

# Object Detection in Videos with Tubelet Proposal Networks

Kai Kang<sup>1,2</sup> Hongsheng Li<sup>2,\*</sup> Tong Xiao<sup>1,2</sup> Wanli Ouyang<sup>2,5</sup> Junjie Yan<sup>3</sup> Xihui Liu<sup>4</sup> Xiaogang Wang<sup>2,\*</sup>

<sup>1</sup>Shenzhen Key Lab of Comp. Vis. & Pat. Rec., Shenzhen Institutes of Advanced Technology, CAS, China

<sup>2</sup>The Chinese University of Hong Kong <sup>3</sup>SenseTime Group Limited <sup>4</sup>Tsinghua University <sup>5</sup>The University of Sydney

{kkang, hsli, xiaotong, wlouyang, xgwang}@ee.cuhk.edu.hk

yanjunjie@sensetime.com xh-liu13@mails.tsinghua.edu.cn

## Abstract

*Object detection in videos has drawn increasing attention recently with the introduction of the large-scale ImageNet VID dataset. Different from object detection in static images, temporal information in videos is vital for object detection. To fully utilize temporal information, state-of-the-art methods [15, 14] are based on spatiotemporal tubelets, which are essentially sequences of associated bounding boxes across time. However, the existing methods have major limitations in generating tubelets in terms of quality and efficiency. Motion-based [14] methods are able to obtain dense tubelets efficiently, but the lengths are generally only several frames, which is not optimal for incorporating long-term temporal information. Appearance-based [15] methods, usually involving generic object tracking, could generate long tubelets, but are usually computationally expensive. In this work, we propose a framework for object detection in videos, which consists of a novel tubelet proposal network to efficiently generate spatiotemporal proposals, and a Long Short-term Memory (LSTM) network that incorporates temporal information from tubelet proposals for achieving high object detection accuracy in videos. Experiments on the large-scale ImageNet VID dataset demonstrate the effectiveness of the proposed framework for object detection in videos.*

## 1. Introduction

The performance of object detection has been significantly improved recently with the emergence of deep neural networks. Novel neural network structures, such as GoogLeNet [29], VGG [27] and ResNet [8], were proposed to improve the learning capability on large-scale computer vision datasets for various computer vision tasks, such as object detection [5, 24, 23, 21], semantic segmentation [20, 2, 16], tracking [31, 1, 33], scene understanding [25, 26, 19], person search [18, 32], etc. State-of-the-art

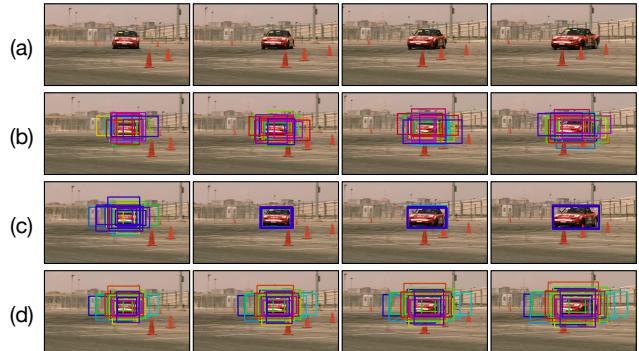


Figure 1. Proposals methods for video object detection. (a) original frames. (b) static proposals have no temporal association, which is hard to incorporate temporal information for proposal classification. (c) bounding box regression methods would focus on the dominant object, lose proposal diversity and may also cause recall drop since all proposals tend to aggregate on the dominant objects. (d) the ideal proposals should have temporal association and have the same motion patterns with the objects while keeping their diversity.

object detection frameworks for static images are based on these networks and consist of three main stages [6]. Bounding box proposals are first generated from the input image based on how likely each location contains an object of interest. The appearance features are then extracted from each box proposal to classify them as one of the object classes. Such bounding boxes and their associated class scores are refined by post-processing techniques (e.g., Non-Maximal Suppression) to obtain the final detection results. Multiple frameworks, such as Fast R-CNN [5] and Faster R-CNN [24], followed this research direction and eventually formulated the object detection problem as training end-to-end deep neural networks.

Although great success has been achieved in detecting objects on static images, video object detection remains a challenging problem. Several factors contribute to the difficulty of this problem, which include the drastic appearance and scale changes of the same object over time, object-to-object occlusions, motion blur, and the mismatch

\*Corresponding authors

between the static-image data and video data. The new task of detecting objects in videos (VID) introduced by the ImageNet challenge in 2015 provides a large-scale video dataset, which requires labeling every object of 30 classes in each frame of the videos. Driven by this new dataset, multiple systems [7, 14, 15] were proposed to extend static-image object detectors for videos.

Similar to the bounding box proposals in the static object detection, the counterpart in videos are called tubelets, which are essentially sequences of bounding boxes proposals. State-of-the-art algorithms for video object detection utilize the tubelets to some extend to incorporate temporal information for obtaining detection results. However, the tubelet generation is usually based on the frame-by-frame detection results, which is extremely time consuming. For instance, the tracking algorithm used by [14, 15] needs 0.5 second to process each detection box in each frame, which prevents the systems to generate enough tubelet proposals for classification in an allowable amount of time, since the video usually contains hundreds of frames with hundreds of detection boxes on each frame. Motion-based methods, such as optical-flow-guided propagation [14], can generate dense tubelets efficiently, but the lengths are usually limited to only several frames (*e.g.*, 7 frames in [14]) because of their inconsistent performance for long-term tracking. The ideal tubelets for video object detection should be long enough to incorporate temporal information while diverse enough to ensure high recall rates (Figure 1).

To mitigate the problems, we propose a framework for object detection in videos. It consists of a Tubelet Proposal Network (TPN) that simultaneously obtains hundreds of diverse tubelets starting from static proposals, and a Long Short-Term Memory (LSTM) sub-network for estimating object confidences based on temporal information from the tubelets. Our TPN can efficiently generate tubelet proposals via feature map pooling. Given a static box proposal at a starting frame, we pool features from the same box locations across multiple frames to train an efficient multi-frame regression neural network as the TPN. It is able to learn complex motion patterns of the foreground objects to generate robust tubelet proposals. Hundreds of proposals in a video can be tracked simultaneously. Such tubelet proposals are shown to be of better quality than the ones obtained on each frame independently, which demonstrates the importance of temporal information in videos. The visual features extracted from the tubelet boxes are automatically aligned into feature sequences and are suitable for learning temporal features with the following LSTM network, which is able to capture long-term temporal dependency for accurate proposal classification.

The contribution of this paper is that we propose a new deep learning framework that combines tubelet proposal generation and temporal classification with visual-temporal features. An efficient tubelet proposal generation algorithm is developed to generate tubelet proposals that capture spatiotemporal locations of objects in videos. A tempo-

ral LSTM model is adopted for classifying tubelet proposals with both visual features and temporal features. Such high-level temporal features are generally ignored by existing detection systems but are crucial for object detection in videos.

## 2. Related work

**Object detection in static images.** State-of-the-art object detection systems are all based on deep CNNs. Girshick *et al.* [6] proposed the R-CNN to decompose the object detection problem into multiple stages including region proposal generation, CNN finetuning, and region classification. To accelerate the training process of R-CNN, Fast R-CNN [5] was proposed to avoid time-consumingly feeding each image patch from bounding box proposals into CNN to obtain feature representations. Features of multiple bounding boxes within the same image are warped from the same feature map efficiently via ROI pooling operations. To accelerate the generation of candidate bounding box proposals, Faster R-CNN integrates a Region Proposal Network into the Fast R-CNN framework, and is able to generate box proposals directly with neural networks.

**Object detection in videos.** Since the introduction of the VID task by the ImageNet challenge, there have been multiple object detection systems for detecting objects in videos. These methods focused on post-processing class scores by static-image detectors to enforce temporal consistency of the scores. Han *et al.* [7] associated initial detection results into sequences. Weaker class scores along the sequences within the same video were boosted to improve the initial frame-by-frame detection results. Kang *et al.* [15] generated new tubelet proposals by applying tracking algorithms to static-image bounding box proposals. The class scores along the tubelet were first evaluated by the static-image object detector and then re-scored by a 1D CNN model. The same group [14] also tried a different strategy for tubelet classification and re-scoring. In addition, initial detection boxes were propagated to nearby frames according to optical flows between frames, and the class scores not belonging to the top classes were suppressed to enforce temporal consistency of class scores.

**Object localization in videos.** There have been works and datasets [3, 13, 22] on object localization in videos. However, they have a simplified problem setting, where each video is assumed to contain only one known or unknown class and requires annotating only one of the objects in each frame.

## 3. Tubelet proposal networks

Existing methods on object detection in videos generates tubelet proposals utilizing either generic single-object tracker starting at a few key frames [15] or data association methods (*i.e.* tracking-by-detection methods) on per-frame object detection results [7]. These methods either are too computationally expensive to generate enough dense

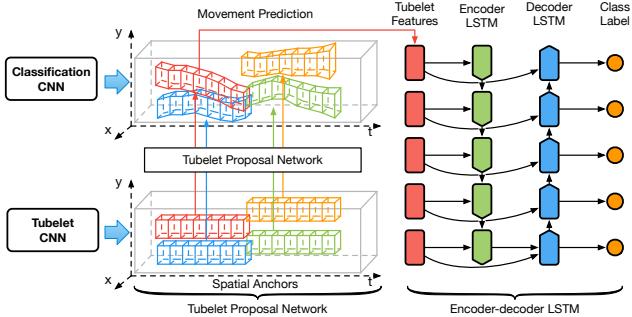


Figure 2. The proposed object detection system, which consists of two main parts. The first is a tubelet proposal network to efficiently generating tubelet proposals. The tubelet proposal network extracts multi-frame features within the spatial anchors, predicts the object motion patterns relative to the spatial anchors and generates tubelet proposals. The gray box indicates the video clip and different colors indicate proposal process of different spatial anchors. The second part is an encoder-decoder CNN-LSTM network to extract tubelet features and classify each proposal boxes into different classes. The tubelet features are first fed into the encoder LSTM by a forward pass to capture the appearance features of the entire sequence. Then the states are copied to the decoder LSTM for a backward pass with the tubelet features. The encoder-decoder LSTM processes the entire clip before outputting class probabilities for each frame.

tublets, or are likely to drift and result in tracking failures. Even for an 100-fps single-object tracker, it might take about 56 GPU days to generate tubelets with 300 bounding boxes per frame for the large-scale ImageNet VID dataset.

We propose a Tubelet Proposal Network (TPN) which is able to generate tubelet proposals efficiently for videos. As shown in Figure 2, the Tubelet Proposal Network consists of two main components, the first sub-network extracts visual features across time based on static region proposals at a single frame. Our key observation is that, since the receptive fields (RF) of CNNs are generally large enough, we can perform feature map pooling simply at the same bounding box locations across time to extract the visual features of moving objects. Based on the pooled visual features, the second component is a regression layer for estimating bounding boxes' temporal displacements to generate tubelet proposals.

### 3.1. Preliminaries on ROI-pooling for regression

There are existing works that utilize feature map pooling for object detection. The Fast R-CNN framework [5] utilizes ROI-pooling on visual feature maps for object classification and bounding box regression. The input image is fed into a CNN and forward propagated to generate visual feature maps. Given different object proposals, their visual features are directly ROI-pooled from the feature maps according to the box coordinates. In this way, CNN only needs to forward propagate once for each input image and saves much computational time. Let  $b_t^i = (x_t^i, y_t^i, w_t^i, h_t^i)$  denote

the  $i$ th static box proposal at time  $t$ , where  $x, y, w$  and  $h$  represent the two coordinates of the box center, width and height of the box proposal. The ROI-pooling obtains visual features  $\mathbf{r}_t^i \in \mathbb{R}^f$  at box  $b_t^i$ .

The ROI-pooled features  $\mathbf{r}_t^i$  for each object bounding box proposal can be used for object classification, and, more interestingly, for bounding box regression, which indicates that the visual features obtained by feature map pooling contain necessary information describing objects' locations. Inspired by this technique, we propose to extract multi-frame visual features via ROI-pooling, and use such features for generating tubelet proposals via regression.

### 3.2. Static object proposals as spatial anchors

Static object proposals are class-free bounding boxes indicating the possible locations of objects, which could be efficiently obtained by different proposal methods such as SelectiveSearch [30], Edge Boxes [34] and Region Proposal Networks [24]. For object detection in videos, however, we need both spatial and temporal locations of the objects, which are crucial to incorporate temporal information for accurate object proposal classification.

For general objects in videos, movements are usually complex and difficult to predict. The static object proposals usually have high recall rates (*e.g.* >90%) at individual frames, which is important because it is the upper bound of object detection performance. Therefore, it is natural to use static proposals as starting anchors for estimating their movements at following frames to generate tubelet proposals. If their movements can be robustly estimated, high object recall rate at the following times can be maintained.

Let  $b_1^i$  denote a static proposal of interest at time  $t = 1$ . Particularly, to generate a tubelet proposal starting at  $b_1^i$ , visual features within the  $w$ -frame temporal window from frame 1 to  $w$  are pooled at the same location  $b_1^i$  as  $\mathbf{r}_1^i, \mathbf{r}_2^i, \dots, \mathbf{r}_w^i$  in order to generate the tubelet proposal. We call  $b_1^i$  a “spatial anchor”. The pooled regression features encode visual appearances of the objects. Recovering correspondences between the visual features  $(\mathbf{r}_1^i, \mathbf{r}_2^i, \dots, \mathbf{r}_w^i)$  leads to accurate tubelet proposals, which is modeled by a regression layer detailed in the next subsection.

The reason why we are able to pool multi-frame features from the same spatial location for tubelet proposals is that CNN feature maps at higher layers usually have large receptive fields. Even if visual features are pooled from a small bounding box, its visual context is far greater than the bounding box itself. Pooling at the same box locations across time is therefore capable of capturing large possible movements of objects. In Figure 2, we illustrate the “spatial anchors” for tubelet proposal generation. The features in the same locations are aligned to predict the movement of the object.

We use a GoogLeNet with Batch Normalization (BN) model [12] for the TPN. In our settings, the ROI-pooling layer is connected to “inception\_4d” of the BN model, which has a receptive field of 363 pixels. Therefore, the

network is able to handle up to 363-pixel movement when ROI-pooling the same box locations across time, which is more than enough to capture short-term object movements. Each static proposal is regarded as an anchor point for feature extraction within a temporal window  $w$ .

### 3.3. Supervisions for tubelet proposal generation

Our goal is to generate tubelet proposals that have high object recall rates at each frame and can accurately track objects. Based on the pooled visual features  $\mathbf{r}_1^i, \mathbf{r}_2^i, \dots, \mathbf{r}_w^i$  at box locations  $b_t^i$ , we train a regression network  $R(\cdot)$  that effectively estimates the relative movements w.r.t. the spatial anchors,

$$m_1^i, m_2^i, \dots, m_w^i = R(\mathbf{r}_1^i, \mathbf{r}_2^i, \dots, \mathbf{r}_w^i), \quad (1)$$

where the relative movements  $m_t^i = (\Delta x_t^i, \Delta y_t^i, \Delta w_t^i, \Delta h_t^i)$  are calculated as

$$\begin{aligned} \Delta x_t^i &= (x_t^i - x_1^i)/w_1^i, \quad \Delta y_t^i = (y_t^i - y_1^i)/h_1^i, \\ \Delta w_t^i &= \log(w_t^i/w_1^i), \quad \Delta h_t^i = \log(h_t^i/h_1^i). \end{aligned} \quad (2)$$

Once we obtain such relative movements, the actual box locations of the tubelet could be easily inferred. We adopt a fully-connected layer that takes the concatenated visual features  $[\mathbf{r}_1^i, \mathbf{r}_2^i, \dots, \mathbf{r}_w^i]^T$  as the input, and outputs  $4w$  movement values of a tubelet proposal by

$$[m_1^i, \dots, m_w^i]^T = W_w [\mathbf{r}_1^i, \dots, \mathbf{r}_w^i]^T + b_w, \quad (3)$$

where  $W_w \in \mathbb{R}^{fw \times 4w}$  and  $b_w \in \mathbb{R}^{4w}$  are the learnable parameters of the layer.

The remaining problem is how to design proper supervisions for learning the relative movements. Our key assumption is that the tubelet proposals should have consistent movement patterns with the ground-truth objects. However, given static object proposals as the starting boxes for tubelet generation, they usually do not have a perfect 100% Intersection-over-Union (IoU) ratio with the ground truth object boxes. Therefore, we require static box proposals that are close to ground truth boxes to follow the movement patterns of the ground truth boxes. More specifically, if a static object proposal  $b_t^i$  has a greater-than-0.5 IoU value with a ground truth box  $\hat{b}_t^i$ , and the IoU value is greater than those of other ground truth boxes, our regression layer tries to generate tubelet boxes following the same movement patterns  $\hat{m}_t^i$  of the ground truth  $\hat{b}_t^i$  as much as possible. The relative movement targets  $\hat{m}_t^i = (\hat{x}_t^i, \hat{y}_t^i, \hat{w}_t^i, \hat{h}_t^i)$  can be defined w.r.t. the ground truth boxes at time 1,  $\hat{b}_1^i$ , in the similar way as Eq. (2). It is trivial to see that  $\hat{m}_1^i = (0, 0, 0, 0)$ . Therefore, we only need to predict  $\hat{m}_2^i$  to  $\hat{m}_w^i$ . Note that by learning relative movements w.r.t to the spatial anchors at the first frame, we can avoid cumulative errors in conventional tracking algorithms to some extend.

The movement targets are normalized by their mean  $\bar{m}_t$  and standard deviation  $\sigma_t$  as the regression objectives,

$$\tilde{m}_t^i = (\hat{m}_t^i - \bar{m}_t)/\sigma_t, \quad \text{for } t = 1, \dots, w. \quad (4)$$

To generate  $N$  tubelets that follow movement patterns of their associated ground truth boxes, we minimize the following object function w.r.t. all  $x_t^i, y_t^i, w_t^i, h_t^i$ ,

$$L(\{\tilde{M}\}, \{M\}) = \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^w \sum_{k \in \{x, y, w, h\}} d(\Delta k_t^i), \quad (5)$$

where  $\{\tilde{M}\}$  and  $\{M\}$  are the sets of all normalized movement targets and network outputs, and

$$d(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1, \\ |x| - 0.5 & \text{otherwise.} \end{cases} \quad (6)$$

is the smoothed  $L_1$  loss for robust box regression in [5].

The network outputs  $\hat{m}_t^i$  are mapped back to the real relative movements  $m_t^i$  by

$$m_t^i = (\hat{m}_t^i + \bar{m}_t) * \sigma_t. \quad (7)$$

By our definition, if a static object proposal covers some area the object, it should cover the same portion of object in the later frames (see Figure 1 (d) for examples).

### 3.4. Initialization for multi-frame regression layer

The size of the temporal window is also a key factor in the TPN. The simplest model is a 2-frame model. For a given frame, the features within the spatial anchors on current frame and the next frames are extracted and concatenated,  $[\mathbf{r}_1^i, \mathbf{r}_2^i]^T$ , to estimate the movements of  $b_1^i$  on the next frames. However, since the 2-frame model only utilizes minimal temporal information within a very short temporal window, the generated tubelets may be non-smooth and easy to drift. Increasing the temporal window utilizes more temporal information so as to estimate more complex movement patterns.

Given the temporal window size  $w$ , the dimension of the extracted features are  $fw$ , where  $f$  is the dimension of visual features in a single frame within the spatial anchors (e.g., 1024-dimensional “inception\_5b” features from the BN model in our settings). Therefore, the parameter size of the regress layer is of  $\mathbb{R}^{fw \times 4w}$  and grows quadratically with the temporal window size  $w$ .

If the temporal window size is large, randomly initializing such a large matrix has difficulty in learning a good regression layer. We propose a “block” initialization method to use the learned features from 2-frame model to initialize the multi-frame models.

In Figure 3, we show how to use a pre-trained 2-frame model’s regression layer to initialize that of a 5-frame model. Since the target  $\hat{m}_1^i$  in Equation (2) is always  $(0, 0, 0, 0)$  we only need to estimate movements for the later frames. The parameter matrix  $W_2$  is of size  $\mathbb{R}^{2f \times 4}$  since the input features are concatenations of two frames and the bias term  $b_2$  is of size  $\mathbb{R}^4$ . For the 5-frame regression layer, the parameter matrix  $W_5$  is of size  $\mathbb{R}^{5f \times (4 \times 4)}$  and the bias term

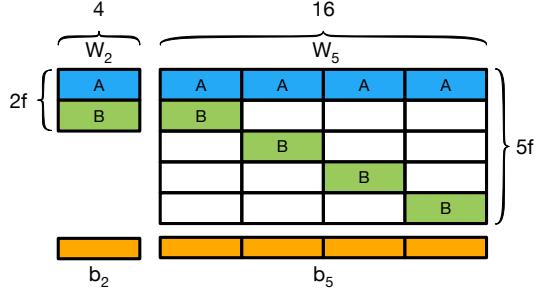


Figure 3. Illustration of the “block” initialization method. The 2-frame model’s regression layer has weights  $W_2$  and bias  $b_2$ , the  $W_2$  consists of two sub-matrices  $A$  and  $B$  corresponding to the features of the first and second frames. Then a 5-frame model’s regression layer can be initialized with the sub-matrices as shown in the figure. The bias term  $b_5$  is a simple repetition of  $b_2$ .

$b_5$  is of  $\mathbb{R}^{(4 \times 4)}$ . Essentially, we utilize visual features from frame 1 & 2 to estimate movements in frame 2, frame 1 & 3 for frame 3, and so on. The matrix  $W_2$  is therefore divided into two sub-matrices  $A \in \mathbb{R}^{f \times 4}$  and  $B \in \mathbb{R}^{f \times 4}$  to fill the corresponding entries of matrix  $W_5$ . The bias term  $b_5$  is a repetition of  $b_2$  for 4 times.

In our experiments, we first train a 2-frame model with random initialization and then use the 2-frame model to initialize the multi-frame regression layer.

#### 4. Overall detection framework with tubelet generation and tubelet classification

Based on the Tubelet Proposal Networks, we propose a framework that is efficient for object detection in videos. Compared with state-of-the-art single object tracker, It only takes our TPN 9 GPU days to generate dense tubelet proposals on the ImageNet VID dataset. It is also capable of utilizing useful temporal information from tubelet proposals to increase detection accuracy. As shown in Figure 2, the framework consists of two networks, the first one is the TPN for generating candidate object tubelets, and the second network is a CNN-LSTM classification network that classifies each bounding box on the tubelets into different object categories.

##### 4.1. Efficient tubelet proposal generation

The TPN is able to estimate movements of each static object proposal within a temporal window  $w$ . For object detection in videos in large-scale datasets, we need to not only efficiently generate tubelets for hundreds of spatial anchors in parallel, but also generate tubelets with sufficient lengths to incorporate enough temporal information.

To generate tubelets with length of  $l$ , (see illustration in Figure 4 (a)), we utilize static object proposals on the first frame as spatial anchors, and then iteratively apply TPN with temporal window  $w$  until the tubelets cover all  $l$  frames. The last estimated locations of the previous iteration

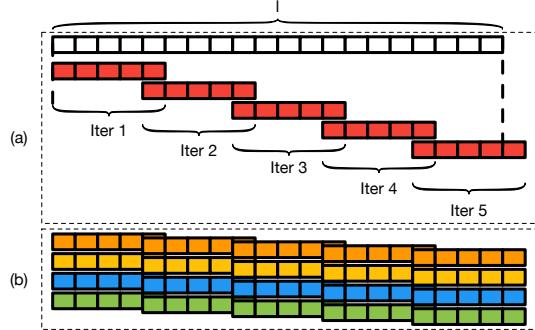


Figure 4. Efficiently generating tubelet proposals. (a) the TPN generates the tubelet proposal of temporal window  $w$  and uses the last-frame output of the proposal as static anchors for the next iteration. This process iterates until the whole track length is covered. (b) multiple static anchors in a frame are fed to the Fast R-CNN network with a single forward pass for simultaneously generating multiple tubelet proposals. Different colors indicate different spatial anchors

are used as spatial anchors for the next iteration. This process can iterate to generate tubelet proposals of arbitrary lengths.

For  $N$  static object proposals in the same starting frame, the bottom CNN only needs to conduct an one-time forward propagation to obtain the visual feature maps, and thus enables efficient generation of hundreds of tubelet proposals (see Figure 4 (b)).

Compared to previous methods that adopt generic single object trackers, our proposed methods is dramatically faster for generating a large number of tubelets. The tracking method used in [15] has reported 0.5 fps running speed for a single object. For a typical frame with 300 spatial anchors, it takes 150s for each frame. Our method has an average speed of 0.488s for each frame, which is about 300× faster. Even compared to the recent 100 fps single object tracker in [9], our method is about 6.14× faster.

##### 4.2. Encoder-decoder LSTM (ED-LSTM) for temporal classification

After generating the length- $l$  tubelet proposal, visual features  $\mathbf{u}_t^1, \dots, \mathbf{u}_t^l, \dots, \mathbf{u}_l^l$  can be pooled from tubelet box locations for object classification with temporal information. Existing methods [15, 7, 14] mainly use temporal information in post processing, either propagating detections to neighboring frames or temporally smoothing detection scores. The temporal consistency of detection results is important, but to capture the complex appearance changes in the tubelets, we need to learn discriminative spatiotemporal features at the tubelet box locations.

As shown in Figure 2, the proposed classification sub-network contains a CNN that processes input images to obtain feature maps. Classification features ROI-pooled from each tubelet proposal across time are then fed into a one-layer Long Short-Term Memory (LSTM) network [11] for

tubelet classification. It is a special type of recurrent neural network (RNN) and is widely investigated for learning spatiotemporal features in recent years. Each LSTM unit has a memory unit that conveys visual information across the time for incorporating temporal information.

The input for each time step  $t$  of the LSTM for the  $i$ th tubelet are the cell state  $c_{t-1}^i$ , hidden state  $h_{t-1}^i$  of the previous frame, and the classification features  $\mathbf{u}_t^i$  pooled at the current time  $t$ . The starting state  $(c_0^i, h_0^i)$  of the LSTM is set to zeros. The output is the hidden states  $h_t^i$ , which is connected to a fully-connected layer for predicting class confidences and another FC layer for box regression. One problem with the vanilla LSTM is that the initial state may dramatically influence the classification of the first several frames. Inspired by sequence-to-sequence LSTM in [28], we propose an encoder-decoder LSTM model for object detection in videos as shown in Figure 2. The input features are first fed into an encoder LSTM to encode the appearance features of the entire tubelet into the memory. The memory and hidden states are then fed into the decoder LSTM, which then classifies the tubelet in the reverse order with the reversed inputs from the last frame back to the first frame. In this way, better classification accuracy can be achieved by utilizing both past and future information. The low prediction confidences caused by the all-zero initial memory states can be avoided.

## 5. Experiments

### 5.1. Datasets and evaluation metrics

The proposed framework is evaluated on the ImageNet object detection from video (VID) dataset introduced in the ILSVRC 2015 challenge. There are 30 object classes in the dataset. The dataset is split into three subsets: the training set that contains 3862 videos, the validation set that contains 555 videos, and the test set that contains 937 videos. Objects of the 30 classes are labeled with ground truth bounding boxes on all the video frames. Since the ground truth labels for the test set are not publicly available, we report all results on the validation set as a common practice on the ImageNet detection tasks. The mean average precision (Mean AP) of 30 classes is used as the evaluation metric.

In addition, we also evaluate our system on the YouTubeObjects (YTO) [22] dataset for the object localization task. The YTO dataset has 10 object classes, which are a subset of the ImageNet VID dataset. The YTO dataset is weakly annotated with only one object of one ground truth class in the video. We only use this dataset for evaluation and the evaluation metric is CorLoc performance measure used in [3], *i.e.*, the recall rate of ground-truth boxes with IoU above 0.5.

### 5.2. Base CNN model training

We choose GoogLeNet with Batch Normalization (BN) [12] as our base CNN models for both our TPN and CNN-

LSTM models without sharing weights between them. The BN model is pre-trained with the ImageNet classification data and fine-tuned on the ImageNet VID dataset. The static object proposals are generated by a RPN network trained on the ImageNet VID dataset. The recall rate of the per-frame RPN proposals on the VID validation set is 95.92 with 300 boxes on each frame.

To integrate with Fast RCNN framework, we placed the ROI-pooling layer after “inception\_4d” rather than the last inception module (“inception\_5b”), because “inception\_5b” has  $32 \times$  down-sampling with a receptive field of 715 pixels, which is too large for ROI-pooling to generate discriminative features. The output size of ROI-pooling is  $14 \times 14$  and we keep the later inception modules and the final global pooling after “inception\_5b”. We then add one more FC layer for different tasks including tubelet proposal, classification or bounding box regression.

The BN model is trained on 4 Titan X GPUs for 200,000 iterations, with 32 RoIs from 2 images on each card in every iteration. The initial learning rate is  $5 \times 10^{-4}$  and decreases to 1/10 of its previous value for every 60,000 iterations. All BN layers are frozen during the fine-tuning. After fine-tuning on DET data, the BN model achieves 50.3% mean AP on the ImageNet DET data. After fine-tuning the BN model on the VID data with the same hyper-parameter setting for 90,000 iterations, it achieves 63.0% mean AP on the VID validation set.

### 5.3. TPN training and evaluation

With the fine-tuned BN model, we first train a 2-frame model on the ImageNet VID dataset. Since the TPN needs to estimate the movement of the object proposals according ground-truth objects’ movements, we only select static proposals that have greater-than-0.5 IoU overlaps with ground-truth annotations as spatial anchors following Section 3.3. For those proposals that do not have greater-than-0.5 overlaps with ground-truth boxes, they are not used for training the TPN. During the test stage, however, all static object proposals in every 20 frames are used as spatial anchors for tubelet proposal generation. All tubelets are 20-frame long. The ones starting from negative static proposals are likely to stay in the background regions, or track the foreground objects when they appear in their nearby regions.

We investigate different temporal window sizes  $w$  and initialization methods described in Section 3.4. Since the ground truth movements  $\hat{m}_t^i$  can be obtained from the ground truth annotations, each positive static proposal has an “ideal” tubelet proposal in comply with its associated ground-truth’s movements. Three metrics are used to evaluate the accuracy of generated tubelets by different models (Table 1). One is the mean absolute pixel difference (MAD) of the predicted coordinates and their ground truth. The second one is the mean relative pixel difference (MRD) with  $x$  differences normalized by widths and  $y$  differences normalized by heights. The third metric is the mean intersection-over-union (IOU) between predicted

Method	Window	MAD	MRD	Mean IOU
MoveTubelets Random	2	15.50	0.0730	0.7966
MoveTubelets Random	5	26.00	0.1319	0.6972
MoveTubelets RNN	5	13.87	0.0683	0.8060
MoveTubelets Block	5	<b>12.98</b>	<b>0.0616</b>	<b>0.8244</b>
MoveTubelets Block	11	15.20	0.0761	0.8017
MoveTubelets Block	20	18.03	0.0874	0.7731

Table 1. Evaluation of tubelet proposals obtained with varying window sizes and different initialization methods. As the parameter size grows quadratically with the temporal window. The 5-frame model with random initialization has much worse accuracy compared to the proposed transformation initialization. As the temporal window grows, the motion pattern becomes more complex and the movement displacement may also exceed the receptive field, which also causes accuracy decreases.

boxes and target boxes. From the table, we can see that the 2-frame baseline model has a MAD of 15.50, MRD of 0.0730 and Mean IOU of 0.7966. For the 5-frame model, if we initialize the fully-connected regression layer randomly without using the initialization technique (other layers are still initialized by the finetuned BN model), the performance drops significantly compared to that of the 2-frame model. The reason might be that the parameter size of the 5-frame model increases by 10 times (as shown in Figure 3), which makes it more difficult to train without a good initial point. However, with the proposed technique, the multi-frame regression layer with the 2-frame model, the generated tubelets have better accuracy than the 2-frame model because of the larger temporal context.

If the temporal window continues to increase, even with the proposed initialization techniques, the performance decreases. This might be because if the temporal window is too large, the movement of the objects might be too complex for the TPN to recover the visual correspondences between far-away frames. In the later experiments, we use the 5-frame TPN to generate 20-frame-long tubelet proposals.

In comparison with our proposed method, an RNN baseline with is implemented by replacing the tubelet regression layer with an RNN layer of 1024 hidden neurons and a regression layer to predict 4 motion targets. As shown in Table 1, the RNN baseline performs worse than our method.

#### 5.4. LSTM Training

After generating the tubelet proposals, the proposed CNN-LSTM models extract classification features  $u_t^i$  at tubelet box locations with the finetuned BN model. The dimension of the features at each time step is 1024.

The LSTM has 1024 cell units and 1024 hidden outputs. For each iteration, 128 tubelets from 4 videos are randomly chosen to form a mini-batch. The CNN-LSTM is trained using stochastic gradient descent (SGD) optimization with momentum of 0.9 for 20000 iterations. The parameters are initialized with standard deviation of 0.0002 and the initial learning rate is 0.1. For every 2,000 iteration, the learning rate decreases by a factor of 0.5.

#### 5.5. Results

**Baseline methods.** The most basic baseline method is Fast R-CNN static detector [5] (denoted as “Static”), which needs static proposals on every frame and does not involve any temporal information. This baseline uses static proposals from the same RPN we use and the Fast R-CNN model is the same as our base BN model. To validate the effectiveness of the tubelet regression targets, we change them into the precise locations of the ground truth on each frame and also generate tubelet proposals (see Figure 1 (c)). Then we apply a vanilla LSTM on these tubelet proposals and denote the results as “LocTubelets+LSTM”. Our tubelet proposal method is denoted as “MoveTubelets”. We also compare with a state-of-the-art single-object tracking method [10] denoted as “KCF”. As for the CNN-LSTM classification part, the baseline methods are the vanilla LSTM (denoted as “LSTM”), and our proposed encoder-decoder LSTM is denoted as “ED-LSTM”.

**Results on ImageNet VID dataset.** The quantitative results on the ImageNet VID dataset are shown in Table 2 and 3. As a convention of detection tasks on the ImageNet dataset, we report the results on the validation set. The performance of the baseline Fast R-CNN detector finetuned on the ImageNet VID dataset has a Mean AP of 0.630 (denoted as “Static”). Compare to the best single model performance in [14], which has a Mean AP of 0.615 using only the VID data, the baseline detector has an 1.5% performance gain.

Directly applying the baseline static detector on the TPN tubelets with temporal window of 5 results in a Mean AP of 0.623 (denoted as “MoveTubelets+Fast RCNN”). In comparison, a state-of-the-art tracker [10] with the baseline static detector (“KCF+Fast RCNN”) has a Mean AP of only 0.567. In addition, although the KCF tracker runs at 50 fps for single object tracking, it takes 6 seconds to process one frame with 300 proposals. Our method is  $12 \times$  faster.

Applying the vanilla LSTM on the tubelet proposals increases the Mean AP to 0.678 (denoted as “MoveTubelets+LSTM”), which has 5.5% performance gain over the tubelet results and 4.8% increase over the static baseline results. This shows that the LSTM is able to learn appearance and temporal features from the tubelet proposals to improve the classification accuracy. Especially for class of “whale”, the AP has over 25% improvement since whales constantly emerge from the water and submerge. A detector has to observe the whole process to classify them correctly.

Compared to bounding box regression tubelet proposal baseline, our tubelet proposal model has 2.5% improvement which shows that our tubelet proposals have more diversity to incorporate temporal information. Changing to the encoder-decoder LSTM model has a Mean AP of 0.684 (denoted as “MoveTubelets+ED-LSTM”) with a 0.6% performance gain over the vanilla LSTM model with performance increases on over half of the classes. One thing to notice is that our encoder-decoder LSTM model performs better than or equal to the tubelet baseline results on all

Method	airplane	antelope	bear	bike	bird	bus	car	cattle	dog	d_car	elephant	fox	g_panda	hamster	horse	lion
Static (Fast RCNN)	0.821	0.784	0.665	0.656	0.661	0.772	0.523	0.491	0.571	0.720	0.681	0.768	0.718	0.897	0.651	0.201
MoveTubelets+Fast RCNN	0.776	0.778	0.663	0.654	0.649	0.766	0.514	0.493	0.559	0.724	0.684	0.775	0.710	0.900	0.642	0.208
LocTubelets+LSTM	0.759	0.783	0.660	0.646	0.682	<b>0.813</b>	0.538	0.528	0.605	0.722	0.698	0.782	0.724	0.901	0.664	0.212
MoveTubelets+LSTM	0.839	<b>0.794</b>	0.715	0.652	<b>0.683</b>	0.794	0.533	<b>0.615</b>	0.608	0.765	0.705	<b>0.839</b>	0.769	<b>0.916</b>	0.661	0.158
MoveTubelets+ED-LSTM	<b>0.846</b>	0.781	<b>0.720</b>	<b>0.672</b>	0.680	0.801	<b>0.547</b>	0.612	<b>0.616</b>	<b>0.789</b>	<b>0.716</b>	0.832	<b>0.781</b>	0.915	<b>0.668</b>	<b>0.216</b>

Method	lizard	monkey	motor	rabbit	r_panda	sheep	snake	squirrel	tiger	train	turtle	Watercraft	whale	zebra	mean AP
Static (Fast RCNN)	0.638	0.347	0.741	0.457	0.558	0.541	0.572	0.298	0.815	0.720	0.744	0.557	0.432	<b>0.894</b>	0.630
MoveTubelets+Fast RCNN	0.646	0.320	0.691	0.454	0.582	0.540	0.567	0.286	0.806	0.730	0.737	0.543	0.414	0.885	0.623
LocTubelets+LSTM	0.743	0.334	0.727	0.513	0.555	0.613	<b>0.688</b>	0.422	0.813	0.781	0.760	0.609	0.429	0.874	0.653
MoveTubelets+LSTM	<b>0.746</b>	0.347	<b>0.771</b>	<b>0.525</b>	<b>0.710</b>	0.609	0.637	0.406	0.845	<b>0.786</b>	<b>0.774</b>	0.602	0.637	0.890	0.678
MoveTubelets+ED-LSTM	0.744	<b>0.366</b>	0.763	0.514	0.706	<b>0.642</b>	0.612	<b>0.423</b>	<b>0.848</b>	0.781	0.772	<b>0.615</b>	<b>0.669</b>	0.885	<b>0.684</b>

Table 2. AP list on ImageNet VID validation set by the proposed method and compared methods.

Static (Fast RCNN)	0.630
TCNN [14]	0.615
Seq-NMS [7]	0.522
Closed-loop [4]	0.500
KCF Tracker [10] + Fast R-CNN	0.567
MoveTubelets + Fast R-CNN	0.623
MoveTubelets+LSTM	0.678
MoveTubelets+ED-LSTM (proposed)	<b>0.684</b>

Table 3. Mean AP for baseline models and proposed methods.



Figure 5. Qualitative results on the ImageNet VID dataset. The bounding boxes are tight and stably concentrate on the objects since the RoIs for each frame are based on the predicted locations on the previous frame. The last 3 rows show the robustness to handle scenes with multiple objects.)

the classes, which means that learning the temporal features consistently improves the detection results.

The qualitative results on the ImageNet VID dataset are shown in Figure 5. The bounding boxes are tight to the objects and we able to track and detect multiple objects during long periods of time.

**Localization on the YouTubeObjects dataset.** In addition to the object detection in video task on the ImageNet VID dataset. We also evaluate our system on video object localization task with the YouTubeObjects (YTO) dataset.

For each test video, we generate tubelet proposals and apply the encoder-decoder LSTM model to classify the tubelet proposals. For each test class, we select the tubelet box with the maximum detection score on the test frames,

if the box has over 0.5 IOU overlap with one of the ground truth boxes, this frame is accurately localized. The system is trained on the ImageNet VID dataset and is directly applied for testing without any finetuning on the YTO dataset. We compare with several state-of-the-art results on the YTO

Method	aero	bird	boat	car	cat	cow	dog	horse	mbike	train	Avg.
Prest <i>et al.</i> [22]	51.7	17.5	34.4	34.7	22.3	17.9	13.5	26.7	41.2	25.0	28.5
Joulin <i>et al.</i> [13]	25.1	31.2	27.8	38.5	41.2	28.4	33.9	35.6	23.1	25.0	31.0
Kvak <i>et al.</i> [17]	56.5	66.4	58.0	76.8	39.9	69.3	50.4	56.3	53.0	31.0	55.7
Kang <i>et al.</i> [15]	<b>94.1</b>	69.7	88.2	79.3	76.6	18.6	89.6	<b>89.0</b>	87.3	75.3	76.8
MoveTubelets+ED-LSTM	91.2	<b>99.4</b>	93.1	<b>94.8</b>	94.3	<b>99.3</b>	<b>90.2</b>	87.8	<b>89.7</b>	<b>84.2</b>	<b>92.4</b>

Table 4. Localization results on the YouTubeObjects dataset. Our model outperforms previous method with large margin.

dataset, and our system outperforms them with a large margin. Compared to the second best results in [15], our system has 15.6% improvement.

## 6. Conclusion

In this work, we propose a system for object detection in videos. The system consists of a novel tubelet proposal network that efficiently generates tubelet proposals and an encoder-decoder CNN-LSTM model to learn temporal features from the tubelets. Our system is evaluated on the ImageNet VID dataset for object detection in videos and the YTO dataset for object localization. Experiments demonstrate the effectiveness of our proposed framework.

**Acknowledgments.** This work is supported in part by SenseTime Group Limited, in part by the General Research Fund through the Research Grants Council of Hong Kong under Grants CUHK14207814, CUHK14206114, CUHK14205615, CUHK14213616, CUHK14203015, CUHK14239816, and CUHK419412, in part by the Hong Kong Innovation and Technology Support Programme Grant ITS/121/15FX, in part by National Natural Science Foundation of China under Grant 61371192, in part by the Ph.D. Program Foundation of China under Grant 20130185120039, and in part by the China Postdoctoral Science Foundation under Grant 2014M552339.

## References

- [1] S.-H. Bae and K.-J. Yoon. Robust online multi-object tracking based on tracklet confidence and online discriminative appearance learning. *CVPR*, 2014. 1
- [2] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. In *ICLR*, 2015. 1
- [3] T. Deselaers, B. Alexe, and V. Ferrari. Localizing Objects While Learning Their Appearance. *ECCV*, 2010. 2, 6
- [4] L. Galteri, L. Seidenari, M. Bertini, and A. Del Bimbo. Spatio-temporal closed-loop object detection. *TIP*, 2017. 8
- [5] R. Girshick. Fast r-cnn. *ICCV*, 2015. 1, 2, 3, 4, 7
- [6] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CVPR*, 2014. 1, 2
- [7] W. Han, P. Khorrami, T. L. Paine, P. Ramachandran, M. Babaeizadeh, H. Shi, J. Li, S. Yan, and T. S. Huang. Seq-nms for video object detection. *arXiv preprint arXiv:1602.08465*, 2016. 2, 5, 8
- [8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 1
- [9] D. Held, S. Thrun, and S. Savarese. Learning to Track at 100 FPS with Deep Regression Networks. In *ECCV*, 2016. 5
- [10] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *TPAMI*, 2015. 7, 8
- [11] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 5
- [12] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 3, 6
- [13] A. Joulin, K. Tang, and L. Fei-Fei. Efficient Image and Video Co-localization with Frank-Wolfe Algorithm. *ECCV*, 2014. 2, 8
- [14] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang, et al. T-cnn: Tubelets with convolutional neural networks for object detection from videos. *arXiv preprint arXiv:1604.02532*, 2016. 1, 2, 5, 7, 8
- [15] K. Kang, W. Ouyang, H. Li, and X. Wang. Object detection from video tubelets with convolutional neural networks. In *CVPR*, 2016. 1, 2, 5, 8
- [16] K. Kang and X. Wang. Fully convolutional neural networks for crowd segmentation. *arXiv preprint arXiv:1411.4464*, 2014. 1
- [17] S. Kwak, M. Cho, I. Laptev, J. Ponce, and C. Schmid. Unsupervised Object Discovery and Tracking in Video Collections. *ICCV*, 2015. 8
- [18] S. Li, T. Xiao, H. Li, B. Zhou, D. Yue, and X. Wang. Person search with natural language description. In *CVPR*, 2017. 1
- [19] Y. Li, W. Ouyang, and X. Wang. Vip-cnn: A visual phrase reasoning convolutional neural network for visual relationship detection. In *CVPR*, 2017. 1
- [20] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1
- [21] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, C.-C. Loy, et al. DeepID-net: Deformable deep convolutional neural networks for object detection. *CVPR*, 2015. 1
- [22] A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari. Learning object class detectors from weakly annotated video. *CVPR*, 2012. 2, 6, 8
- [23] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640*, 2015. 1
- [24] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NIPS*, 2015. 1, 3
- [25] J. Shao, K. Kang, C. Change Loy, and X. Wang. Deeply learned attributes for crowded scene understanding. In *CVPR*, 2015. 1
- [26] J. Shao, C.-C. Loy, K. Kang, and X. Wang. Slicing convolutional neural network for crowd video understanding. In *CVPR*, 2016. 1
- [27] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015. 1
- [28] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014. 6
- [29] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CVPR*, 2015. 1
- [30] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *IJCV*, 2013. 3
- [31] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. *ICCV*, 2015. 1
- [32] T. Xiao, S. Li, B. Wang, L. Lin, and X. Wang. Joint detection and identification feature learning for person search. In *CVPR*, 2017. 1
- [33] F. Zhu, X. Wang, and N. Yu. Crowd tracking with dynamic evolution of group structures. In *ECCV*, 2014. 1
- [34] C. L. Zitnick and P. Dollar. Edge Boxes: Locating Object Proposals from Edges. *ECCV*, 2014. 3