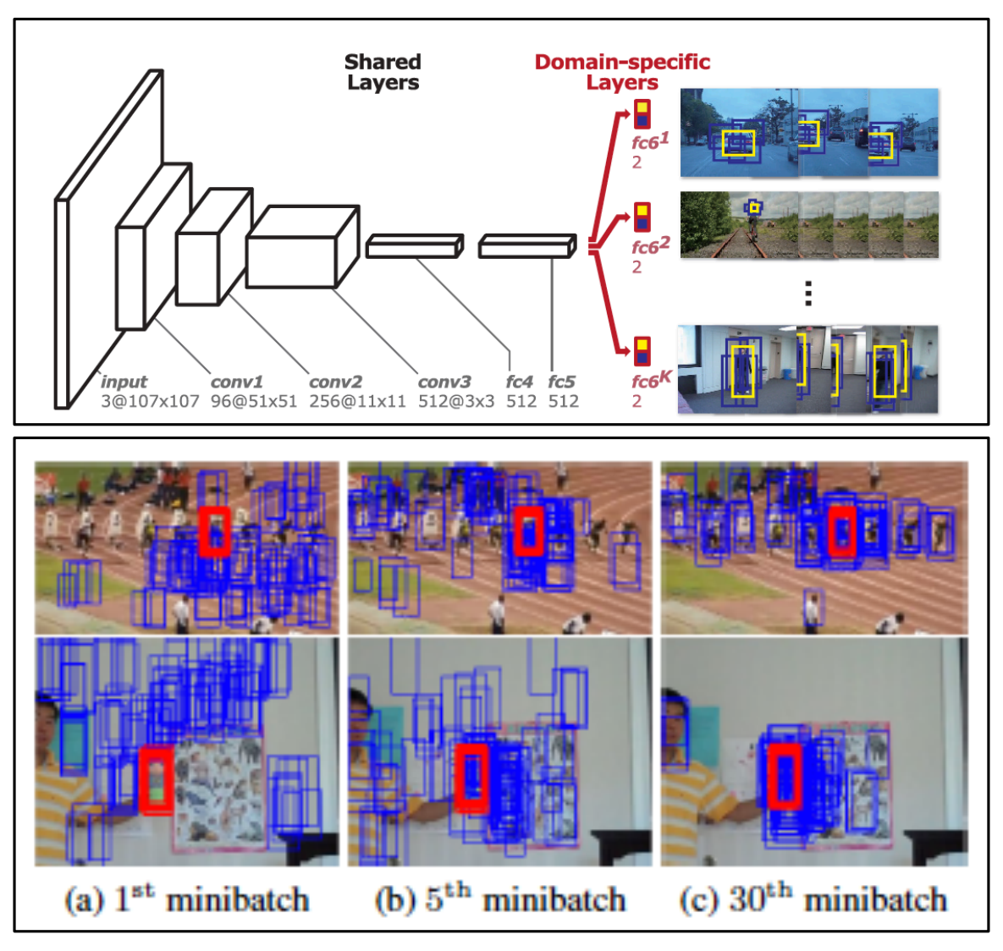
Embedded *MDNet*

Mobile Embedded Multi-Domain Convolutional Neural Networks for Visual Tracking

Jeongbin Choe
Advisor: Prof. Bohyung Han, CV Lab
Dept. Computer Science and Engineering, POSTECH

Introduction

- Visual Tracking
  - Locating, identifying and determining the dynamic configuration of moving objects
- Convolutional Neural Network (CNN)
  - Deep (Multi-layer) neural network
  - Convolution Layer & Pooling Layer
  - Applied to various computer vision tasks
- *MDNet*
  - Multi-Domain Network for Visual Tracking
  - Consists of shared CNN layers and multiple branches of domain-specific layers
  - MATLAB based framework
  - The winner of the VOT2015 Challenge



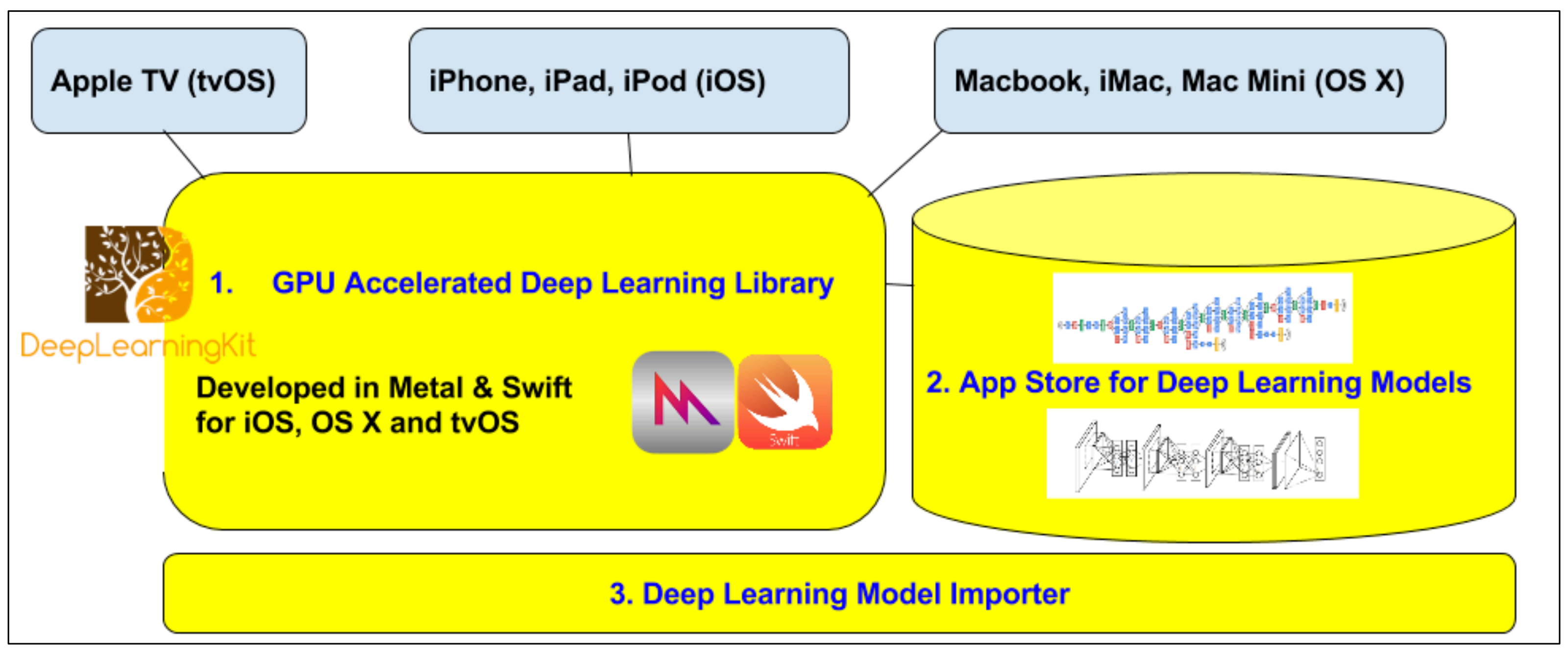
(a) 1^{st} minibatch (b) 5^{th} minibatch (c) 30^{th} minibatch

- *MDNet* performed 1 fps with 8 cores Intel *Xeon* CPU + NVIDIA *Tesla*
- What about on mobile devices?

- Project Goal
  - Port *MDNet* to Apple iPhone 6+
    - Apple A8 processor Dual-core 1.4 GHz Typhoon, PowerVR GX6450
  - Optimize its performances
  - Implement *MDNet* based real-time learning and object tracking application

Technical Resources

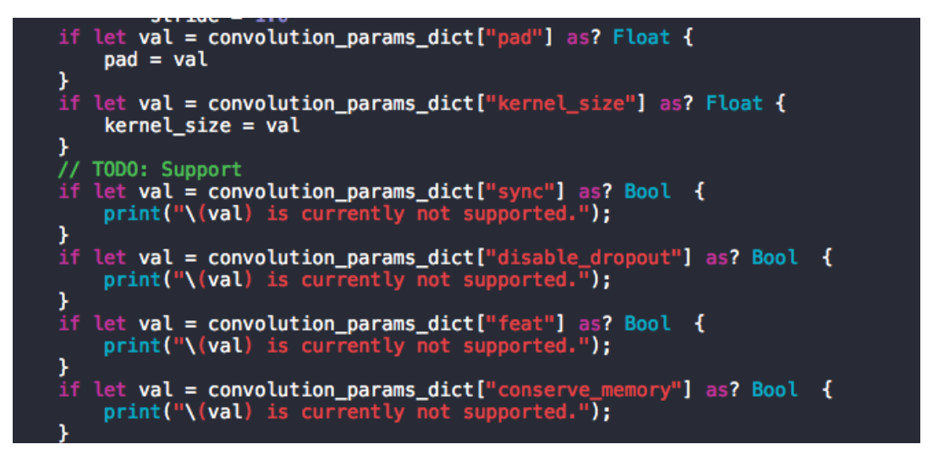
- *DeepLearningKit*
  - *Caffe* based open-sourced deep learning framework for iOS
  - Written in Swift and Metal
  - Supports CNN


Apple TV (tvOS)
iPhone, iPad, iPod (iOS)
Macbook, iMac, Mac Mini (OS X)
DeepLearningKit
1. GPU Accelerated Deep Learning Library
Developed in Metal & Swift for iOS, OS X and tvOS
2. App Store for Deep Learning Models
3. Deep Learning Model Importer

Issues

- Open Source *DeepLearningKit* hasn't updated for 5 months
  - Still lots of parameters are not supported
  - Has memory issues to use Metal
  - Hard to port full-*MDNet* to iPhone

fixed access bug in api for init and load func	5 months ago
fixed access bug in api for init and load func	5 months ago
fixed access bug in api for init and load func	5 months ago
fixed access bug in api for init and load func	5 months ago

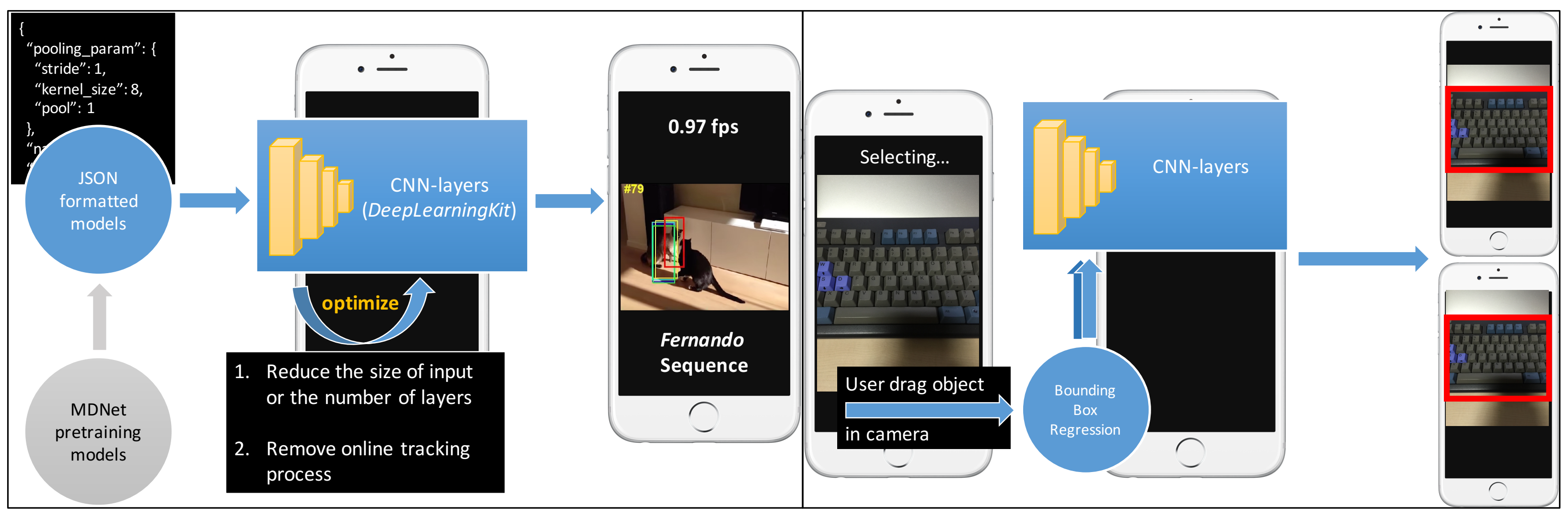


```
if let val = convolution_params_dict["pad"] as? Float {
    pad = val
}
if let val = convolution_params_dict["kernel_size"] as? Float {
    kernel_size = val
}
// TODO: Support
if let val = convolution_params_dict["sync"] as? Bool  {
    print("\(val) is currently not supported.");
}
if let val = convolution_params_dict["disable_dropout"] as? Bool  {
    print("\(val) is currently not supported.");
}
if let val = convolution_params_dict["feat"] as? Bool  {
    print("\(val) is currently not supported.");
}
if let val = convolution_params_dict["conserve_memory"] as? Bool  {
    print("\(val) is currently not supported.");
}
```

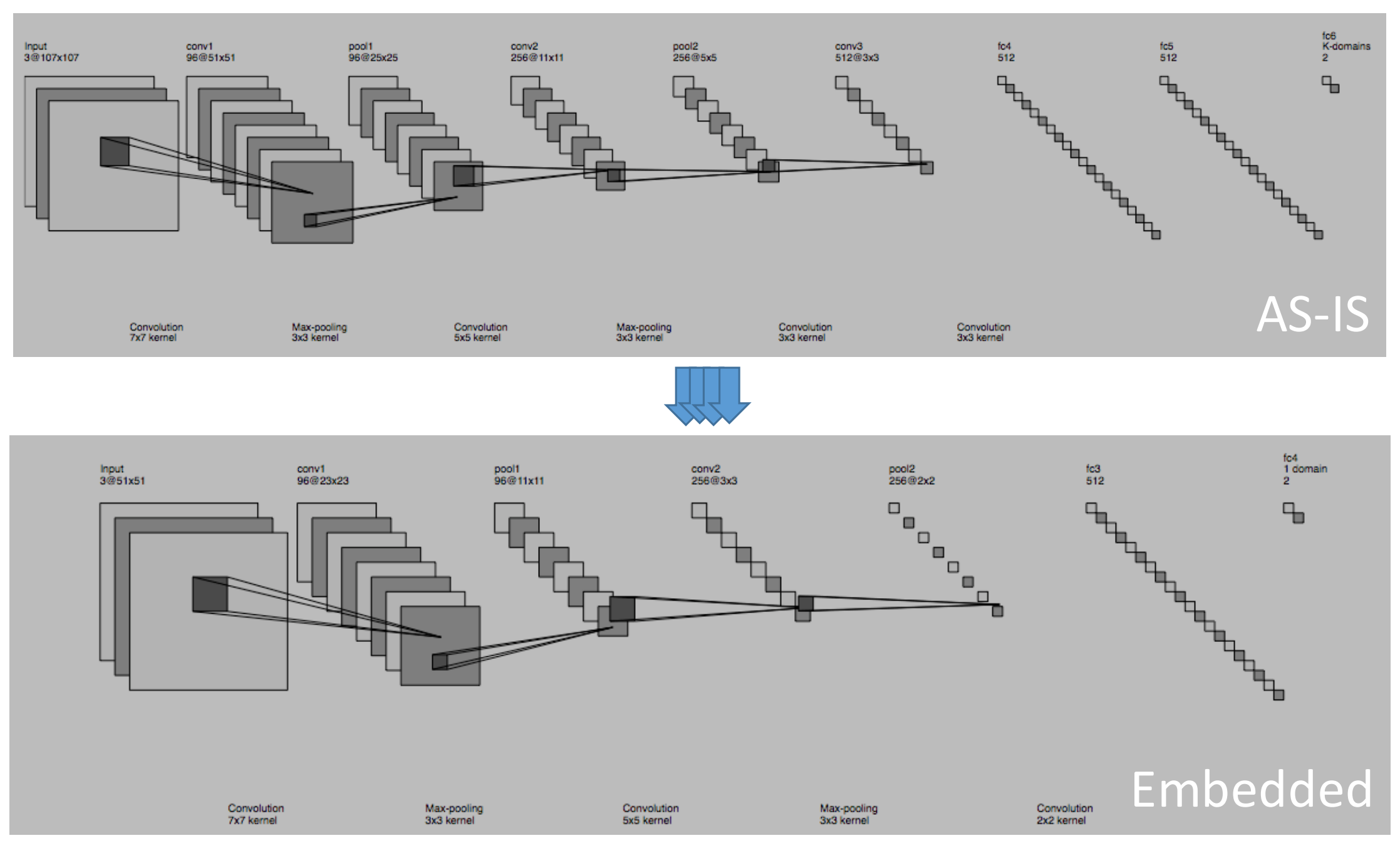
Optimizing Methods

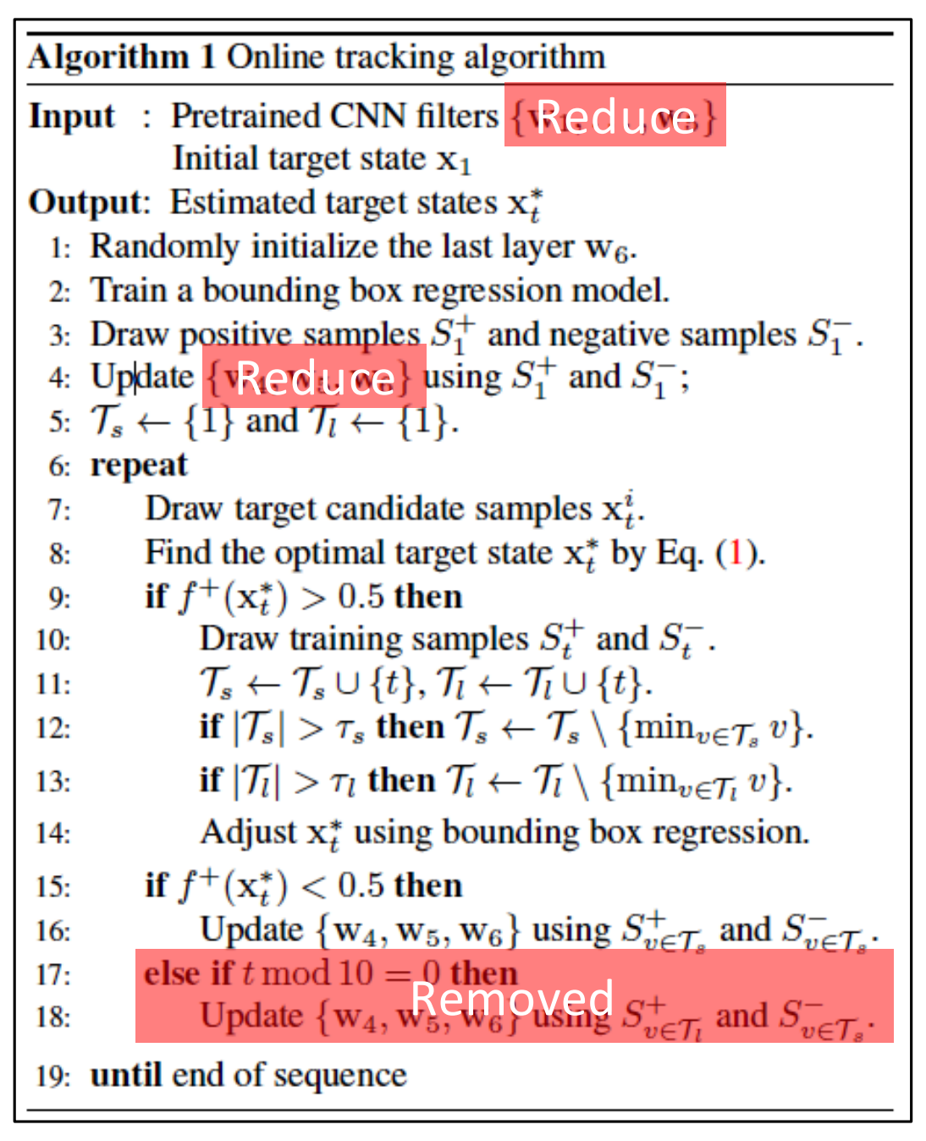
- Reduce the size of input of the first layer: 107 X 107 -> 51 X 51
- Reduce the number of layers: 6 -> 4 (conv1, conv2, fc3, fc4)
- Remove the online tracking process which randomly generates positive and negative candidates to learn per each process

Implemented Architectures



Embedded *MDNet* Real-time Application

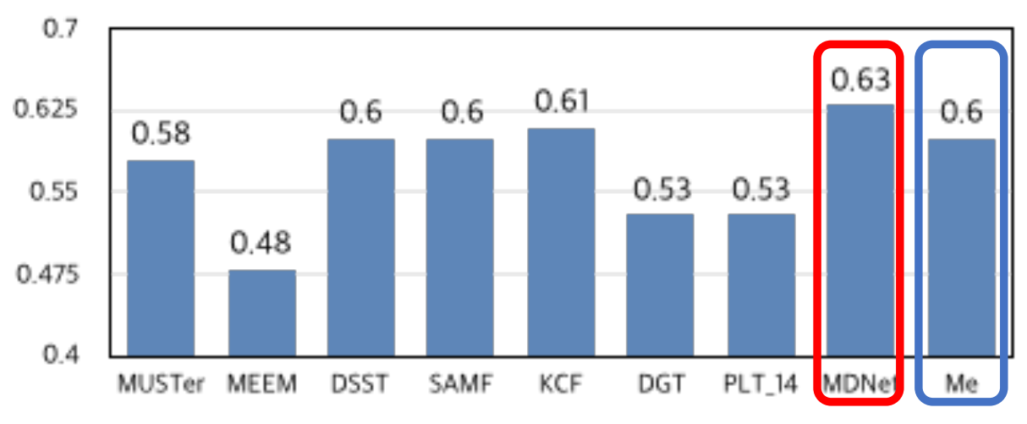


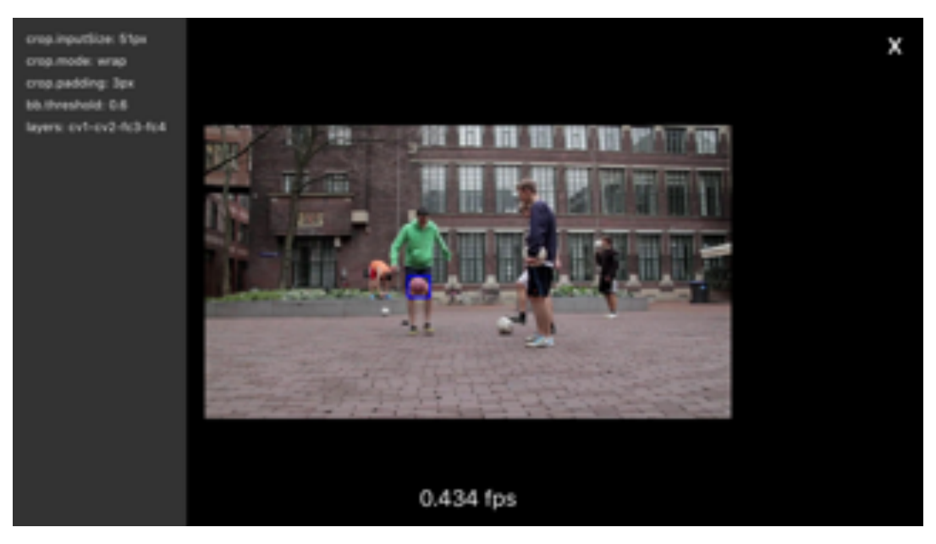


```
Algorithm 1 Online tracking algorithm
Input  : Pretrained CNN filters {Reduce}
         Initial target state $x_1$
Output: Estimated target states $x_t^*$
1: Randomly initialize the last layer $w_6$.
2: Train a bounding box regression model.
3: Draw positive samples $S_1^+$ and negative samples $S_1^-$.
4: Update {Reduce} using $S_1^+$ and $S_1^-$;
5: $\mathcal{T}_s \leftarrow \{1\}$ and $\mathcal{T}_l \leftarrow \{1\}$.
6: repeat
7:     Draw target candidate samples $x_t^i$.
8:     Find the optimal target state $x_t^*$ by Eq. (1).
9:     if $f^+(x_t^*) > 0.5$ then
10:        Draw training samples $S_t^+$ and $S_t^-$.
11:        $\mathcal{T}_s \leftarrow \mathcal{T}_s \cup \{t\}, \mathcal{T}_l \leftarrow \mathcal{T}_l \cup \{t\}$.
12:        if $|\mathcal{T}_s| > \tau_s$ then $\mathcal{T}_s \leftarrow \mathcal{T}_s \setminus \{\min_{v\in\mathcal{T}_s} v\}$.
13:        if $|\mathcal{T}_l| > \tau_l$ then $\mathcal{T}_l \leftarrow \mathcal{T}_l \setminus \{\min_{v\in\mathcal{T}_l} v\}$.
14:        Adjust $x_t^*$ using bounding box regression.
15:    if $f^+(x_t^*) < 0.5$ then
16:        Update $\{w_4, w_5, w_6\}$ using $S_{v\in\mathcal{T}_s}^+$ and $S_{v\in\mathcal{T}_s}^-$.
17:    else if $t$ mod $10 = 0$ then                                   Removed
18:        Update $\{w_4, w_5, w_6\}$ using $S_{v\in\mathcal{T}_l}^+$ and $S_{v\in\mathcal{T}_s}^-$.
19: until end of sequence
```

Results

- Dataset: VOT2015.*ball1*
  - Baseline result for Accuracy: 0.60 (Not for all sequences)
  - Couldn't check Region_noise
  - 0.40 fps on average





crop.inputSize: 51px
crop.mode: wrap
crop.padding: 3px
bb.threshold: 0.6
layers: cv1-cv2-fc3-fc4

- Real-time Visual Tracking Application

Discussion and Future Research

- Fully-implemented CNN for iOS is needed
  - *DeepLearningKit* is currently not supporting some parameters and has some bugs
  - May be better to use *torch7-ios*
- CNN which fully supports Metal will be more useful
  - Metal is powerful GPU accelerating library, which is currently not fully supported by any CNN for iOS
  - With GPU acceleration, the online tracking process would be enabled