

# Existing Frameworks



## Exhaustive Search

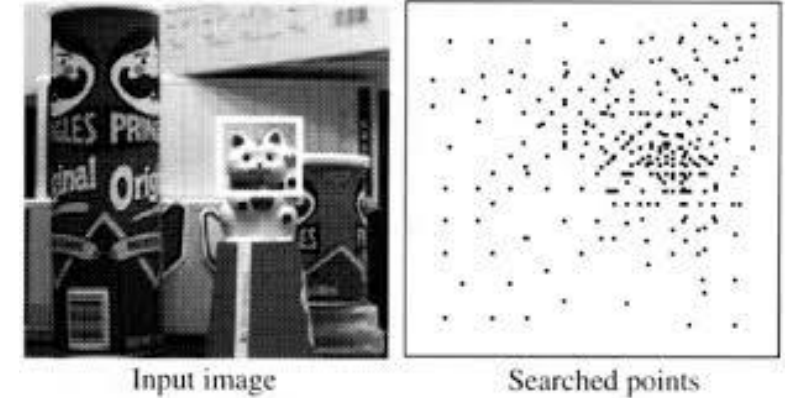
- Accurate but inefficient



## Pixel Subsampling

(e.g., [Tali+, 2015])

- Measure window similarity only with fraction of pixels



## Window Skipping

(e.g., [Vinod+, 1997], [Liu+, 2017])

- Skip unnecessary windows based on feature statistics

☹️ **They still evaluate many windows / pixels not to lose accuracy!**

😊 **We aim at improving the balance between run time and accuracy!**

# Our Idea

- We use **machine learning** to pick and evaluate only the highly prospective regions of the reference image.
- We consider the matching problem as a **sequential decision problem**.

Localize the target region by sequentially determining the pose of the window



Query

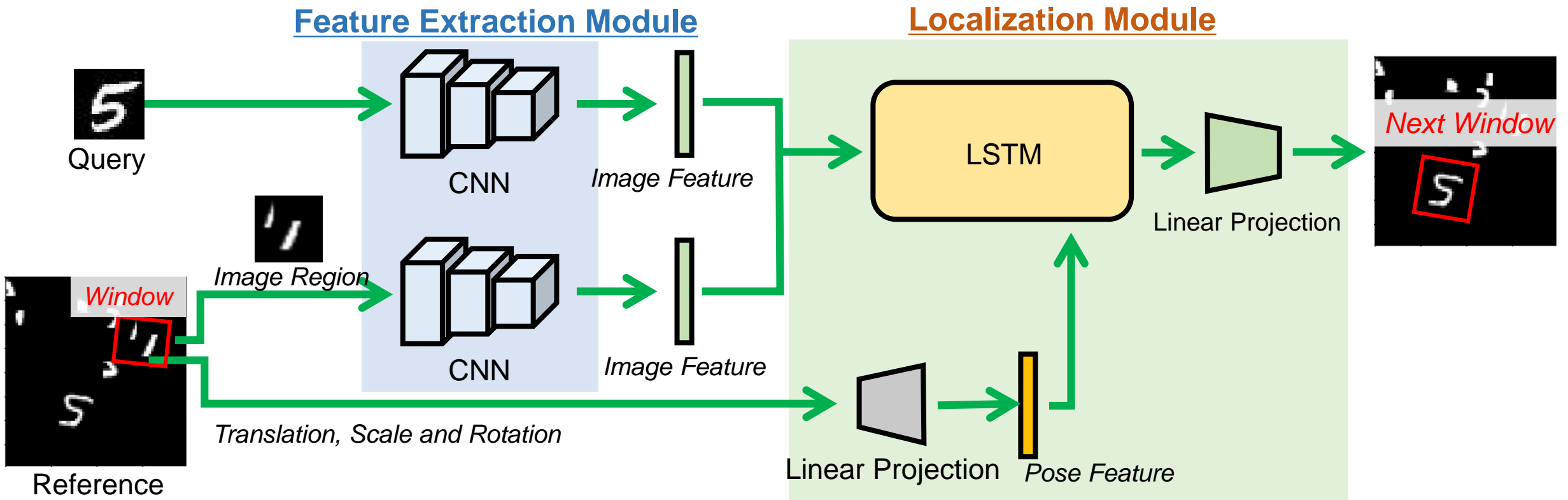


Reference

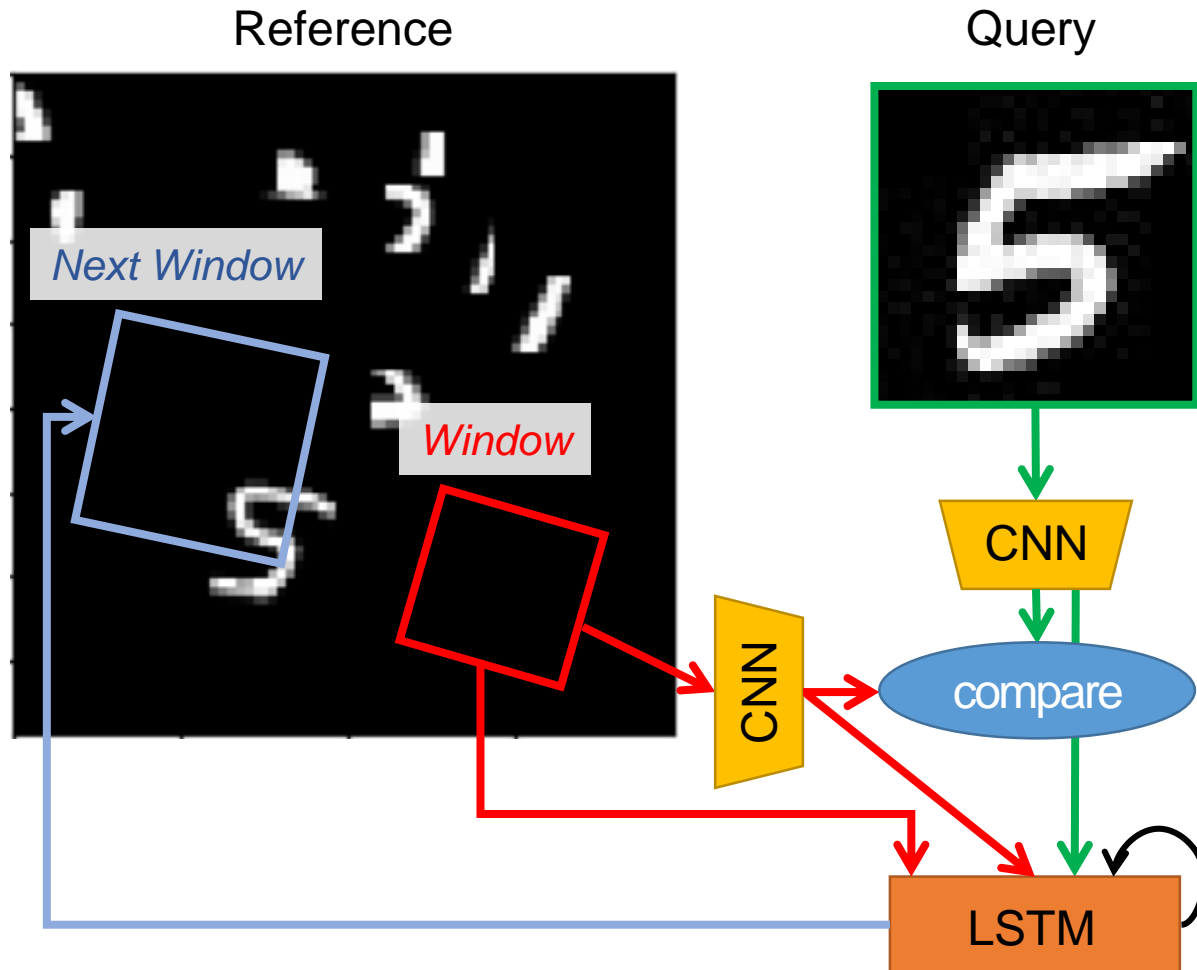
- We use a reinforcement learning (RL) method to **jointly learn good image features as well as search path** in an explorative manner, **without any class label or exact pose supervision**.

# Overview of Model Architecture

Our model has **feature extraction module** and **localization module**



# Algorithm Behavior



Step 0. Choose the initial pose of the window based on the global reference image

Step 1. Compare CNN features between the chosen window and the query

Step 2. Given the features and the current pose of the window, determine the next pose of the window

Step 3. If the number of trials reaches the limit, terminate and output the pose of the window. Otherwise back to Step 1.

# Learning Strategy

Combination of **reward maximization** and **feature loss minimization**

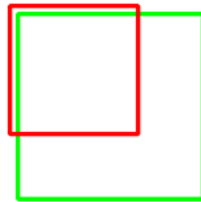
## a. Reward maximization

- Get a reward “1” if  $\text{IoU} > 0.5$ , otherwise “0”.

IoU:

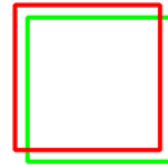
Intersection-over-Union

IoU: 0.4034



Poor

IoU: 0.7330



Good

IoU: 0.9264

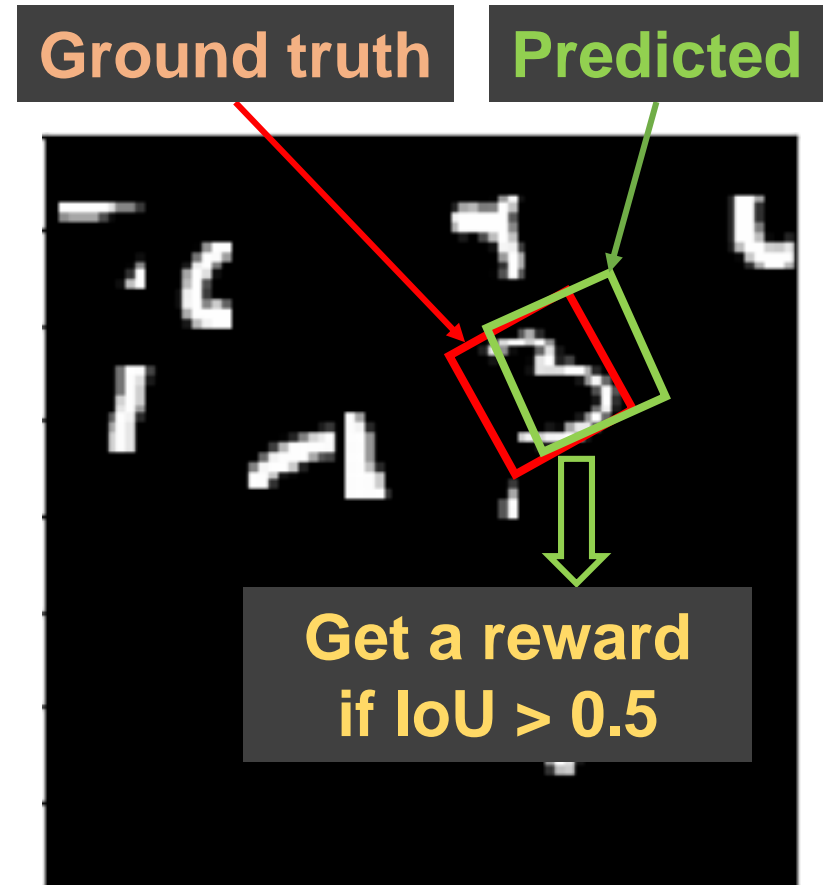


Excellent

- Maximize the expected reward by policy gradient

## b. Feature loss minimization

- Contrastive loss to learn good features for matching



# Datasets

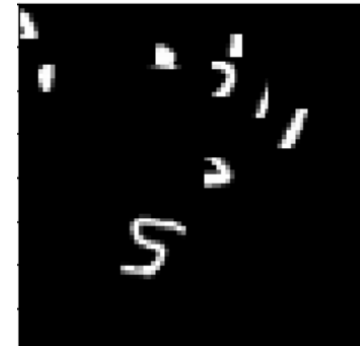
## 1. Transformed MNIST

- Query is affine transformed in reference
- 60K pairs for training and 10K for testing



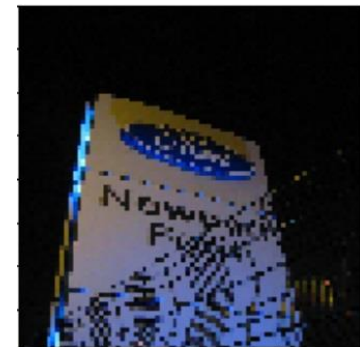
## 2. Transformed+Cluttered MNIST

- Reference is also contaminated by noisy fragments of digits
- 60K pairs for training and 10K for testing



## 3. FlickrLogos-32

- Real world image of logos
- 2000 pairs for training and 240 for testing





# Performance Metrics

**a. Success rate** (“#correct matches” to “#pairs”)

➤ We judge a search successful iff IoU > 0.5

**b. Run time** for each query-reference pair

**c. Number of windows** evaluated

# Quantitative Results -- Success Rate

Success rate | higher is better

Dataset	Transformed MNIST	Transformed+ Cluttered MNIST	FlickrLogos-32
<b>Ours</b>	<b>0.89</b>	<b>0.85</b>	<b>0.34</b>
[Yacov+, ICCV11]	0.51	0.18	0.10
[Tali+, CVPR15]	0.56	0.20	0.31

**Ours is robust to background clutter and able to handle geometric transformations.**





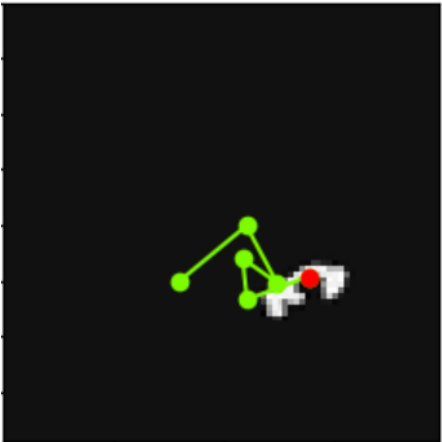
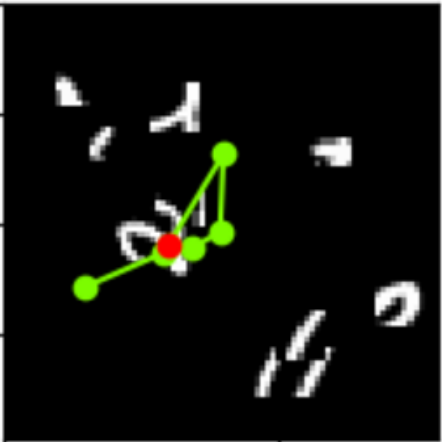
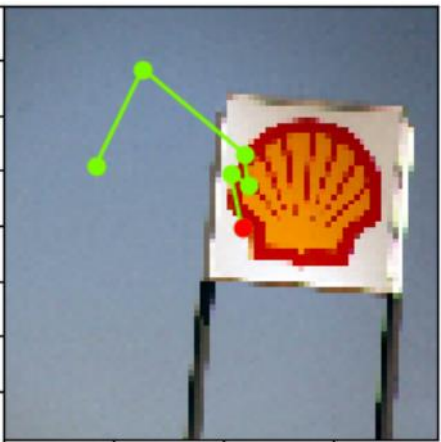





# Quantitative Results -- Run Time

Run time in milliseconds (number of windows) | lower is better

Dataset	Transformed MNIST	Transformed+ Cluttered MNIST	FlickrLogos-32
<b>Ours</b>	<b>3.8 (6)</b>	<b>4.0 (6)</b>	<b>26.2 (6)</b>
[Yacov+, ICCV11]	<b>1.1 (2809)</b>	<b>1.0 (2809)</b>	<b>5.2 (2809)</b>
[Tali+, CVPR15]	90.1 (6400)	90.3 (6400)	110.6 (6400)

**While [Yacov+, ICCV11] is slightly faster, ours is much more accurate with a slight expense of run time.**

# Qualitative Results

Query				
Search Path				
Found Region				

**The window converges to the target region once a part of target region is captured, otherwise it randomly moves to next location.**

# Summary

**We proposed a deep reinforcement learning approach for template matching.**

- **Key Features**

- *Accuracy/speed*: Our method achieved better matching accuracy with highly competitive search speed.
- *Explorative learning*: Our model jointly learns search path and good image features for matching in a reinforcement learning manner.

- **Future Work**

- Analyze the relationship between # windows and performance
- Extension to other types of data (video, audio, point clouds, etc.)

# FAQ

# What exactly our model is learning ? logics

- Which trajectory is most of the time provides positive reward for this particular logo? Upwards or Downward?



*For this logo our model learned that most of the time moving **Upward** is more rewarding than **downward**, since these logos are mostly at higher places!*

# What makes our model robust to noise?

## “deep-features”

- We introduced contrastive loss function which learned good features for matching.

$$L = \begin{cases} d^2(q, g) & \text{If "Success"} \\ \max\{0, m - d(q, g)\}^2 & \text{otherwise} \end{cases}$$

$d(q, g)$ : Euclidian distance between query  
and image region features  $q$  and  $g$

- If Success, make features closer between the window and the query, otherwise farther.



# How to select initial window pose?

