Efficient Visual Tracking with Exemplar Transformer

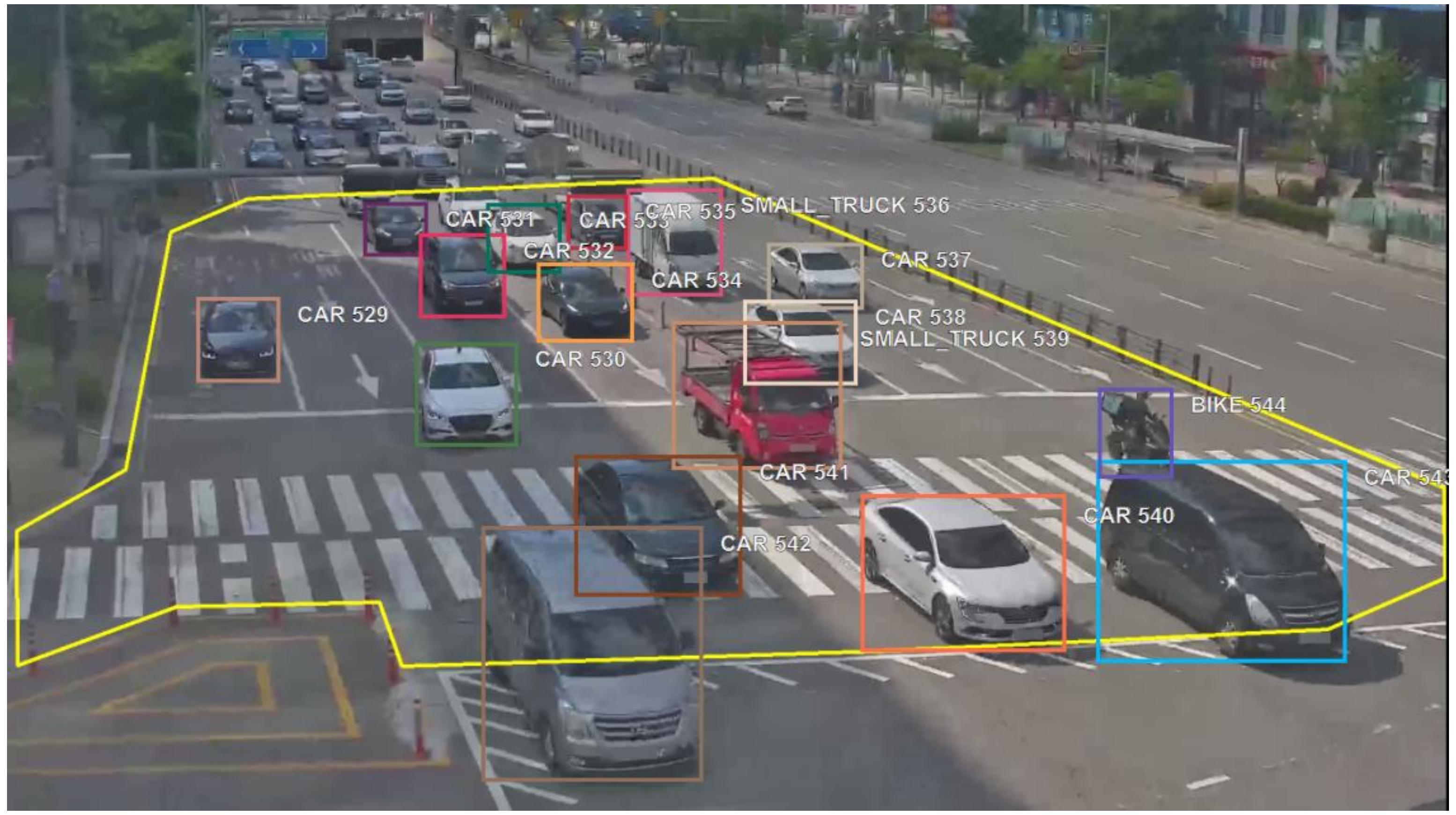
Philippe Blatter, Menelaos Kanakis, Martin Danelljan,
Luc Van Gool, ETH Zurich, KU Leuven
WACV 2023

이채원, 임채연, 김도완
2024-02-21

since 1978
동서울대학교
DONG SEOUL UNIVERSITY

Visual Tracking in Deep Neural Network

- Deeper Network
- More accurate Bounding Boxes
- Transformers



Example of Bounding Box

Their development requires greater costs

이채원, 임채연, 김도완

https://aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&aihubDataSe=realm&dataSetSn=522

Visual Tracking in Deep Neural Network

Increase in demand ⬆

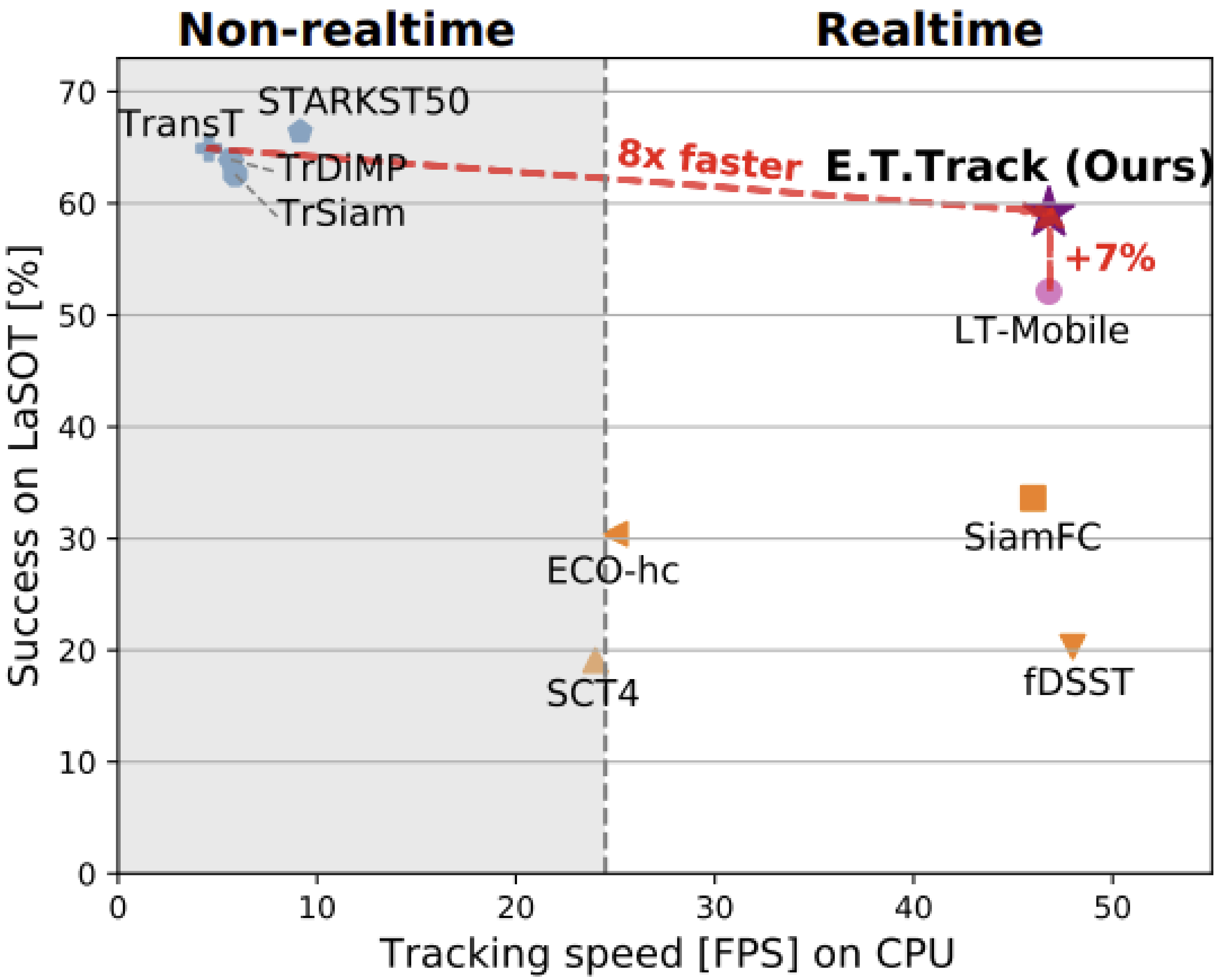
- Autonomous Driving
- Robotics

little attention ⬇

- efficient deep tracking architectures

Need visual tracker capable of real-time operation

ETTrack's Tracking Speed



이채원, 임채연, 김도완

Visual Tracking in Transformer

Excellent performance in images and videos

High cost and increased tracking time

Aims to improve tracking performance without compromising runtime

Hypothesis with Exemplar Attention

1. Explanatory power of a single Global Query value

2. Shared memory role of a small set of Exemplar values

Captures target object information more effectively than
A Standard attention module

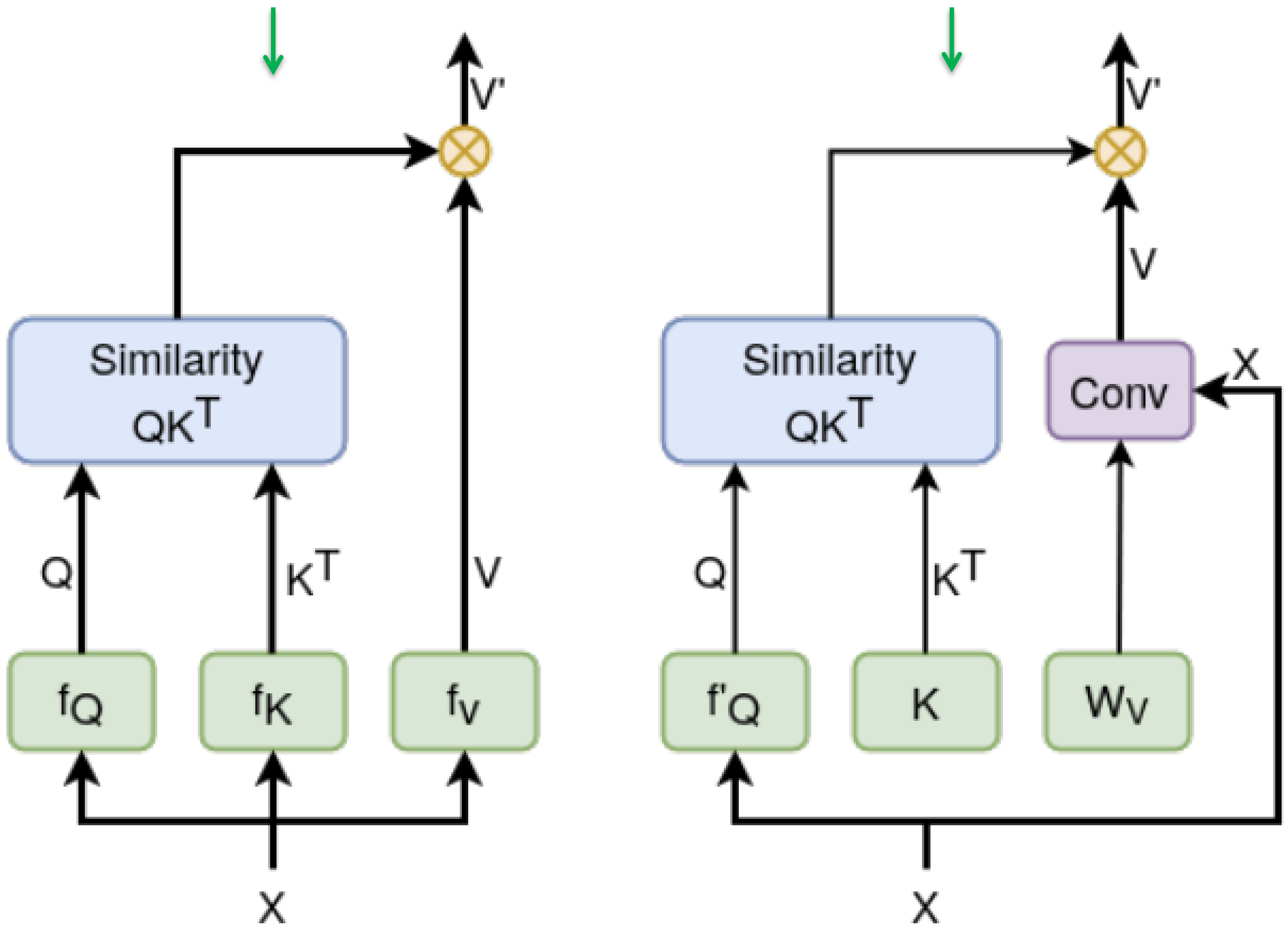
DataSet

DataSet	Explanation
LaSOT	Large-scale single object tracking, Variety of sizes and shapes, Generalization performance evaluation
OTB-100	100 test sequences, Tracking performance evaluation in various scenarios
UAV-123	Tracking objects captured by unmanned aerial vehicles
NFS	Object tracking in high-speed video
Tracking Net	Large-scale online object tracking
VOT-ST2020	Short-term tracking performance evaluation

Efficient Tracking Architectures

Large tensor

Small tensor



이채원, 임채연, 김도완

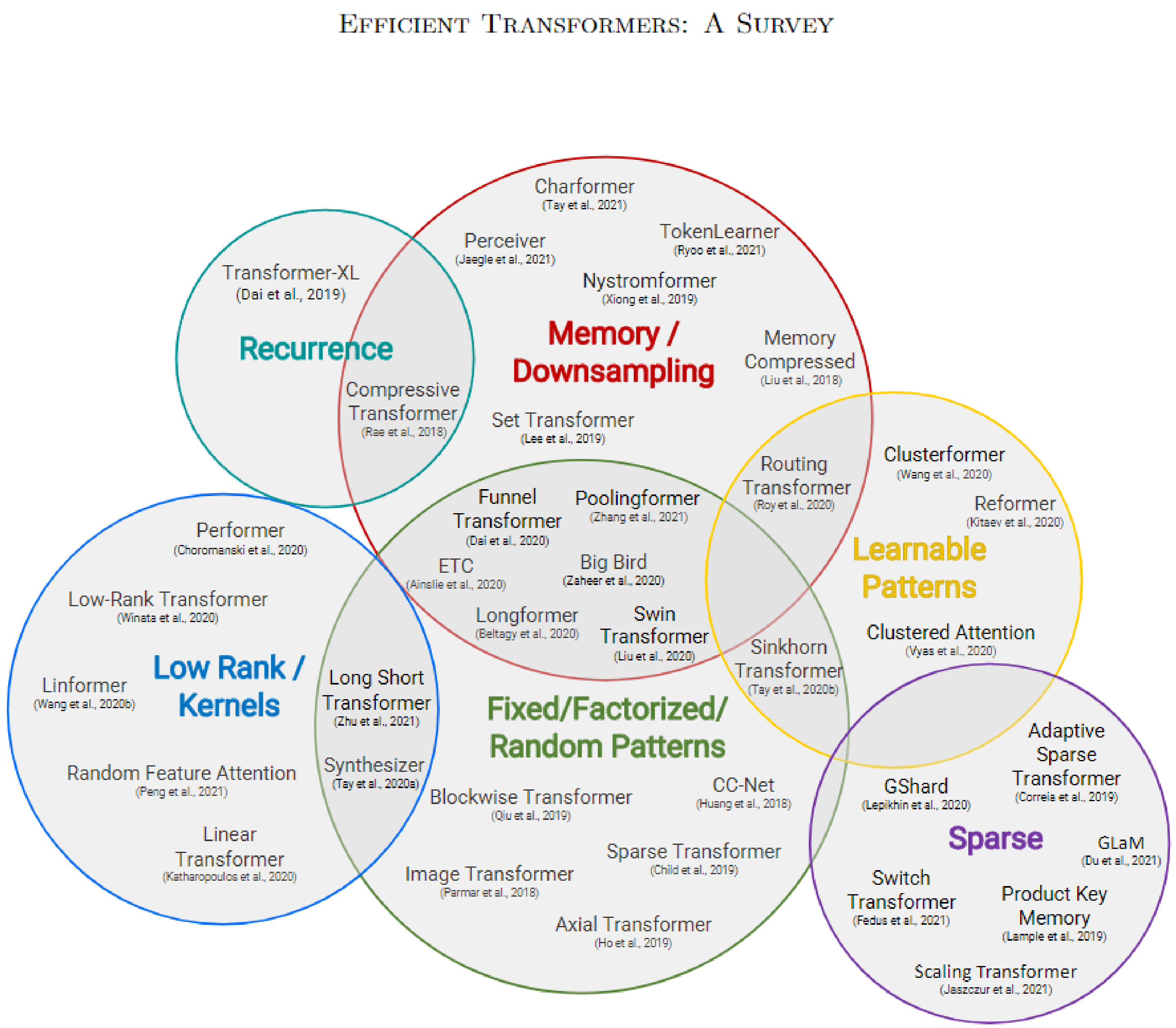
Efficient Transformers

1. Low Rank/ Kernel Methods

2. Memory/ Downsampling Methods

3. Fixed/ Factorized/ Random Patterns

4. Learnable Patterns



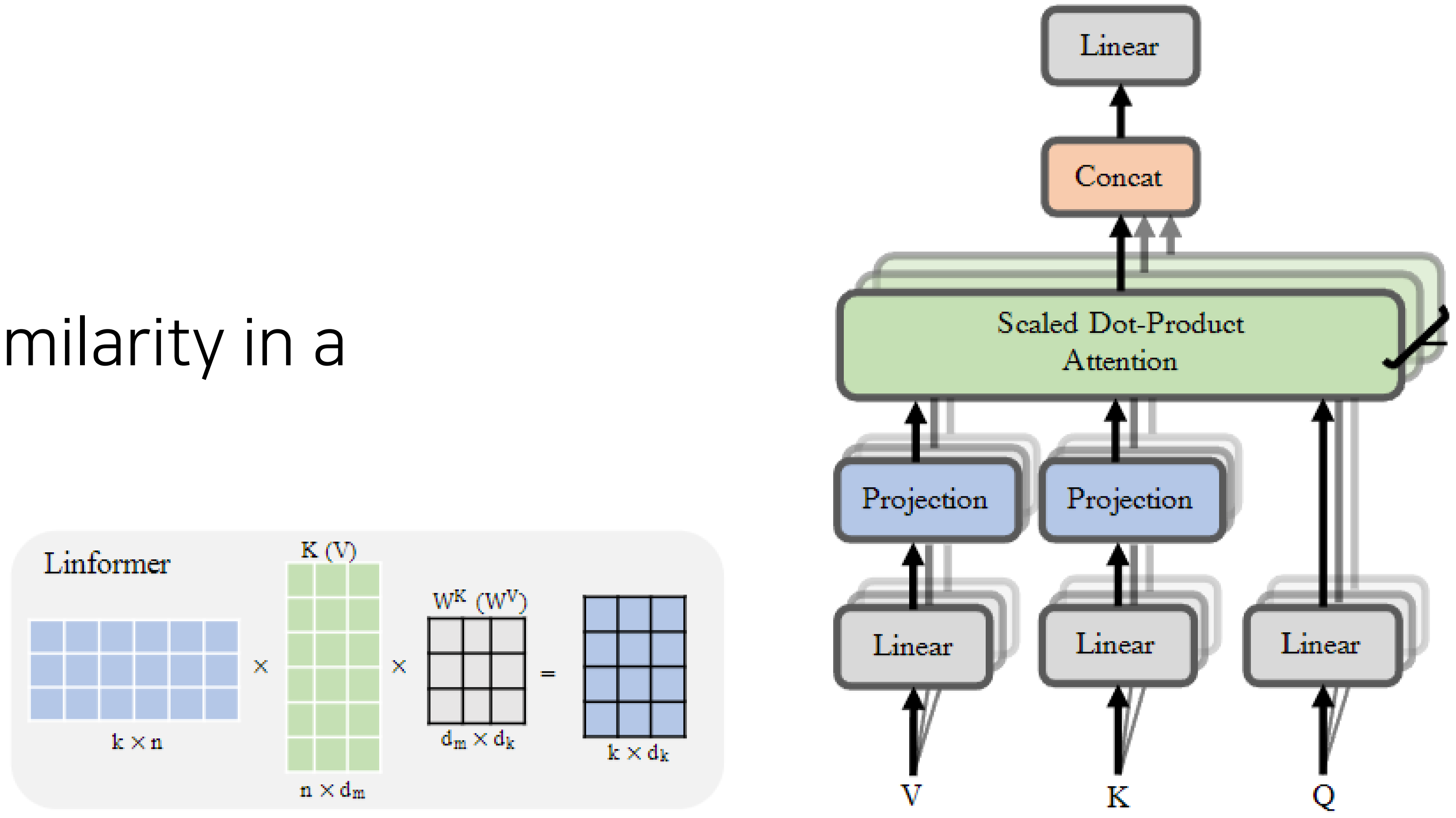
이채원, 임채연, 김도완

Efficient Transformers

1. Low Rank/ Kernel Methods

- Low Rank : low-rank approximation

- Kernel Methods : using a kernel function to compute similarity in a specific feature space



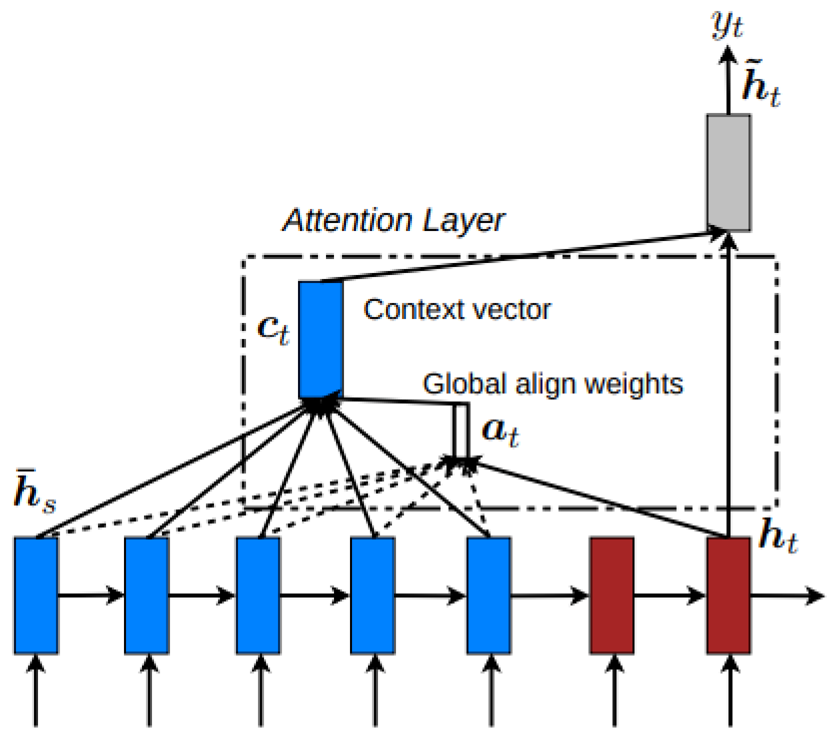
(Linformer)

Assuming and using a simplified version of the self-attention matrix

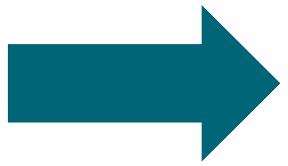
Efficient Transformers

2. Memory/ Downsampling Methods

- Memory : Transformer multitasking information from various positions
- Downsampling : Shortening the sequence



(Global Attention)

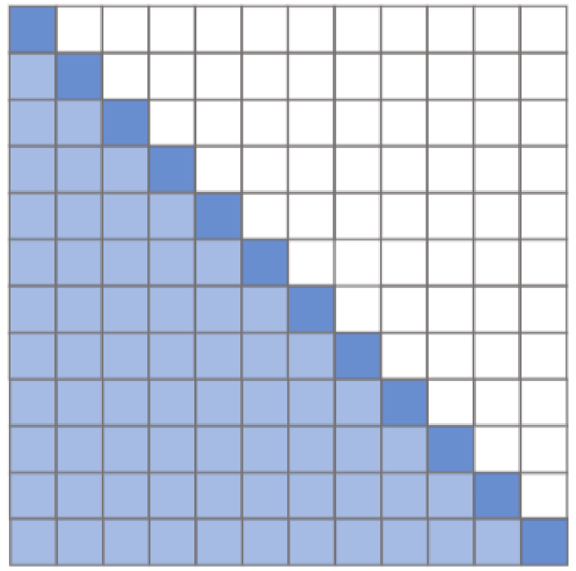


Training extra memory for multitoken access or shortening sequences

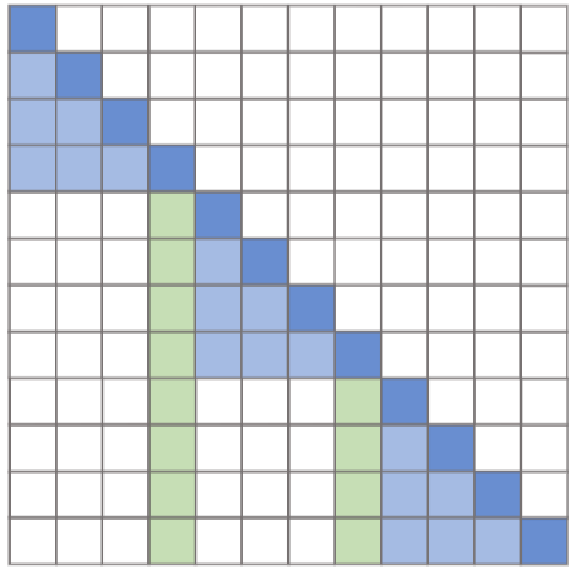
Efficient Transformers

3. Fixed/ Factorized/ Random Patterns

- Fixed : self-attention mechanism with predefined distinctive structure or weights
- Factorized : Breaking down a weight matrix into smaller matrices
- Random : Defining self-attention matrix patterns randomly



(a) Transformer



(b) Sparse Transformer

Limited field of view of the Self-Attention

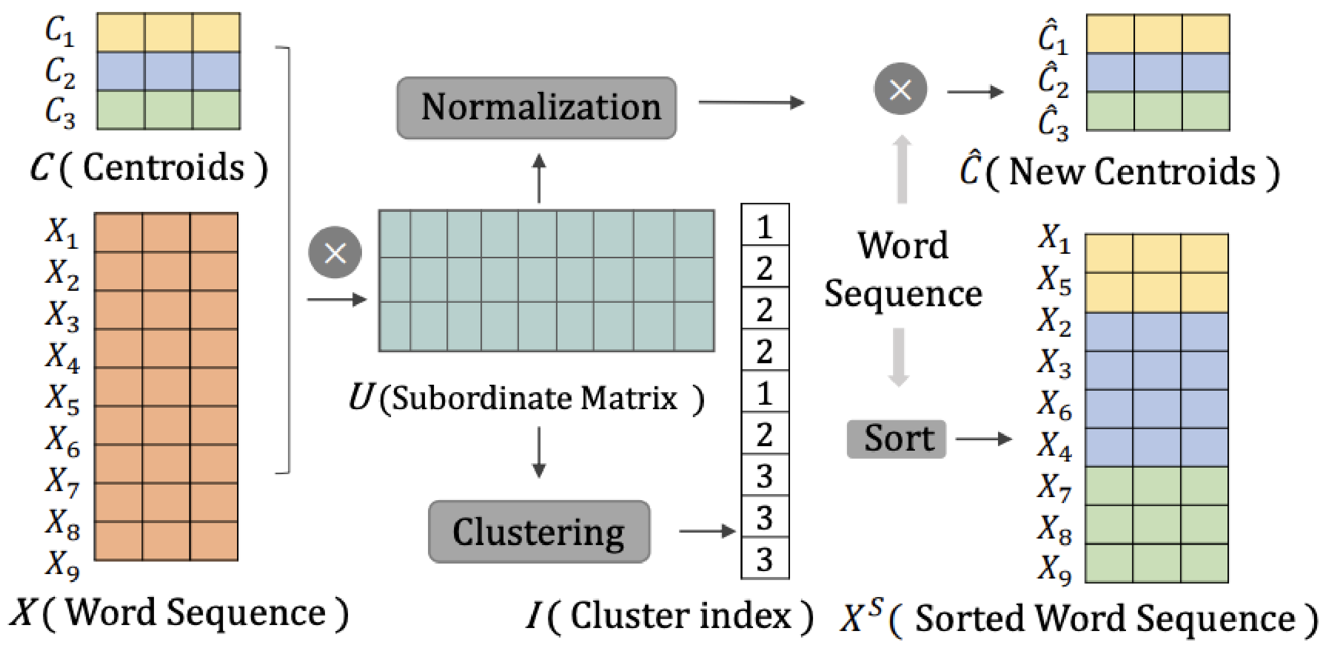
이채원, 임채연, 김도완

https://sh-tsang.medium.com/review-sparse-transformer-80cbba4ebaa4

Efficient Transformers

4. Learnable Patterns

- Learnable Patterns : Model uses trainable weight patterns



(a) Neural Clustering Method

(Clusterformer)

Switching from fixed to dynamic patterns in the standard Transformer

이채원, 임채연, 김도완

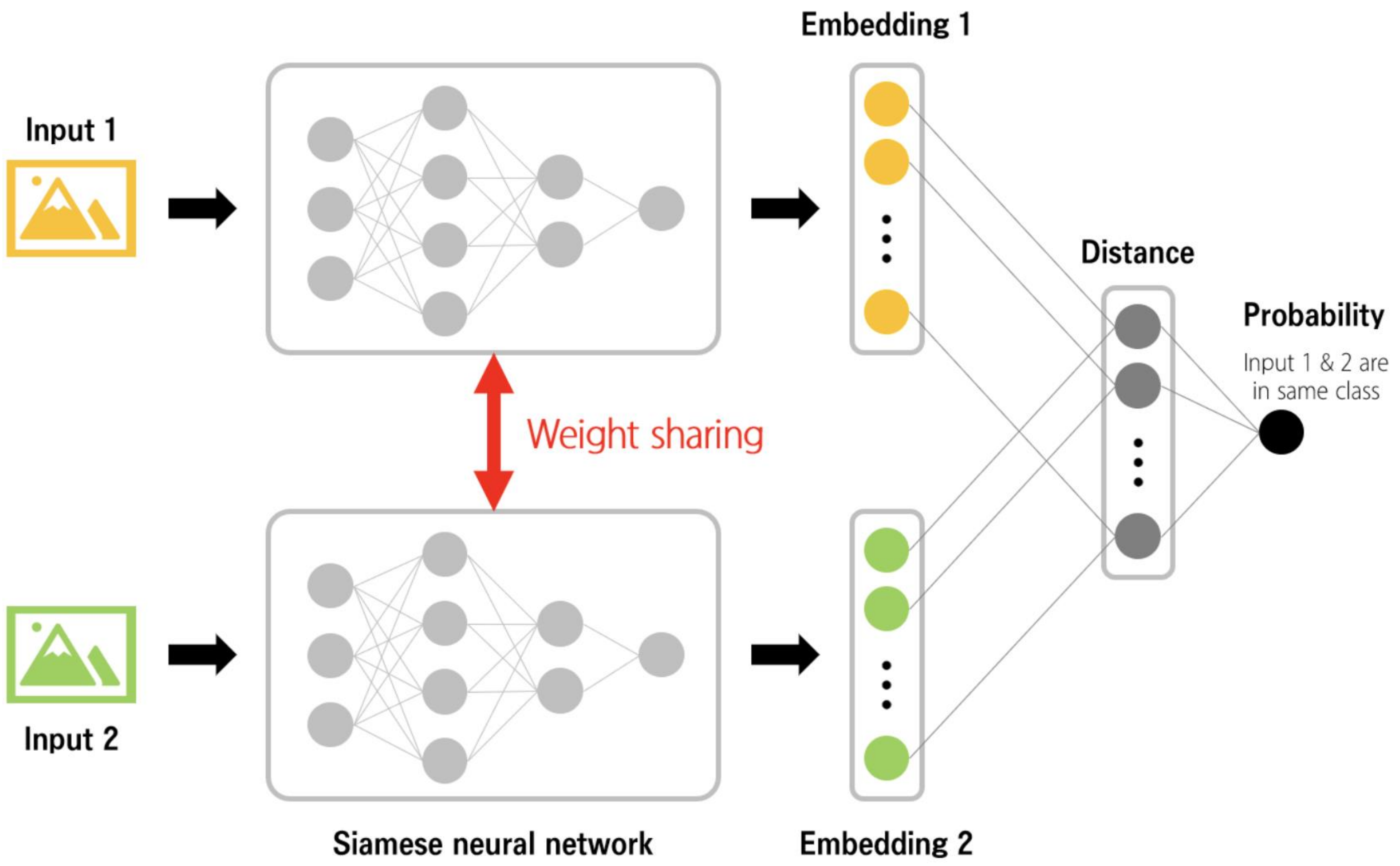
Efficient Transformers

Intersection Memory/Downsampling and Fixed/Factorized/RandomPatterns

➡ Pooling Query(Downsampling) + Independency Key(Fixed)

Siamese Tracker

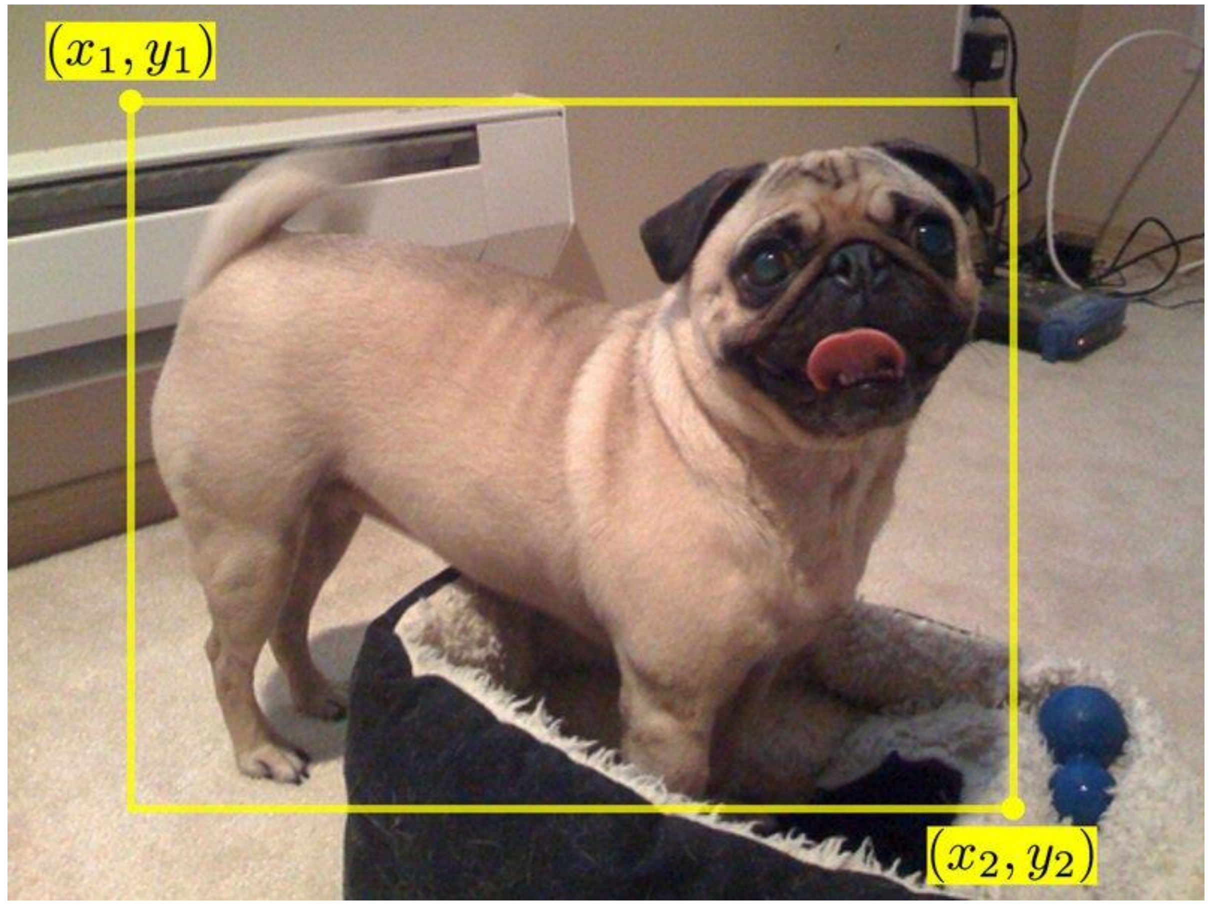
- Neural Network architecture
- Frequently used in visual tracking
- Contains one or more identical networks
- Sharing weight
- Same the parameters and weights
- Learning a distance function



https://velog.io/@hsbc/230602-Siamese-Network

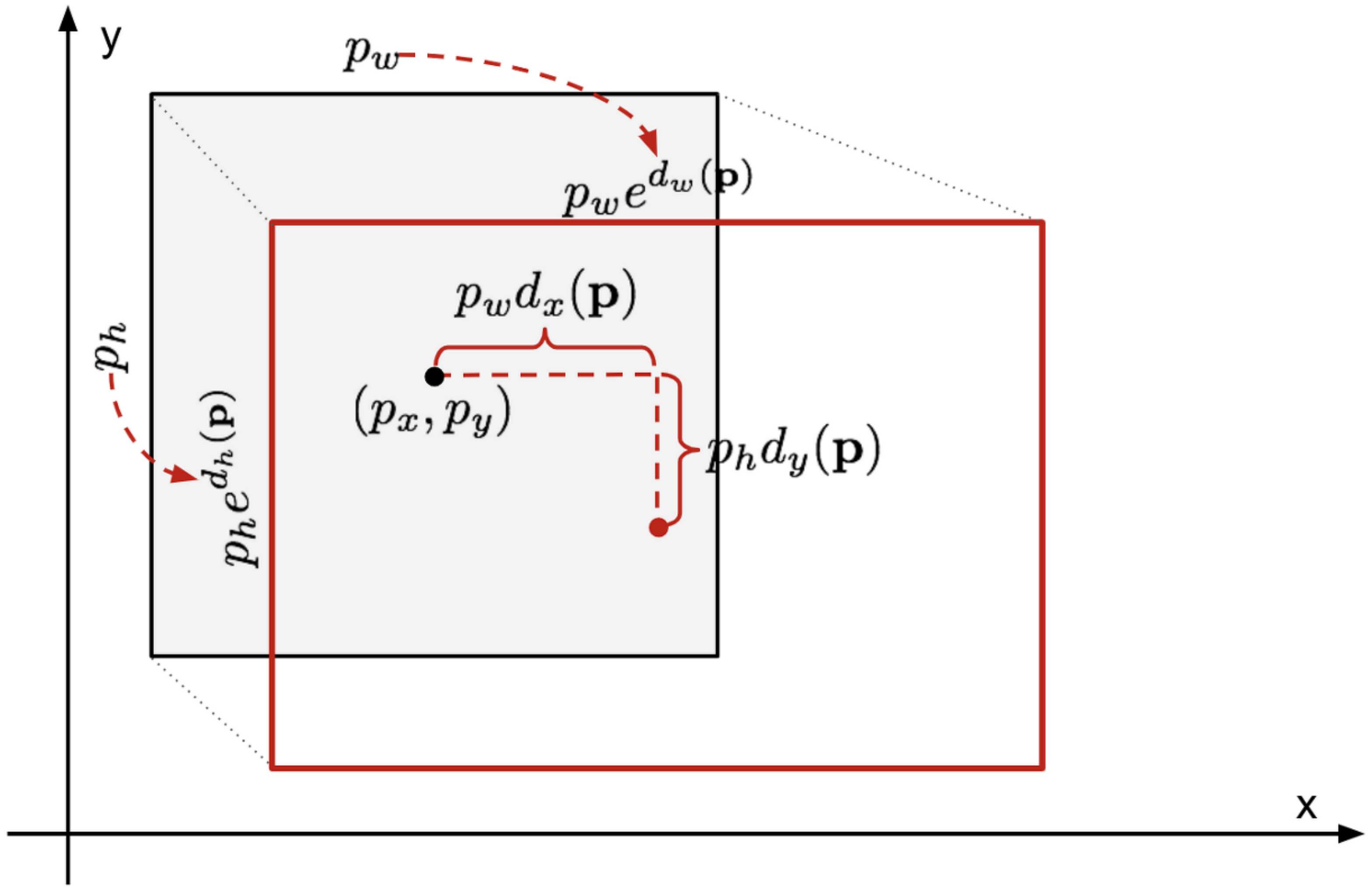
Bounding Box

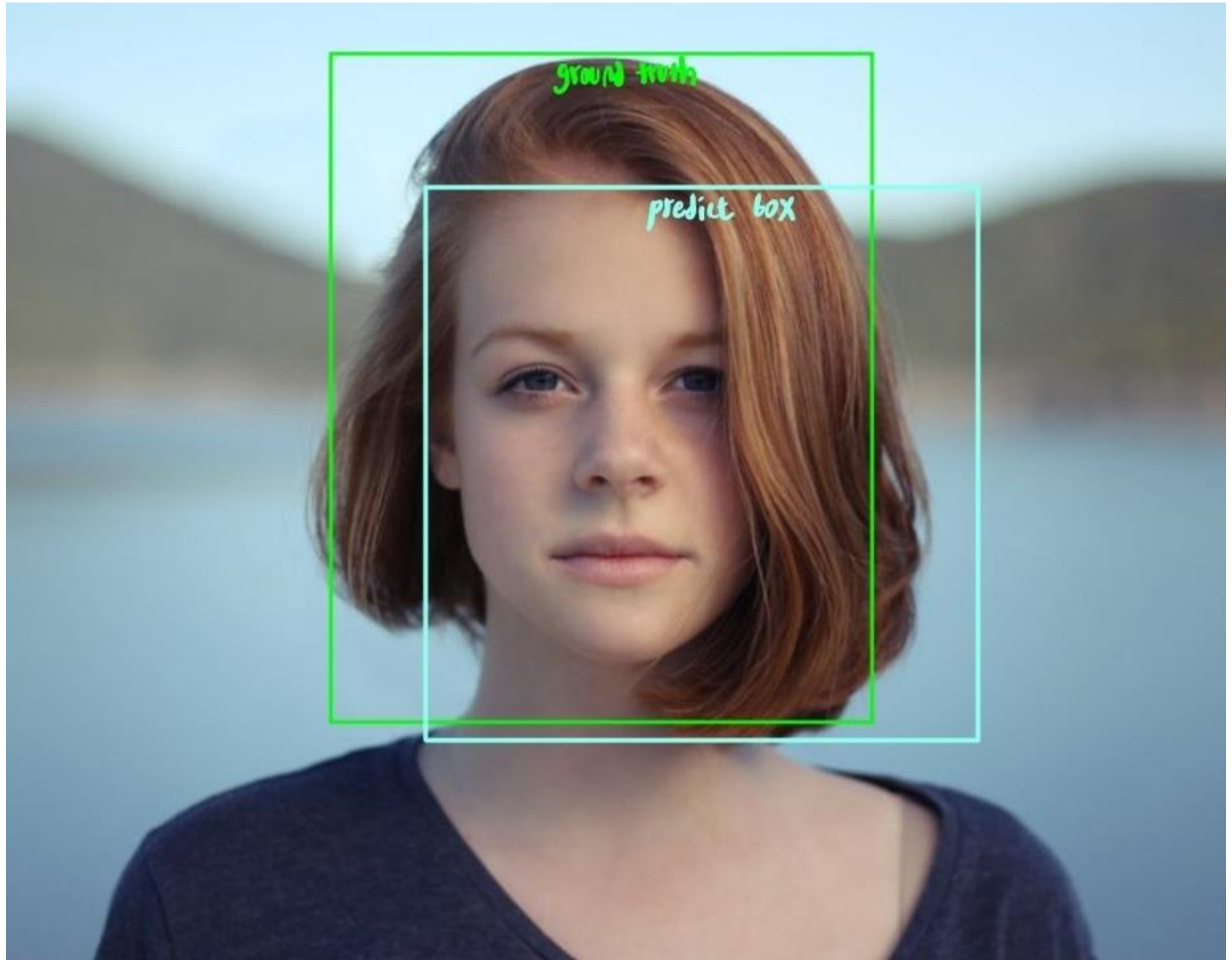
- Rectangular shapes
- Define the location and size of an object
- Commonly used as object detection and tracking
- Identifying and localizing objects
- Bounding box regression
- Precise object localization



https://www.cognex.com/ko-kr/blogs/deep-learning/research/overview-image-recognition-pascal-voc-challenge-2

Evaluation indicators

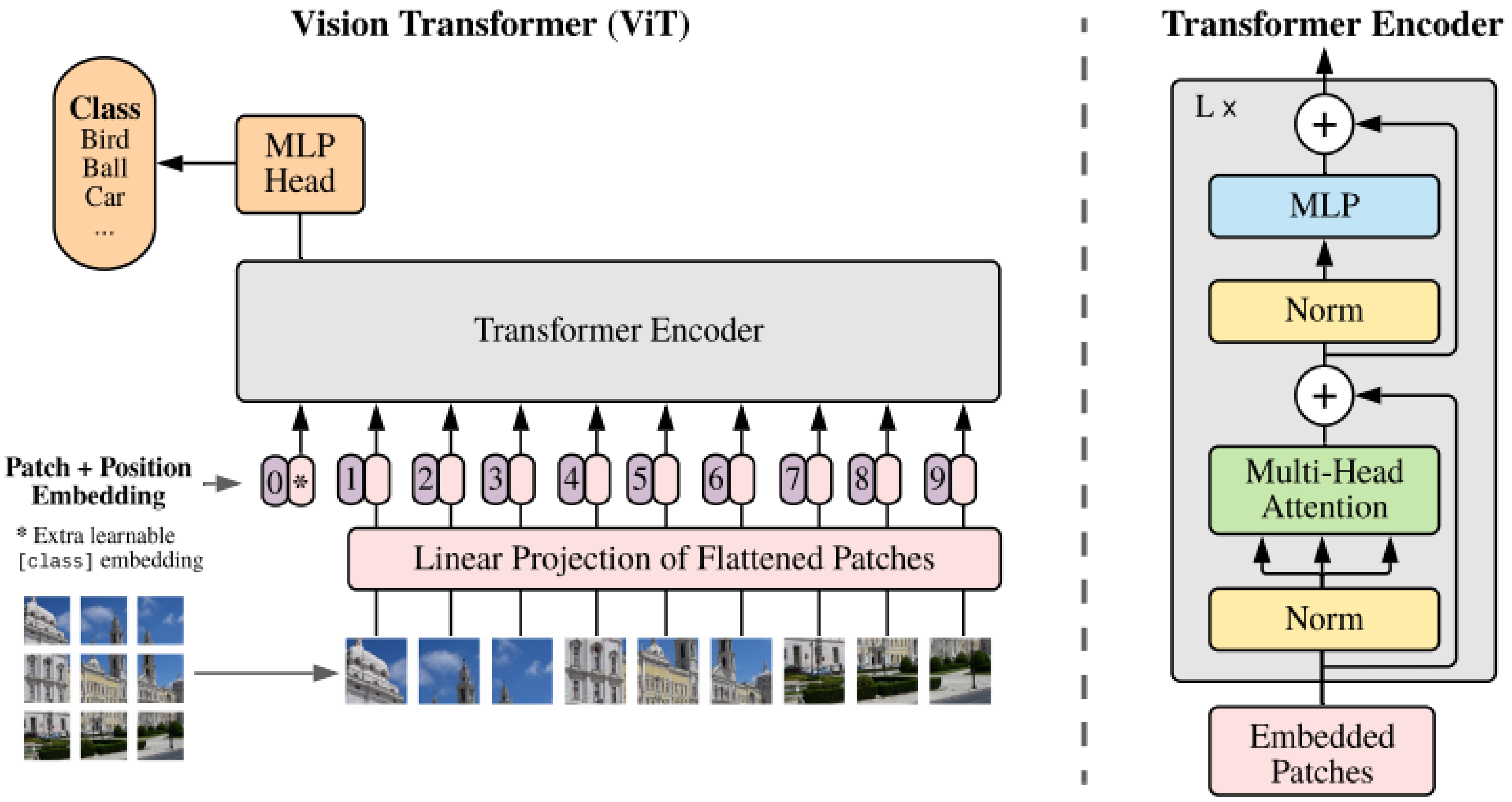




- https://lilianweng.github.io/posts/2017-12-31-object-recognition-part-3/
- https://velog.io/@qsdcfd/Theory-of-object-detection

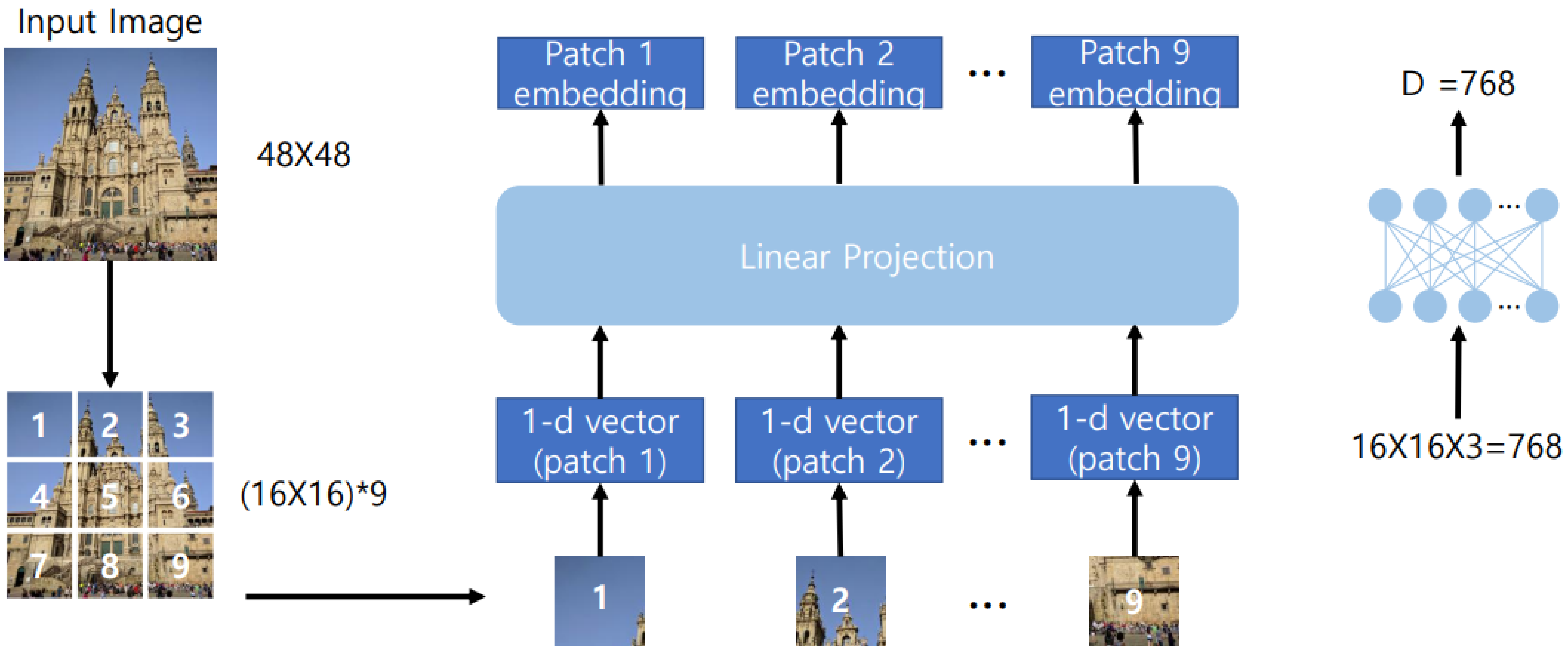
Transformer in Tracking

- Neural Network architecture
- Frequently used in NLP
- Successfully applied to computer vision tasks
- Utilize self-attention mechanisms
- ViTs are a specific variant of transformers designed
- ViTs have a hierarchical structure



https://velog.io/@changwoo7463/CV-10.-ViTVision-Transformer

Transformer in Tracking

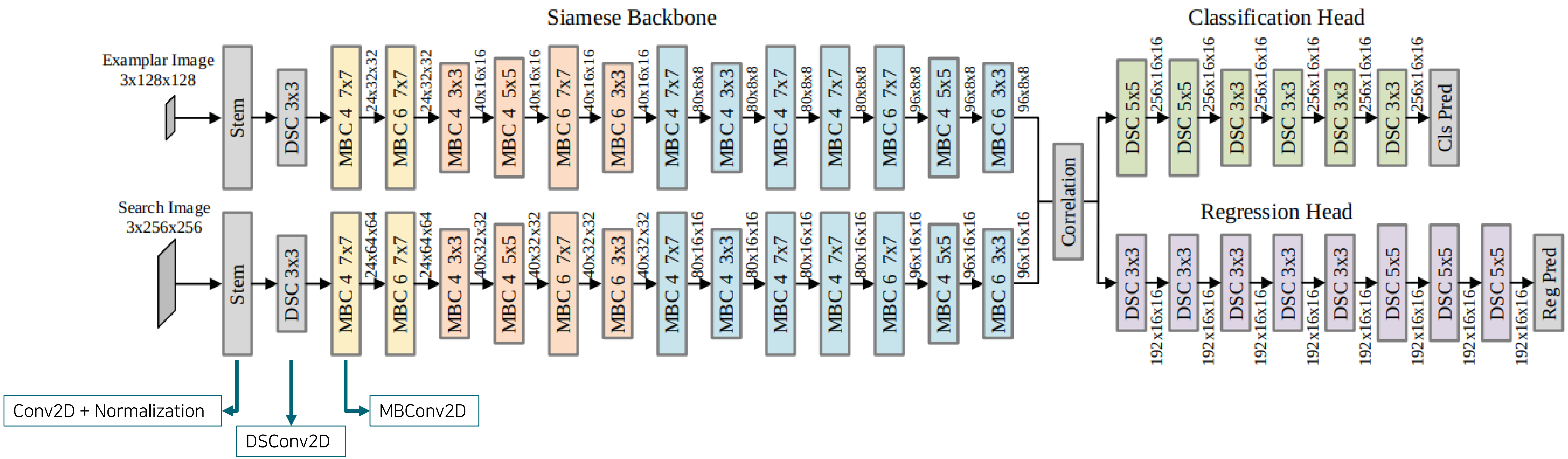


이채원, 임채연, 김도완

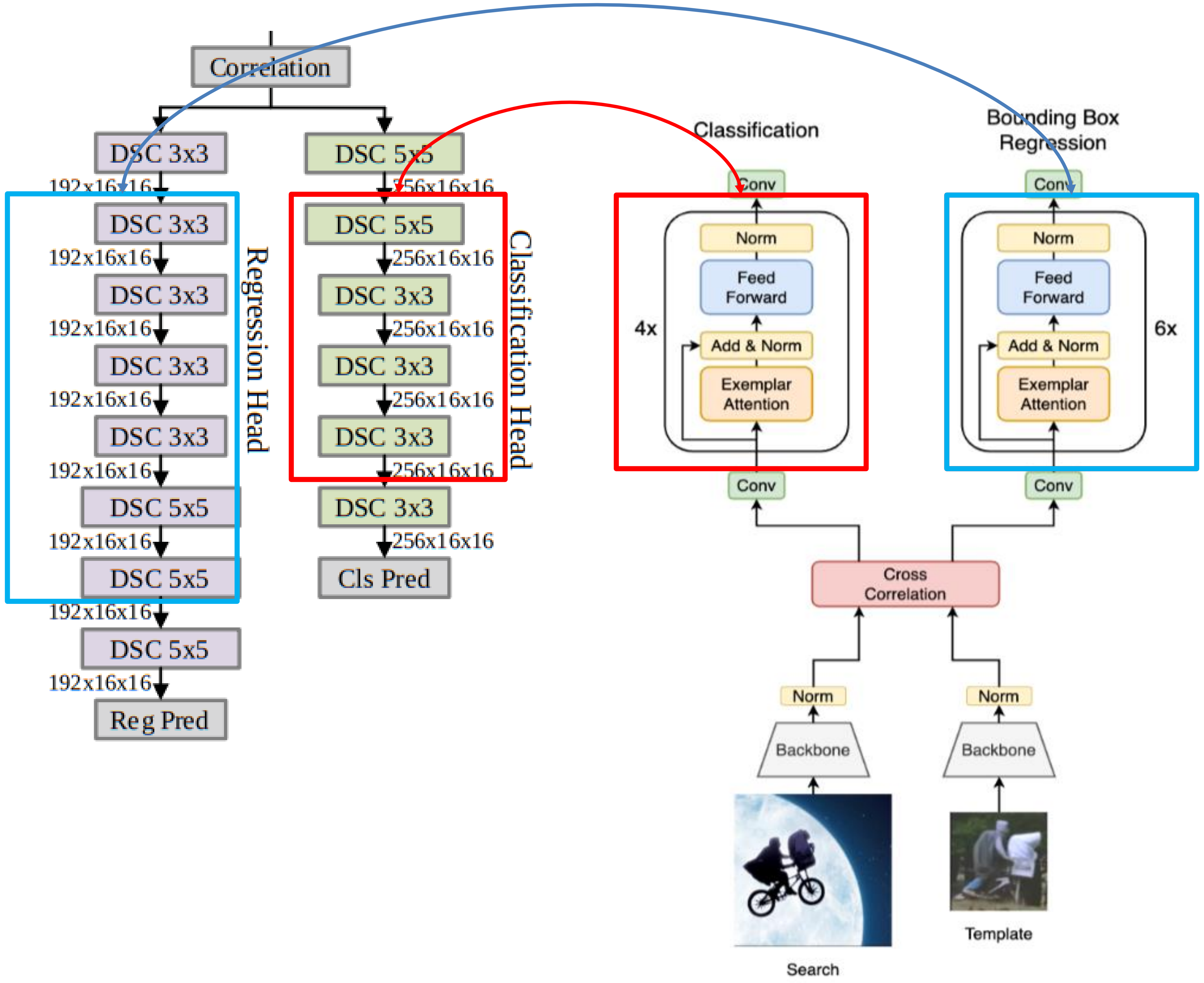
https://velog.io/@changwoo7463/CV-10.-ViTVision-Transformer

Light Tracker(Mobile)

LightTrack: Finding Lightweight Neural Networks for Object Tracking via One-Shot Architecture Search



Light Tracker(Mobile)



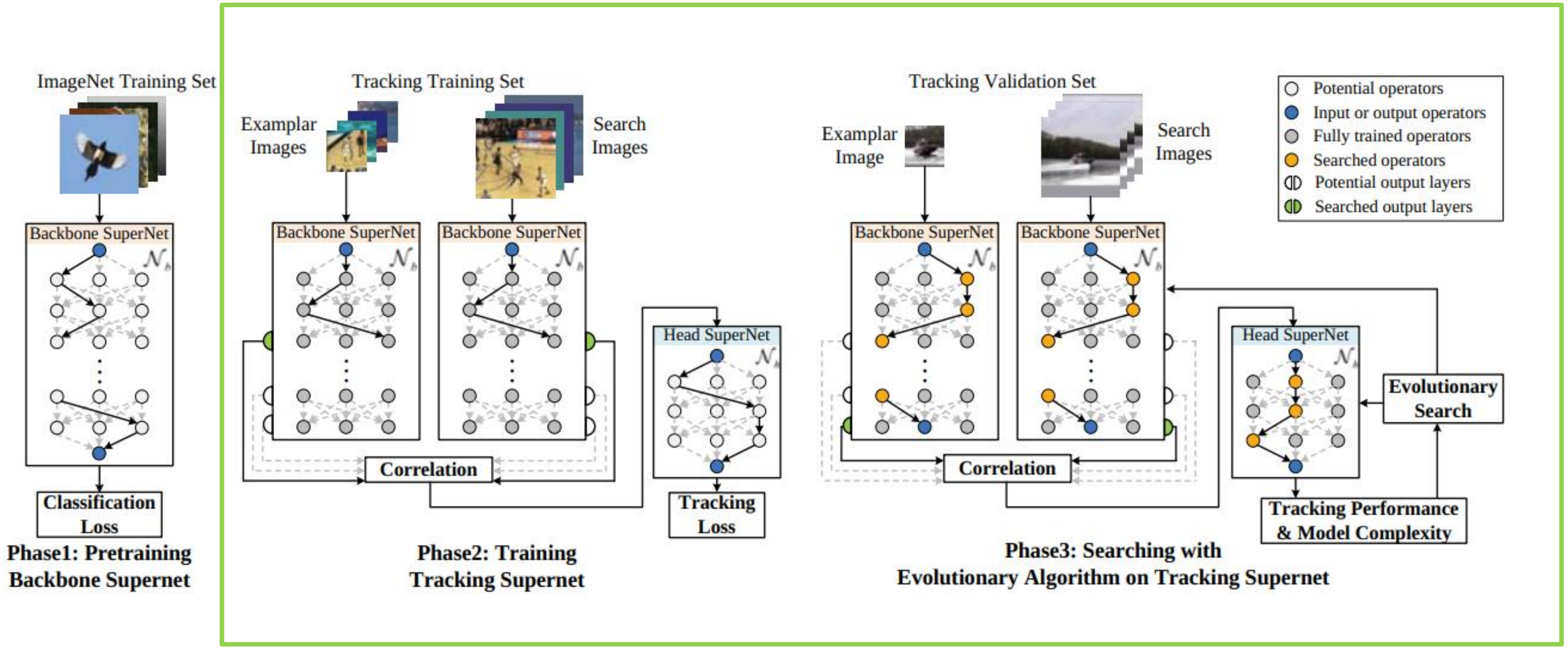
Replace Convolution to Exemplar Transformer

- Exemplar Transformer's operation = Convolution's operation
- Eliminate the need for retrain the backbone on ImageNet

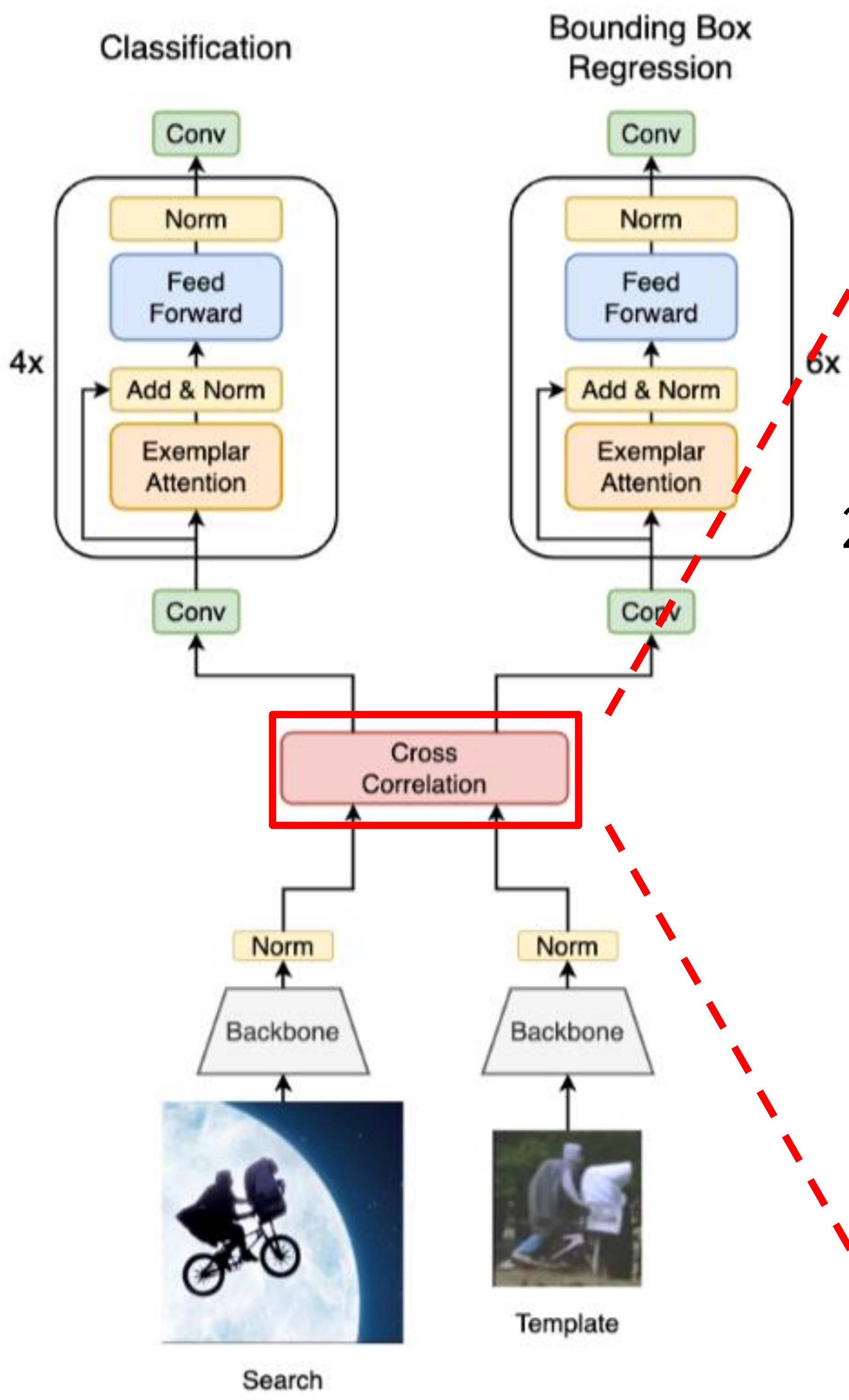
이채원, 임채연, 김도완

Light Tracker(Mobile)

Search pipeline of the proposed LightTrack



Cross-Correlation



1

```python
# extract features
zf = self.backbone_net(template)
xf = self.backbone_net(search)

# Batch Normalization before Corr
zf, xf = self.neck(zf, xf)

# pixelwise correlation
feat_dict = self.feature_fusor(zf, xf)
```

2

```python
# feature fusor
class Point_Neck_Mobile_simple_DP(nn.Module):
    def __init__(self, num_kernel_list=(256,64), cat=False, matrix=True, adjust=True, adj_channel=128):
        super(Point_Neck_Mobile_simple_DP, self).__init__()
        self.adjust = adjust
        '''Point-wise Correlation & Adjust Layer (unify the num of channels)'''
        self.pw_corr = torch.nn.ModuleList()
        self.adj_layer = torch.nn.ModuleList()
        for num_kernel in num_kernel_list:
            self.pw_corr.append(GroupPW(num_kernel, cat=cat, CA=True, matrix=matrix))
            self.adj_layer.append(nn.Conv2d(num_kernel, adj_channel, 1))

    def forward(self, kernel, search, stride_idx=None):
        '''stride_idx: 0 or 1. 0 represents stride 8. 1 represents stride 16'''
        if stride_idx is None:
            stride_idx = -1
        oup = {}
        corr_feat = self.pw_corr[stride_idx]([kernel], [search])
        #print("corr_feat shape: ", corr_feat.shape)
        #print(f'type of corr_feat: {type(corr_feat)}')
        if self.adjust:
            corr_feat = self.adj_layer[stride_idx](corr_feat)
        oup['cls'], oup['reg'] = corr_feat, corr_feat
        return oup
```

3

```python
def forward(self, z, x):
    z11 = z[0] # kernel (template)
    # print(z11.size())
    x11 = x[0] # search
    # print(x11.size())
    '''pixel-wise correlation'''
    '''2020.09.16 Matrix Mul Version'''
    if self.matrix:
        re11 = pixel_corr_mat(z11, x11)
    else:
        re11 = pixel_corr(z11, x11)
    if self.CA:
        '''channel attention'''
        s = self.CA_layer(re11)
        if self.cat:
            return torch.cat([s,x11],dim=1)
        else:
            return s
    else:
        return re11
```

4

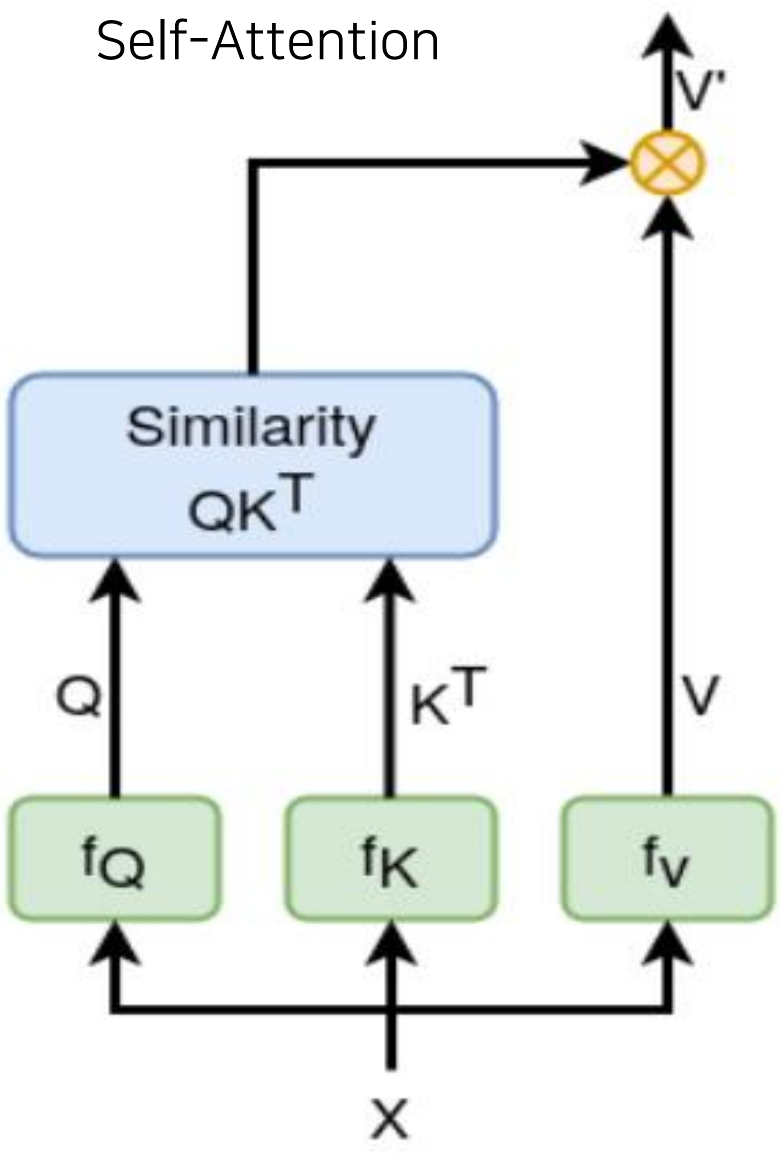
```python
def pixel_corr_mat(z, x):
    '''
    z is kernel ([32, 96, 8, 8])
    x is search ([32, 96, 16, 16])

    z -> (32, 64, 96)
    x -> (32, 96, 256)
    '''
    b, c, h, w = x.size()
    z_mat = z.view((b,c,-1)).transpose(1,2) # (b,64,c)
    x_mat = x.view((b,c,-1)) # (b,c,256)
    # matmul = (b,64,256) == every pixel of z is correlated with every pixel of x
    return torch.matmul(z_mat, x_mat).view((b,-1,h,w)) # (32, 64, 16, 16)
```

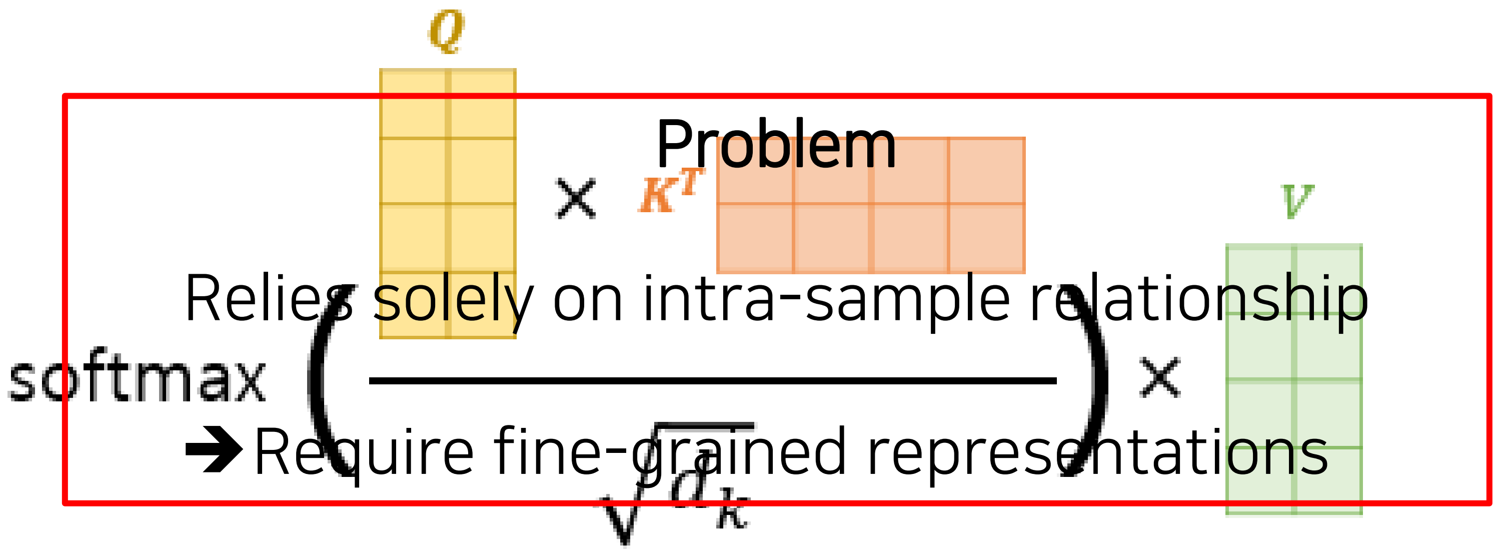
Calculate correlation through Dot product

이채원, 임채연, 김도완

Self Attention



$$\text{softmax}\left(\frac{\overbrace{(xW_Q)}^{f_Q(x)}\overbrace{(W_K^T x^T)}^{f_K(x)}}{\underbrace{\sqrt{d_k}}_{\text{constant}}}\right)\overbrace{(xW_V)}^{f_V(x)}$$



Problem

Relies solely on intra-sample relationship

➔ Require fine-grained representations

StandardTransformer

Machine Translation

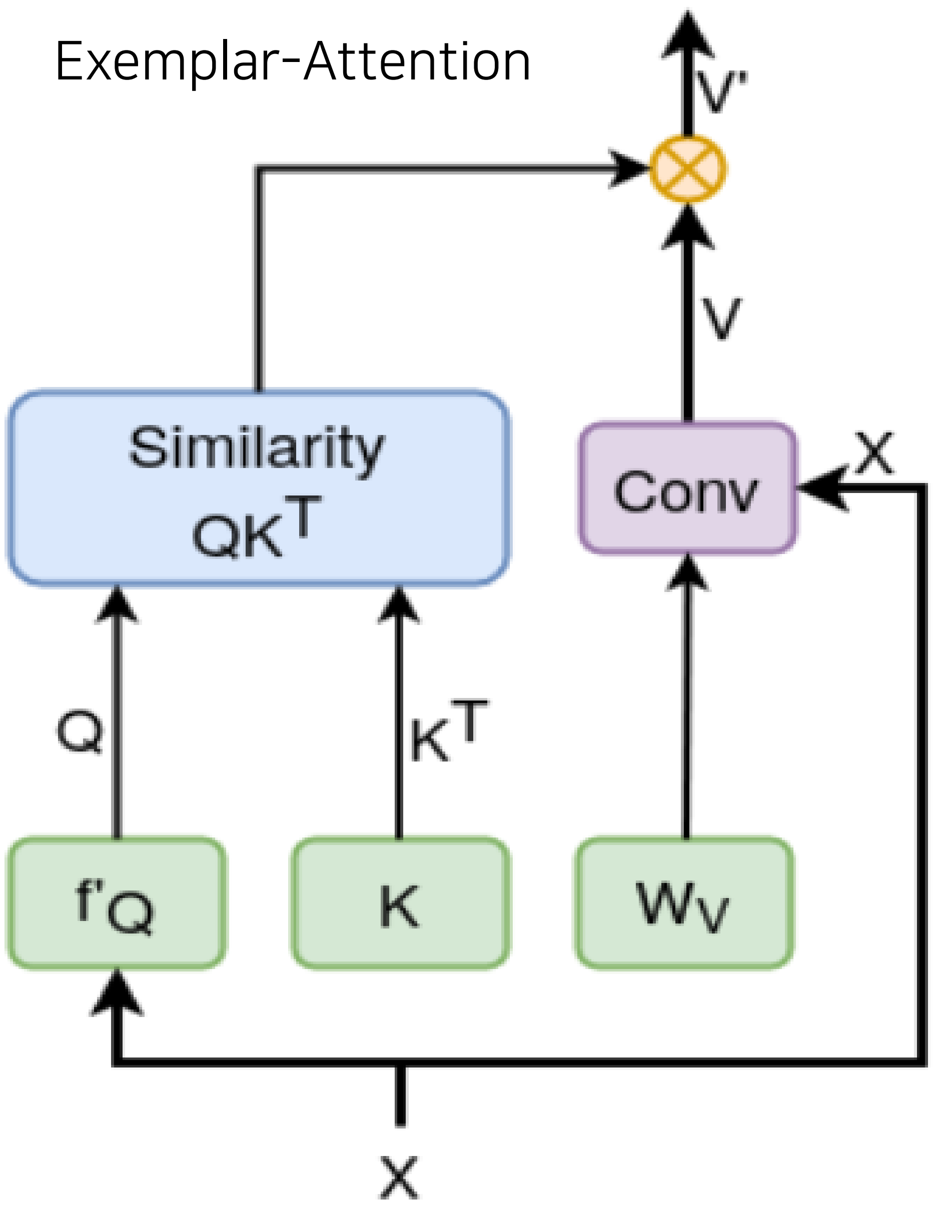
- Every feature represents a specific word or token

Vision

- Adjacent spatial representation often correspond to the same object

➡ What about use only one Query(Global Query)?

Exemplar Attention



$$\text{softmax}\left(\frac{\overbrace{(\Psi_S(X)W_Q)}^{f_Q(x)}\overbrace{(\hat{W}_K^T)}^{f_K(\cdot)}}{\underbrace{\sqrt{d_k}}_{\text{constant}}}\right)\overbrace{(W_V \circledast X)}^{f_V(x)}$$

이채원, 임채연, 김도완

Exemplar Attention

$\Psi_S(X)$: AvgPooling + Flatten

$$\underline{Q} = \Psi_S(X) W_Q \in \mathbb{R}^{S^2 \times D_{QK}}$$

One global Query

Compressed Representation X ($\Psi_S(X)$) -> Identify the object(W_Q)

One Global Query -> Decreasing the Complexity

Examplar Attention

Input shape = (B, 256, 16, 16)

```
self.global_pooling = nn.AdaptiveAvgPool2d(seq_red)
```

1

(B, 256, 1, 1)

```
self.flatten = nn.Flatten(start_dim = 2)
```

(B, 256, 1)

```
self.fc1 = nn.Linear(c_dim, hidden_dim)
```

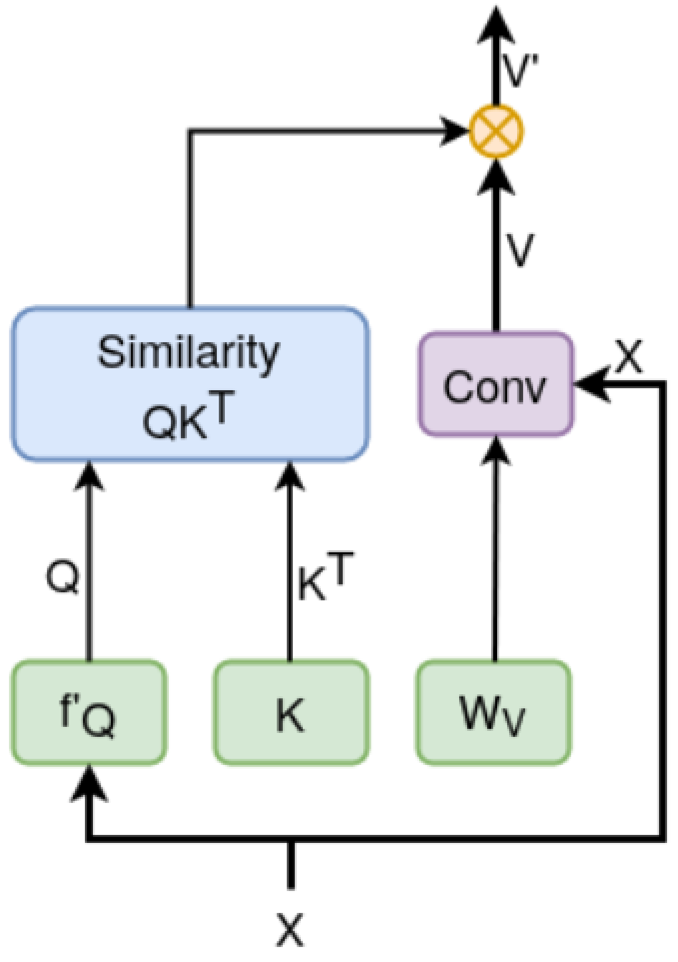
256 128

permute(0, 2, 1) -> (B, 1, 128)

```
self.act = nn.ReLU(inplace=False)
```

(B, 1, 128)

Exemplar Attention



Key & Value capture Object information

* E = 4

$$K = \hat{W}_K \in \mathbb{R}^{E \times D_{QK}}$$

- Independent of the Input

(Can be applied to various inputs)

* E = 4

$$V = W_V \circledast X \in \mathbb{R}^{E \times H \times W \times D_V}$$

- Use Convolutional Operation

(Good for visual pattern identification)

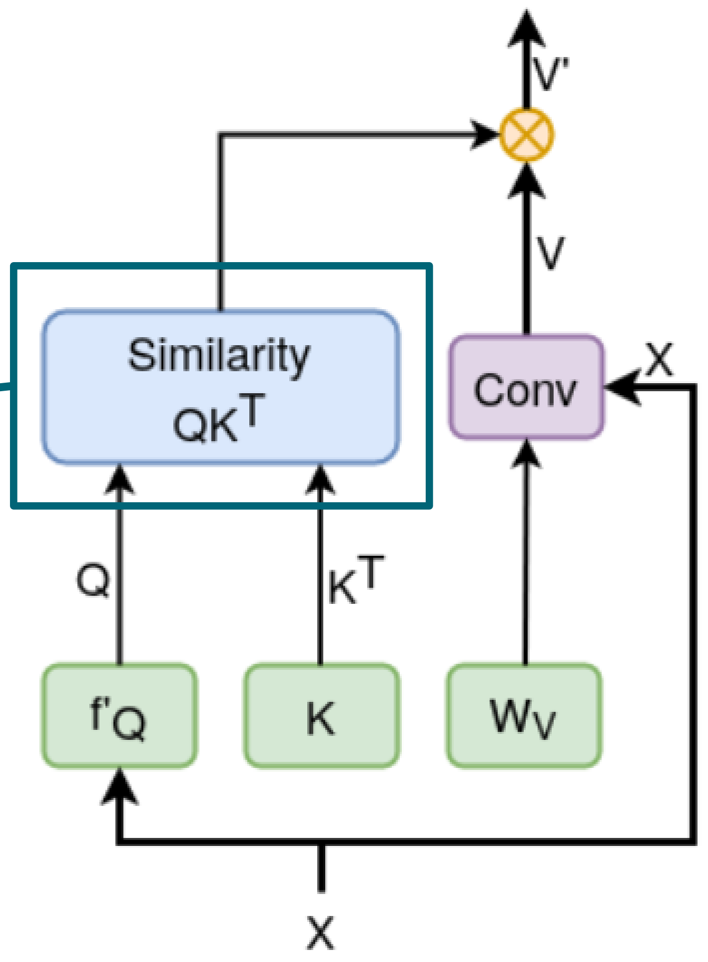
Exemplar Attention

Similarity QK

```
qk = torch.matmul(q, self.K.T)

if self.sm_norm:
    qk = 1/math.sqrt(d_k) * qk

# apply softmax
attn = self.softmax(qk/self.temperature) # -> [batch_size, e_exemplars]
```



Exemplar Attention

Depth-wise Conv

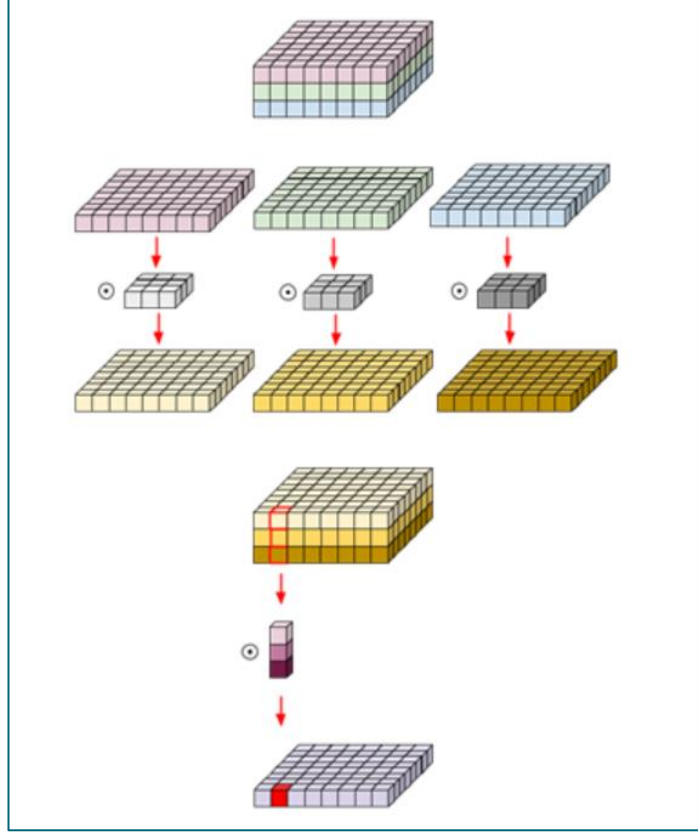
```
# apply convolution
x = F.conv2d(
    x, dw_weight, bias=None, stride=self.dw_stride, padding=self.dw_padding,
    groups=self.dw_groups * B)

x = x.permute([1, 0, 2, 3]).view(B, self.out_channels, x.shape[-2], x.shape[-1])
x = self.dw_bn(x)      # Normalization
x = self.dw_act(x)     # ReLu
```

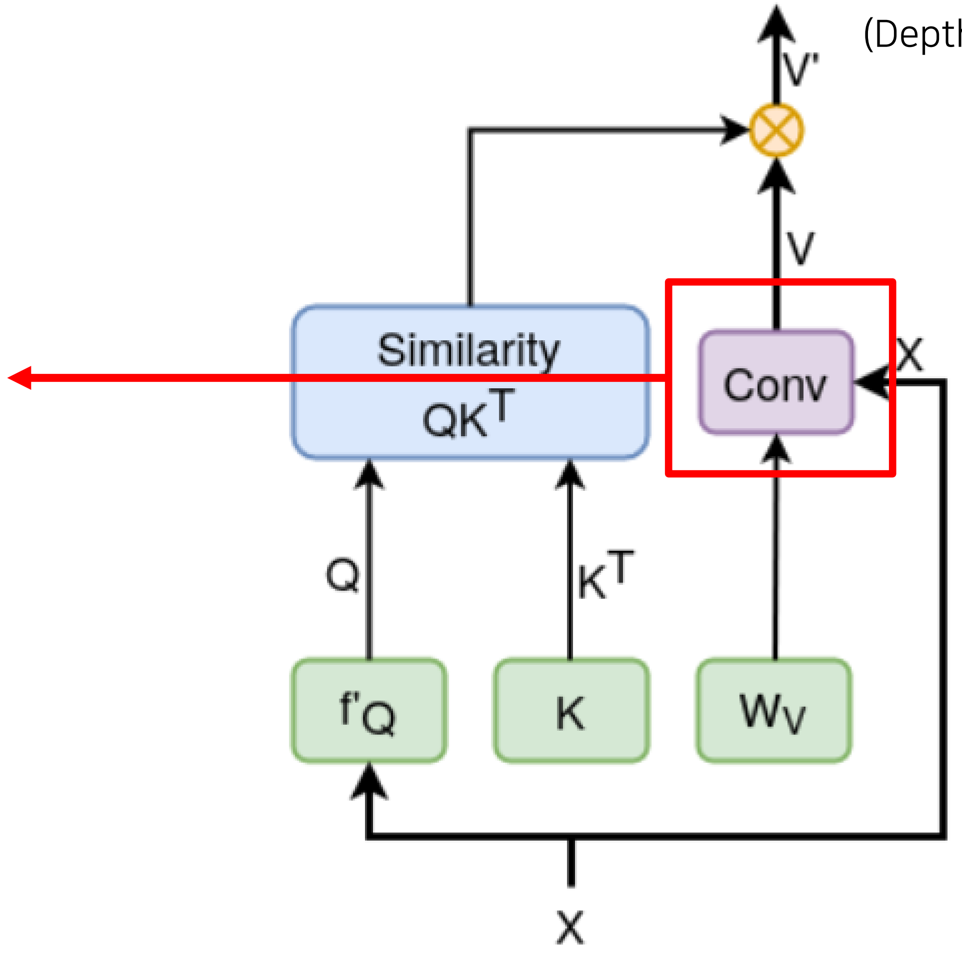
Point-wise Conv

```
# apply convolution
x = F.conv2d(
    x, pw_weight, bias=None, stride=self.pw_stride, padding=self.pw_padding,
    groups=self.pw_groups * B)

x = x.permute([1, 0, 2, 3]).view(B, self.out_channels, x.shape[-2], x.shape[-1])
x = self.pw_bn(x)     # Normalization
x = self.pw_act(x)    # ReLu
```



(Depth-wise Separable Convolution)



Exemplar Attention

$$A(x) = \text{softmax}\left(\frac{\overbrace{(\Psi_S(X)W_Q)}^{f_Q(x)}\overbrace{(\hat{W}_K^T)}^{f_K(\cdot)}}{\underbrace{\sqrt{d_k}}_{\text{constant}}}\right)\overbrace{(W_V \circledast X)}^{f_V(x)},$$

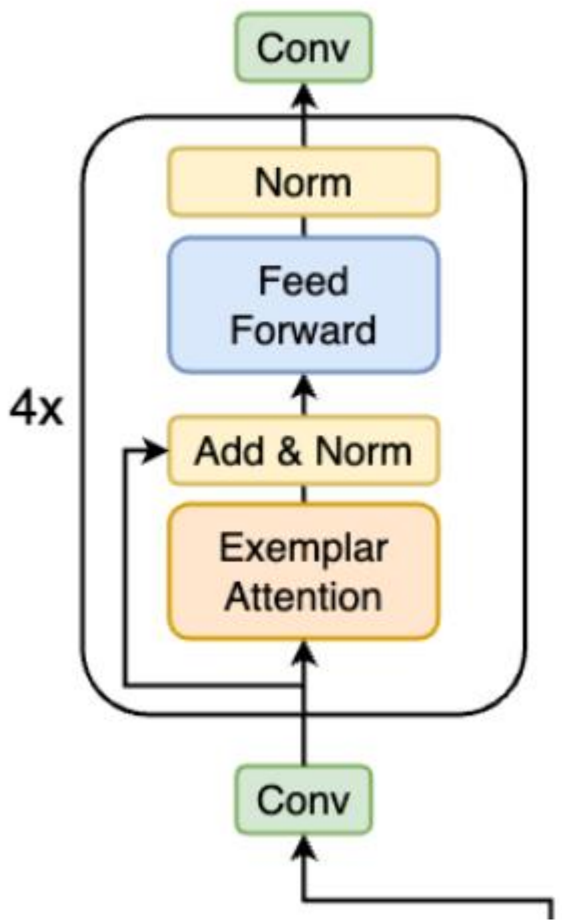
$$\Downarrow$$

$$A(x) = \left[\text{softmax}\left(\frac{(\Psi_S(X)W_Q)(\hat{W}_K^T)}{\sqrt{d_k}}\right)W_V\right] \circledast X$$

Exemplar Attention ➡ Exemplar Representation

Classification & Regression

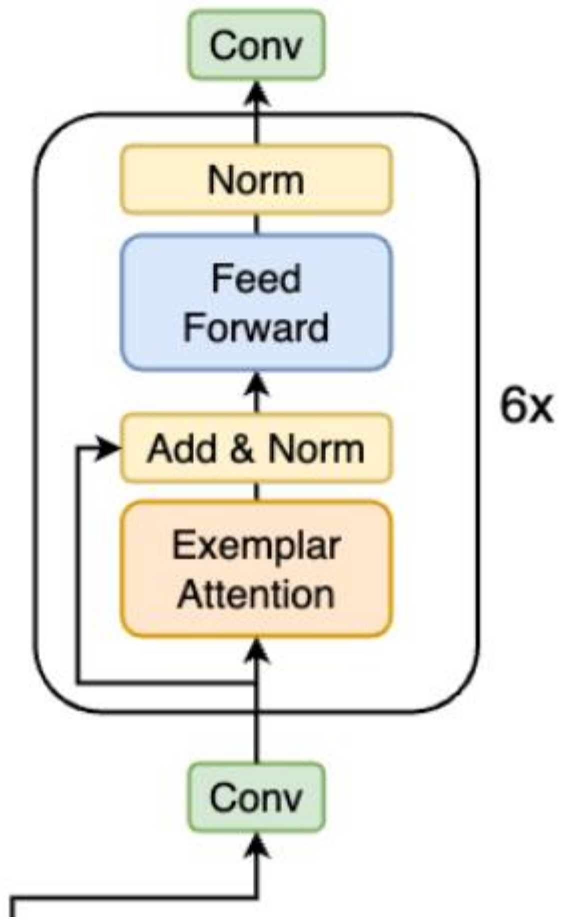
Classification → Foreground or Background



Loss: Binary Cross Entropy

$$\mathcal{L}_r = -\sum_j p_r^* log(p_r) + (1 - p_r^*) log(1 - p_r)$$

Bounding Box Regression → Predict the distance to all four side



Loss: IoU

$$\mathcal{L}_{reg} = -\sum_i ln(IoU(p_{reg}, T^*))$$

Total Loss: $\mathcal{L} = \mathcal{L}_{reg} + \lambda_1 \mathcal{L}_r$

Training

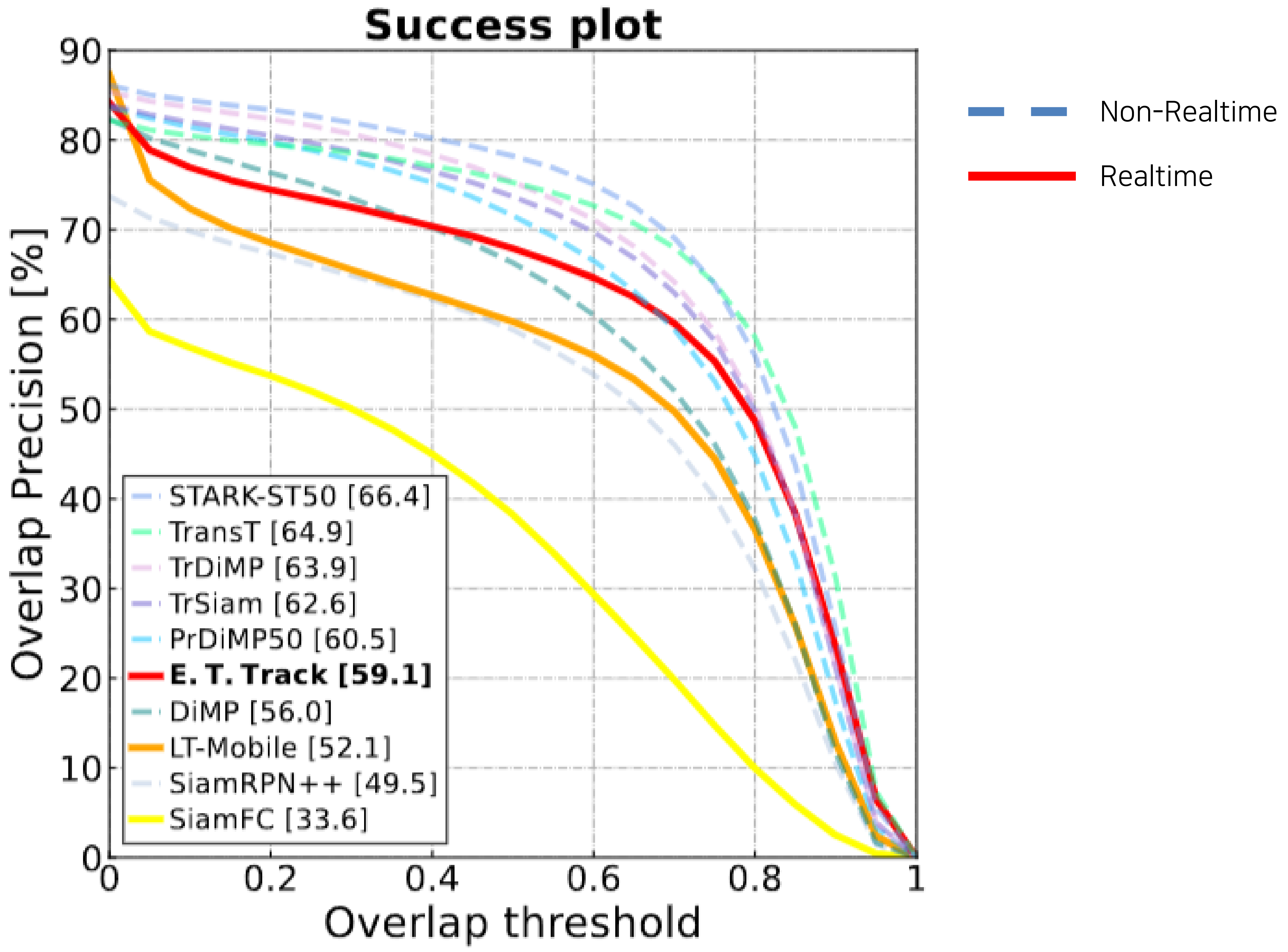
Hyper Parameter	Value
Optimizer	SGD(momentum=0.9)
Epoch	50 (first 10 epoch backbone parameters frozen)
Weight Decay	1e-4(0.0001)[L2 regulation]
LR Scheduler	Increasing 2e-2(0.02) -> 1e-1(0.1) first 5 epoch / Decreasing 1e-1(0.1) -> 2e-4(0.0002)

Training

DataSet	Sampling range
LaSOT	Select 2 frames within 100 range of frames
TrackingNet	Select 2 frames within 100 range of frames
GOT10k	Select 2 frames within 30 range of frames
COCO	Select 1 frame within 1 range of frame

Training

LaSOT



이채원, 임채연, 김도완

Training

	non-realtime									realtime		
	ATOM [10]	SiamRPN++ [30]	DiMP-50 [5]	PrDiMP-50 [12]	SiamR-CNN [46]	TransT [8]	TrDiMP [48]	TrSiam [48]	STARK-ST50 [53]	ECO [11]	LT-Mobile [54]	**E.T.Track (Ours)**
NFS	58.4	50.2	62	63.5	63.9	65.7	66.5	65.8	66.4	46.6	55.3	**59.0**
UAV-123	64.2	61.3	65.3	68	64.9	69.4	67.5	67.4	68.8	51.3	**62.5**	62.3
OTB-100	66.9	69.6	68.4	69.6	70.1	69.1	71.1	70.8	67.3	64.3	66.2	**67.8**
CPU Speed	20	15	15	15	15	5	6	6	9	25	47	47

Scores of NFS, UAV-123, OTB-100

Ablation Study

Conv	Att	FFN	T-Cond.	NFS	OTB-100	LaSOT
✓				55.3	66.2	52.1
	✓			56.6	65.8	53.6
	✓	✓		58	67.3	59.1
	✓	✓	✓	59.0	66.9	57.9

Configuration of example transformer

Use Examplar Attention + FFN

Ablation Study

	Conv	1-Ex	4-Ex	16-Ex
NFS	55.3	57.6	58.0	58.0
OTB-100	66.2	66.5	67.3	66.1
LaSOT	52.1	57.2	59.1	57.4

Performance difference of Examplar Transformer number

	S=1	S=2	S=4
NFS	59.0	46.6	46.7
OTB-100	67.8	55.5	57.5
LaSOT	59.1	43.7	42.6

Performance difference of Q number

Ablation Study

	ShuffleNet [60]		MobileNetV3 [22]		ResNet-18 [20]		LT-Mobile [54]	
Conv	✓		✓		✓		✓	
E.T. (Ours)		✓		✓		✓		✓
NFS	54.9	56.2	56.8	56.8	55.8	57.3	55.3	59.0
OTB-100	61.3	61.8	64.5	65.3	65.3	65.7	66.2	67.8
LaSOT	48.6	49.8	52.1	52.7	55.9	56.5	52.1	59.1

Comparison of Convolution and Examplar Transformer

The performance of Examplar is better than that of Conv

Ablation Study

	Conv [54]	Standard [45]	Clustered [47]	Linear [26]	Local [38]	Swin [33]	E.T.Track (Ours)
		Other Transformer					Examplar Transformer
NFS	55.3	55.3	57.5	55.8	55.8	55.4	59.0
OTB-100	66.2	65.3	67.5	65.4	64.8	64.2	67.8
LaSOT	52.1	54.2	56.5	53.5	53.4	56.9	59.1

Examplar Transformer has good performance

Conclusion

Exemplar transformer

- Single-object visual tracking performance enhancement

E.T.Track

- Real-time tracking suitability

Progress

Evaluated the learned model using the pytracking library(UAV-123 DataSet)

Console 1/A ×

```
Tracker: et_tracker et_tracker 0 ,  Sequence: uav_boat5
checkpoint epoch provided: 35
checkpoint path:  ./checkpoints/et_tracker\checkpoint_e35.pth
loading model from:  ./checkpoints/et_tracker\checkpoint_e35.pth
loading the checkpoint strict: True
model initializing successful
tracker weight style: regular
FPS: 13.888883737872115
```

6 hours

541	296	188	130
536	296	186	130
540	296	185	130
539	296	182	130
539	295	181	130
540	296	180	130

Windows compatibility issue with Python's multiprocessing

Progress (UAV-123)

Progress (WebCam)

Progress (WebCam)

Pytracking/run_webcam.py

```
def run_webcam(tracker_name, tracker_param, debug=None, visdom_info=None):
    """Run the tracker on your webcam.
    args:
        tracker_name: Name of tracking method.
        tracker_param: Name of parameter file.
        debug: Debug level.
        visdom_info: Dict optionally containing 'use_visdom', 'server' and 'port' for Visdom visualization.
    """
    visdom_info = {} if visdom_info is None else visdom_info
    tracker = Tracker(tracker_name, tracker_param)
    tracker.run_webcam(debug, visdom_info)


def main():
    parser = argparse.ArgumentParser(description='Run the tracker on your webcam.')
    # parser.add_argument('tracker_name', type=str, help='Name of tracking method.')
    # parser.add_argument('tracker_param', type=str, help='Name of parameter file.')
    parser.add_argument('--debug', type=int, default=0, help='Debug level.')
    parser.add_argument('--use_visdom', type=bool, default=True, help='Flag to enable visdom')
    parser.add_argument('--visdom_server', type=str, default='127.0.0.1', help='Server for visdom')
    parser.add_argument('--visdom_port', type=int, default=8097, help='Port for visdom')

    args = parser.parse_args()

    visdom_info = {'use_visdom': args.use_visdom, 'server': args.visdom_server, 'port': args.visdom_port}
    # run_webcam(args.tracker_name, args.tracker_param, args.debug, visdom_info)

    # 실행하면 바로 et_tracker 실행되게
    run_webcam('et_tracker', 'et_tracker', args.debug, visdom_info)
```

WebCam tracking use ET-Track

Progress (WebCam)

Pytracking/evalution/tracker.py

```
while True:
    # Capture frame-by-frame
    ret, frame = cap.read()
    frame_disp = frame.copy()

    info = OrderedDict()
    info['previous_output'] = prev_output

    if ui_control.new_init:
        ui_control.new_init = False
        init_state = ui_control.get_bb()

        info['init_object_ids'] = [next_object_id, ]
        info['init_bbox'] = OrderedDict({next_object_id: init_state})
        sequence_object_ids.append(next_object_id)

        next_object_id += 1

    # Draw box
    if ui_control.mode == 'select':
        cv.rectangle(frame_disp, ui_control.get_tl(), ui_control.get_br(), (255, 0, 0), 2)

    if len(sequence_object_ids) > 0:
        info['sequence_object_ids'] = sequence_object_ids
        out = tracker.track(frame, info)
        prev_output = OrderedDict(out)

        if 'segmentation' in out:
            frame_disp = overlay_mask(frame_disp, out['segmentation'])

        if 'target_bbox' in out:
            # 추적된 객체가 있을 경우
            # ettrack은 SOT 여서 하나로 고정시켜줌
            state = [int(bbox) for bbox in out['target_bbox']]
            cv.rectangle(frame_disp, (state[0], state[1]), (state[2] + state[0], state[3] + state[1]),
                         (255, 0, 0), 5)
```

Capture Frame

Draw Init box

Track now frame

Draw bounding box

Progress (WebCam)

Pytracking/tracker/et_tracker/et_tracker.py

```
if debug:
    target_pos, target_sz, _, cls_score = self.update(x_crop, target_pos, target_sz * scale_z,
                                                       window, scale_z, p, debug=debug, writer=writer)
    state['cls_score'] = cls_score
else:
    target_pos, target_sz, _ = self.update(x_crop, target_pos, target_sz * scale_z,
                                           window, scale_z, p, debug=debug, writer=writer)

target_pos[0] = max(0, min(state['im_w'], target_pos[0]))
target_pos[1] = max(0, min(state['im_h'], target_pos[1]))
target_sz[0] = max(10, min(state['im_w'], target_sz[0]))
target_sz[1] = max(10, min(state['im_h'], target_sz[1]))

#print("cropped x shape: ", x_crop.shape)
#print("target pos shape: ", target_pos.shape)
#print("target size shape: ", target_sz.shape)
#print("target size: ", target_sz)

# TODO: compute appropriate bounding box in x,y,w,h format (?) and return it
location = cxy_wh_2_rect(target_pos, target_sz)
```

Predict next position & object size

Calculate bounding box (x,y,w,h)

Progress (WebCam)

Tracking environment

Tracking Result

이채원, 임채연, 김도완

24

감사합니다!

질문이 있으신가요?