TKD: Temporal Knowledge Distillation for Active Perception

Conference Paper · February 2020

DOI: 10.1109/WACV45572.2020.9093437

2 authors:

Mohammad Farhadi Bajestani
Arizona State University
7 PUBLICATIONS **9** CITATIONS
SEE PROFILE

Yezhou 'YZ' Yang
Arizona State University
106 PUBLICATIONS **1,211** CITATIONS
SEE PROFILE

TKD: Temporal Knowledge Distillation for Active Perception

Mohammad Farhadi
Arizona State University
mfarhadi@asu.edu

Yezhou Yang
Arizona State University
yz.yang@asu.edu

Abstract

Deep neural network-based methods have been proved to achieve outstanding performance on object detection and classification tasks. Despite the significant performance improvement using the deep structures, they still require prohibitive runtime to process images and maintain the highest possible performance for real-time applications. Observing the phenomenon that human visual system (HVS) relies heavily on the temporal dependencies among frames from the visual input to conduct recognition efficiently, we propose a novel framework dubbed as TKD: temporal knowledge distillation. This framework distills the temporal knowledge from a heavy neural network-based model over selected video frames (the perception of the moments) to a light-weight model. To enable the distillation, we put forward two novel procedures: 1) a Long-short Term Memory (LSTM)-based key frame selection method; and 2) a novel teacher-bounded loss design. To validate our approach, we conduct comprehensive empirical evaluations using different object detection methods over multiple datasets including Youtube-Objects and Hollywood scene dataset. Our results show consistent improvement in accuracy-speed tradoffs for object detection over the frames of the dynamic scene, compared to other modern object recognition methods. It can maintain the desired accuracy with the throughput of around 220 images per second. Implementation: https://github.com/mfarhadi/TKD-Cloud.

1. Introduction

Object detection plays a critical role in a variety of mobile robot tasks such as obstacle avoidance [3, 44], detection and tracking [2] and object searching [46, 45]. During the last decade, we have witnessed the great success of Convolutional Neural Networks (CNNs)-based methods in the object detection task. This success has led researchers to explore deeper models such as RetinaNet [23] or Faster-RCNN [35], which yield high recognition accuracy. The "secret" sauce behind the success of these deeper and deeper CNNs models is the stacking of repetitive layers and increasing the number of model parameters [5]. This practice becomes possible while the applications are running on infrastructures with high processing capabilities.

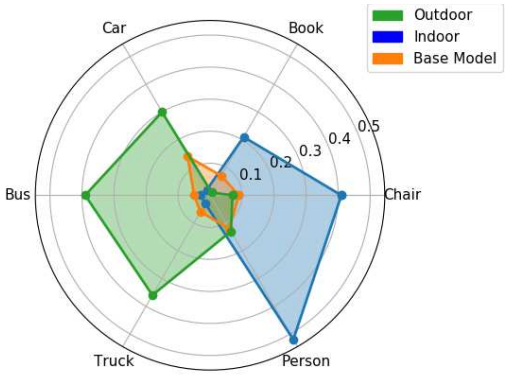


Figure 1: An illustration of our TKD model's actual performance: F-1 score distribution over example object categories in different environments using TKD.

However, the disadvantages of this practice are obvious and the high performance is achieved by the significant growth of the model complexity: stacking up layers and increasing the model parameters which are computationally expensive and also increase the inference time significantly. Hence, these models are not suitable for real-time and embedded visual processing systems, and thus impede their deployment in the era of intelligent robots and autonomous vehicles. The same concerns also lie in the energy conservation and computation limits, since deep models require a large number of matrix multiplications, which are time-consuming and energy-demanding for mobile applications.

The aforementioned concerns trigger various approaches, such as using the alignment of memory and SIMD (Single Instruction, Multiple Data) operations to boost matrix operations [14]. More recently, studies [5] and [18] proposed transferring the knowledge of deep models to shallow models while maintaining the recognition accuracy. Although these approaches do improve the model efficiency, they ignore the temporal dependencies among the frames from dynamic scenes, which is one of the critical capabilities to maintain high recognition accuracy while being

energy-aware.

The motivation for our TKD model comes from the visual adaptation phenomenon observed in the Human Visual Systems (HVS). Visual adaption involves temporary changes in the human perception system when exposed to intense or new stimulus and by the lingering aftereffects when the stimulus is removed [42]. Other studies from [42] show that the visual system adapts to the changes in the environment and this adjustment can happen in a few milliseconds. More specifically, a study from [7] reveals that the face recognition process happens at a higher level of cognition, and later at the stage of visual encoding, we observe that the sensory systems adapt itself to the prevailing environment. This shows that HVS relies heavily on the prior estimation of the objects' appearance distribution to improve the perception capability at the current time-stamp.

Moreover, the adaptation happens both in the "low" and "high" level visual features. The human visual system adapts to the distribution of "low-level" visual features such as color, motion, and texture, as well as the "high-level" visual features such as face classification including identity, gender, expression, or ethnicity [42]. This adaptation can be both short-term and long-term. For instance, our perception system adapts itself to the general visual features of the environment which we are living in for a long time such as faces and colors (like training a model). Also, it can adapt itself dynamically when the environment changes, for example, moving from the indoor environment to the outdoor [42] (like adapting a shallow model). This adaptation capability is essential for our HVS to perform recognition well and efficient, with low energy consumption.

Inspired by the aforementioned findings, we design our TKD framework that utilizes the knowledge distillation techniques. It transfers temporal knowledge from the heavy model to a light model to boost visual processing efficiency while maintaining the heavy model's (a.k.a., oracle model) performance. Figure 1 illustrates the overall goal of this work. In this figure, we show how TKD improves recognition accuracy over different scenes, compared to the oracle model which we assume to be a perfect model. Also, we show the baseline model which is a tiny model with low accuracy compared to oracle recognition due to a much lower number of parameters. TKD achieves higher accuracy by adapting itself to the observed environment. In the case of an indoor scene, the TKD recognition accuracy improves significantly over objects which are more probable to be observed inside a building. In the outdoor case, TKD recognition accuracy improves over the objects such as a car, bus, and truck which are more probable to be observed outside. For a similar amount of model parameters as the baseline tiny model, the TKD will achieve much better performance over the more probable objects by dynamically learning from the oracle model.

To summarize our contributions: 1) we propose an end-to-end trainable framework to transfer the temporal knowledge (a.k.a., the perception of the moment) of the oracle model to the student model; 2) we propose a novel teacher-bonded loss for knowledge distillation which has a simple structure and performs inferences briskly; and 3) we propose an efficient method to select key frames from the dynamic scene, that indicate the right timing to train student model and to improve the detection accuracy. We design and conduct empirical experiments on both the public datasets (the Youtube Object dataset and the Hollywood Scene dataset) as well as on two long videos with multiple scene changes, which validate each of the aforementioned novel design choices, by observing a fast object recognition performance while maintaining high detection accuracy.

2. Related Work

Visual recognition systems, ranging from object recognition [23], action recognition [22], to scene recognition [47] have gained attention in recent years. Significant improvements in recognition accuracy have resulted in economic and societal benefits in AI applications such as autonomous vehicles [20, 21], and IoT systems [39, 40].

Object Detection: Object detection methods based on Convolutional Neural Networks (CNNs) have shown promising results over the past years. There are two main types of object recognition systems which are based on CNNs, one-stage, and two-stage. In one-stage methods, we classify and localize objects in one-stage. Images, when forwarded through the network produce a single output which is then used to classify or localize objects. Some examples of one-stage methods are Yolo [34], RetinaNet [23] and DSSD [13]. These models are faster compared to other methods due to ruining in a single stage. The second types of models are two-stage methods in which classification and localization happen as two different stages, using classification networks and region proposal networks respectively. Two famous two-stage models are FasterRCNN [35], R-FCN [9]. These models reach to higher performance with high intersection over union (IOU). However, Redmon et al. [34] showed at lower IOU (IOU=0.5) one-stage models can perform the same accuracy as two-stage models.

Model Compression: Another thrust of work has focused on reducing the resources consumption of CNNs (due to expensive computation and memory usage) by compressing the network structures [17, 33]. Network pruning is one of well-studied approach which removes unnecessary connections from CNN model, to gain inference speedup [43, 19]. Quantizing [16, 12] and binarizing [33, 1] are two other methods that have been used to reduce network size and computation load. These methods improve performance at the hardware level by reducing the size of weights at the binary code level. However, the standard GPU implementa-

tion remains challenging for these methods to achieve runtime speedup [17]. Also, the advantages of these methods over other one-stage methods without the fully connected layers (the network pruning target in [16]) is not clear.

Domain Adaptation: Object detection in the real world still needs to address challenges such as low image quality, large variance in the backgrounds, illumination variation, etc. These could lead to a significant domain shift between the training, validation and test data. Consequently, the field of domain adaptation has been widely studied in image classification [41, 25] and object detection [6, 8] tasks. These methods improve accuracy on well-known bench-marking datasets. Nevertheless, they typically adopt an offline domain adaptation procedure and do not concern with domain-change during the inference stage.

Knowledge Distillation: Knowledge distillation is another approach to boost accuracy in CNNs. Under the knowledge distillation setting, an ensemble of CNN models or a very deep model will serve as the teacher model, which transfers its knowledge to the student model (shallow model). Hinton et al. [18] proposed a method to apply teacher prediction as a "soft-label" and distill teacher classifier's knowledge to the student. Moreover, they proposed a temperature cross entropy instead of $L2$ distance as the loss function. Romero et al. [36] proposed a so-called "hint" procedure to guide the training of the student model. There are also other approaches to distill knowledge between different domains such as from RGB to depth images [15, 37]. Knowledge distillation has been also applied to the object detection task. Chen et al. [5] proposed a method which adopts all of the soft labeling (labels generated by the teacher), the hard labeling (the ground truth) and the hint procedure to transfer knowledge from the teacher with deep feature extractor to the student with a shallow feature extractor. They adopt a two-stage method (FasterRCNN [35]) in their system. Mehta et al. [27] applied the same procedure to one stage method (Tiny-Yolo v2).

Mullapudi et al. [30] proposed an online model distillation for efficient segmentation. They adopt a light CNN model as a student and a heavy model as a teacher. At the inference time, the student model is trained periodically using the teacher knowledge. However, the naive usage of a fixed period may not be efficient in their approach. Moreover, their shallow model struggles to handle emerging new objects in the scene when these objects are observed in the middle of the fixed period. Here, ours is able to select the period length based on the incoming frames, by which TKD could trigger re-training and thus detecting the emerging new objects, as demonstrated experimentally in Sec. 5.

3. Temporal Knowledge Distillation

The conventional use of knowledge distillation has been proposed for training CNNs based classification models. In these models, we have a dataset $(x_i, y_i), i = 1, 2, ..., n$ where x_i and y_i are input images and the class labels. The student model is trained to optimize the following general loss function (with β is a modulation factor):

$$\begin{aligned} &O_s = Student(x); O_t = Teacher(x), \\ &L(O_s, (y, O_t)) = \beta L_{gt}(O_s, y) + (1-\beta) L_t(O_s, O_t), \end{aligned} \tag{1}$$

where L_t is the loss using teacher output (O_t) and L_{gt} is the loss using ground truth y [27, 5, 18].

In addition to the classification task, object detection also could benefit from the knowledge distillation procedure. However, it's not as straightforward as the classification task. Most notably, the teacher model's output may yield misleading guidance to the student model [5]. The teacher regression result can be contradictory to the ground truth labels, also the output from the teacher regression module is unbounded. To address these issues, [5] proposed a procedure to only adopt teacher's output at beneficial times. For a one-stage object detection setting, [27] optimized the student model with a similar loss function to Eq. 1.

In this paper, we propose a novel and bio-inspired way of adopting the teacher model's knowledge. Namely, temporally estimating the expectation of object labels, their sizes, and shapes based on the previously observed frames or $E[y_i|\alpha_1, \alpha_2, ..., \alpha_{i-1}]$ where y_i is our objects label and α our observations. This expectation changes in time by camera or objects movements, and/or the changing of the field of view. Here, we utilize this extracted knowledge to improve object detection performance. Unlike the previous work such as [27, 5], we are not aiming to improve the feature extractor and/or the general knowledge of the student model. We optimize the decoder inside the student model to adapt it to the current environment. It is done by increasing the likelihood of objects which are more frequently found from the previous observations. Since the model requires online training during the inference stage, it should be able to address the following challenges:

1. Training is a time consuming procedure, running it at the inference stage hurts model efficiency;
2. Selecting the key frames accurately on which the student model needs to be adapted;
3. Objects with low appearance probability may not be detected by the student model after adaptation;
4. The oracle model still introduces noise at locations where there are no objects. Simply training the student model with noisy oracle output decreases the accuracy.

In the following section, we will introduce our approach to address these challenges respectively.

4. Our Approach

In this work, we adopt Yolo-v3 (as teacher) and Tiny-Yolo v3 (as student) [34] as the base object detection methods. These two models are one-stage object detection mod-

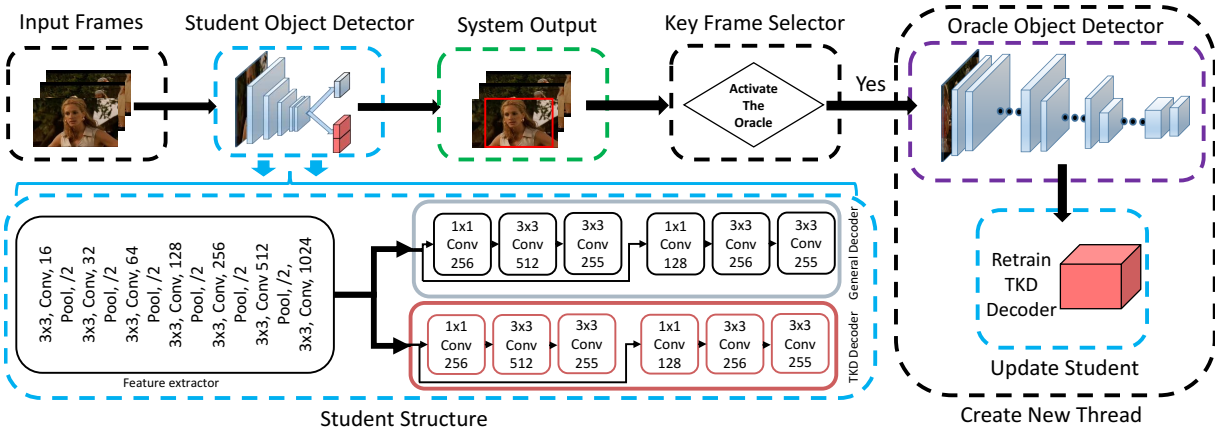


Figure 2: An overview of TKD (Temporal Knowledge Distillation): A low-cost student model is tasked to detect objects in the main thread. To retain high accuracy, a key frame selector decides to activate the oracle model and adapt the student over the environment. Since the execution of the Oracle model and retraining the student model occurs in separate threads, it does not have a significant effect on the inference latency.

els. In both models, object detection is conducted at various layers. The middle layers are used to detect large objects and the last layers to detect small objects. Studies [34], [30] and [23] showed that this strategy successfully improves the object detection accuracy with a significant edge.

As mentioned in Section 1, the overall objective of our system is to estimate the expectation of object labels, their sizes, and shapes on the temporal domain and to improve the performance of the student model. Following this intuition, we put forward a mechanism with a combination of an oracle model (which we consider it as the best possible model) and a student model (which is fast but has considerably lower accuracy compared to the oracle). We are transferring the temporal knowledge of the oracle model to the student model at the inference time. By transferring this knowledge, the student model adapts itself to the current environment or scene. Without loss of generality, We select Yolo-v3 object detection model as the oracle model due to its reliable and dominating performance compared with other one-stage methods. We select Tiny-Yolo model [34] as the student model due to its high base frame rate and having a similar model structure with the Yolo-v3.

4.1. The TKD Architecture

We show our overall framework in Figure 2. In the student model, we include two decoders as the TKD decoder and the general decoder. Then, the pre-trained Yolo-v3 [34] is adopted as the oracle. We run the Oracle model with the input image and the weights of student's TKD decoders get updates at specific frames from the oracle model's result. Finally, we design a decision procedure using an LSTM model, to generate the signals that indicate the right timing to use the Oracle knowledge.

Specifically, we train Tiny-Yolo with a general decoder over the COCO dataset [24]. The design of Tiny-Yolo has two general decoders to improve the accuracy of different object sizes. We first make a copy of the general decoders bounded together as TKD decoder. The TKD decoder is updated during the inference stage. We only update the last three layers of Tiny-Yolo and treat it as the decoder, since it yields enough performance in practice. We keep the general decoder from Tiny-Yolo together with the TKD decoder to make the final detection. TKD decoder and general decoder are executed in two parallel threads which do not increase the latency. This will preserve the chance of detecting viable objects addressing the challenge (3) in Sec. 3.

4.2. Distillation Loss

Before describing our distillation loss, we provide a brief overview of the other distillation loss functions. First, Chen et al. [5] proposed a combination of hint procedure and weighted loss function. They generate boxes and labels using both the student and the teacher model, then calculate two loss values comparing the teacher's output and the ground truth. In the end, they sum up the weighted loss values. If the student model outperforms the teacher model, they continue training only using ground-truth supervision. More recently, Mehta et al. [27] applied the similar procedure to the one-stage object detection models (Tiny-Yolo v2 with some modification). They generate bounding boxes and labels, and apply Non-Maximum Suppression (NMS) to these boxes and then follow the loss function to optimize the student model. The loss is defined in the following equation:

$$L_{final} = L^C_{bb}(b^{gt}_i, \hat{b}_i, b^T_i, o^T_i) + L^C_{cl}(p^{gt}_i, \hat{p}_i, p^T_i, o^T_i) + L^C_{obj}(o^{gt}_i, \hat{o}_i, o^T_i), \quad (2)$$

where $L^C_{bb}, L^C_{cl}, L^C_{obj}$ are objectness loss, classification loss and regression loss which are calculated using both ground truth and the teacher output. Also, $\hat{b_i}, \hat{p_i}, \hat{o_i}$ are bounding box coordinates, class probability and objectness of the the student model. $b^{gt}_i, p^{gt}_i, o^{gt}_i$ and b^T_i, p^T_i, o^T_i are values derived from ground truth and the teacher model output.

In our study of the Yolo-v3 and Tiny-Yolo models, we noticed that the detection layer is the most computationally expensive part. In this layer, several processes are done (sorting, applying softmax to classification cells, removing low confidence boxes, etc.) to produce bounding boxes and then applying NMS to these boxes. These processes are computationally slow due to the multiple steps of processing, and also running over CPU by the implementation. Consequently, directly adopting these loss functions will be also computationally expensive during the inference stage.

With this observation, we adopt the mean square error (MSE) between the tensors generated by the student decoder and the oracle decoder, which should be the fastest method. However, the side effects are also notorious. The oracle model generates noises over some parts of frame which have no object existences; hence directly forcing the student model to retrain will hurt its performance.

Another approach could be calculating the MSE between the tensor cells which have high confidence of object existence. But, the approach will hurt the student's recognition accuracy too. By applying this loss function, the student model tends to generate redundant detection boxes which yield a larger number of false positives.

To alleviate the downsides of both loss designs and still to preserve their advantages, we introduce a novel loss by a combination of them in Equation 3:

$$\begin{aligned} L_{final} &= \sum \|T^H_s - T^H_o\|^2_2 \\ &+ \sum \|T^E_s - ((\lambda * T^E_s) + ((1-\lambda) * T^E_o))\|^2_2, \end{aligned} \tag{3}$$

where $T^H_s \& T^H_o$ are the student and oracle cells with a high chance of object existences and $T^E_s \& T^E_o$ are the cells with a low expectation. More specifically, the first part on the left side of Eq. 3 calculates the MSE between the parts which have high confidence of objects. The second part calculates a modulated MSE between the cells with a low expectation from both the oracle output tensor and the student output tensor. Here, λ is the modulation factor. Figure 3b shows the procedure of creating the target tensor.

By using this loss function, the student model will have a lower chance to generate extra false positives. Also, it would not strictly force the student model to mimic the oracle exactly. We aim to partially address the challenges 1) and 4) in Sec. 3, with such a fast and effective loss function.

4.3. Key Frame Selection

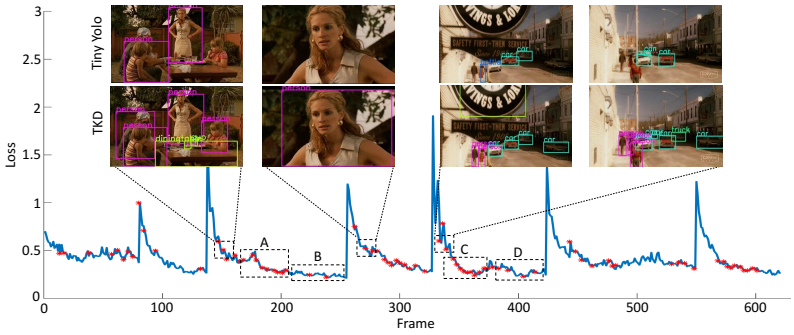
Another crucial module to enable TKD working properly is a procedure to demonically select the time instances to train the student model during the inference stage. Specifically, TKD seeks the frames that by training over them the model has a higher expectation of reducing the loss, thus eventually improves the detection accuracy. For the rest of the paper, we denote these frames as the key frames.

Selecting a larger number of frames as the key frames will hurt the performance since re-training is computationally expensive; While selecting too few number of frames will hurt the detection accuracy as the student may not align well with the oracle model in time. Thus, an effective and fast procedure to select the key frames is highly desired to yield a positive effect on the system's performance.

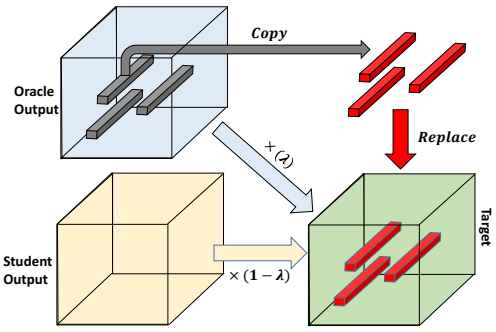
We propose a key frame selection procedure which is both efficient and also practical. First, we check the training prevention factor τ. If the student model has been trained in any last τ frames; we will exit the key selection procedure. It is based on the reasonable assumption that if we have an environment change, it typically takes τ frames that this change to be fully observable. Thus, when we train the student, training for the next τ frames would not be beneficiary. Second, we start our decision process which we formulate in Equations 4:

$$\begin{aligned} &I \in \{0,1\} \begin{cases} 0 & Do\ not\ distill\ knowledge, \\ 1 & Distill\ knowledge, \end{cases} \\ &I = LSTM(F_s) \ \vee \ I_R, \quad I_R \sim B(2, P_t), \\ &P_t = \begin{cases} max((P_{t-1} - 0.05), 0.05) & \Delta L < \sigma, \\ min(2P_{t-1}, 1.0) & \Delta L > \sigma, \end{cases} \end{aligned} \tag{4}$$

where I is the indicator that denotes our final decision. It takes the disjunction of the LSTM's output and the random module's output. We pass the features extracted from the student model F_s (the last layer before the decoder) to the LSTM module (with one LSTM layer & one fully connected layer) which outputs a signal indicating to train the student model or not. Here, it is worth to note that we introduce another binary random module I_R (with binomial distribution $B(2, p_t)$) which decides in a random fashion to train the student model or not. The random procedure is added as a safeguard in case the LSTM model outputs a sequence of erroneous decisions. In the end, we update the LSTM module based on the result feeding back after the training procedure. If the LSTM makes a correct decision where the observed loss decrease $\Delta L < \sigma$ wherein our experiments $\sigma = -0.1$, the random factor P_t will be reduced by 0.05. If the LSTM model makes a wrong decision, we update the LSTM model and double the random factor P_t. Figure 3a shows an example output of key frames selected by our method. We apply knowledge distillation selectively to a few numbers of frames which partially addresses the aforementioned challenges 1) and 2) in Sec. 3.



(a) Key frames selected using TKD over two scenes from the Hollywood scene Dataset [26]. The red crosses indicate the key frames selected by our method. See further discussion in Sec. 5.3.



(b) Target tensor composition.

Figure 3: TKD key frame selection and loss function.

Method	Hollywood Scene Dataset						The pursuit of happiness					
	IOU=0.5		IOU=0.6		IOU=0.75		IOU=0.5		IOU=0.6		IOU=0.75	
	AP	F-1	AP	F-1	AP	F-1	AP	F-1	AP	F-1	AP	F-1
Random Selection	0.71	0.75	0.54	0.68	0.48	0.49	0.65	0.65	0.55	0.58	0.35	0.43
Scene Change Detection	0.68	0.58	0.47	0.50	0.23	0.35	0.54	0.58	0.45	0.53	0.35	0.44
Tiny-Yolo [34]	0.45	0.16	0.38	0.14	0.10	0.28	0.37	0.11	0.25	0.10	0.08	0.06
Tiny-Yolo (73%) + Yolo-v3 (27%)	0.60	0.49	0.59	0.49	0.44	0.46	0.58	0.47	0.52	0.46	0.39	0.44
TKD	**0.75**	**0.76**	**0.58**	**0.69**	**0.49**	**0.50**	**0.73**	**0.67**	**0.59**	**0.61**	**0.40**	**0.46**

Table 1: Performance of TKD with different training methods over Hollywood scene dataset and The pursuit of happiness.

5. Experiments

The presented theoretical framework suggests three hypotheses that deserve empirical tests: 1) TKD can perform visual recognition efficiently, without hurting the recognition performance significantly; 2) the novel loss function can improve online training of the decoder; 3) with our TKD frame selector mechanism, the overall system yields the best performance over other key-frame selection mechanisms, by locating the key frames more accurately (frames which training over them can improve TKD accuracy).

To validate these three hypotheses, we evaluate TKD on the Hollywood scene dataset [26], YouTube-Objects dataset [32], The Pursuit of Happyness [29] and the office [10]. We have trained all the base models (RetinaNet [23], Faster-RCNN [35], Yolo-v3 and Tiny-Yolo [34]) over MS COCO dataset [24]. We implemented the TKD as described in Sec. 4 with two different configurations. First, we perform the process of inference and distillation sequentially among the same thread; the other way, we perform the distillation in a separate thread and run the student and oracle in parallel, both architecture implemented using the PyTorch environment [31]. All experiments are carried out on one single NVIDIA TITAN X Pascal graphics card.

Hollywood scene dataset [26] has 10 classes of scenes distributed over 1152 video. In this dataset, videos are collected from 69 movies. The length of these video clips are from 5 seconds to 180 seconds. The length and diversity of video clips make this dataset a perfect candidate to evaluate our key selector method and the novel loss function.

YouTube-Objects dataset [32] is a weakly annotated dataset from YouTube videos, 10 object classes of the PASCAL VOC Challenge [11] has been used in this dataset. It contains 9 and 24 video clips for each object class which length of these videos are between 30 seconds to 3 minutes. We used this dataset to evaluate TKD's overall performance due to its high-quality objects level annotations.

The pursuit of happyness [29] & The office [10] are two famous movie and TV series. These two video clips contain several scenes which have smooth transitions. The Pursuit of happyness serves a great testbed since it has scenes in different locations such as office, street, etc. It is also more close to the real world scenario from a camera of the intelligent agent. Also, the Office is selected as most of the scenes have been recorded in the same location which make it suitable for testing our novel loss function.

5.1. Ablation Study

As shown in table 1, we compare different strategies to highlight the effectiveness of our proposed novel loss and key frame selector. We consider the output of the oracle model as ground truth and evaluating different methods over it. Here, we compare five methods: 1) TKD with random key frame selection; 2) TKD with Scene Change detection; 3) Tiny-Yolo without any training; 4) Combination of Tiny-Yolo and Yolo-v3 without training; 5) TKD with our proposed key frame selection method.

In the following experiments, we have set the λ to be 0.4 which is obtained heuristically. In 5.3, we will go through the findings which we observed in our search for the best λ.

Random Selection: Here, instead of selecting key frames by our proposed method, decision modules selects

frames purely randomly for further processing. During the testing phase, the probability is set to be 27% (to make sure it selects more frames than our method (25% on average)). Random selection achieves 0.75 F_1 score (IOU=0.5) in the Hollywood scene dataset and achieves 0.65 F_1 score (IOU=0.5) in the pursuit of happiness. On average, it reaches a frame-rate of 89 frames per second (FPS).

Scene Change Detection: This method uses the content-aware scene detection method [4]. It finds areas where the difference between two subsequent frames exceeds the threshold value and used them as key frames for training the student. We selected the threshold with the highest performance and accuracy to report. This method achieves 0.58 F_1 score and 0.58 F_1 score in the Hollywood scene dataset and The pursuit of happyness respectively. This method selected 24% frames as key frames ultimately. On average, the system yields a 93 FPS.

Tiny-Yolo without any training: We test Tiny-Yolo [34] to show the accuracy of a strong baseline model without temporal knowledge distillation. This model achieves 0.16 F_1 score and 0.11 F_1 score in the Hollywood scene dataset and The pursuit of happyness respectively, which are significantly lower than the other mentioned methods. However, This model has 220 FPS, the fastest among all.

Tiny-Yolo + Yolo-v3 without training: In this configuration, we used Tiny-Yolo and Yolo-v3 v3 [34] together. We designed a random procedure which runs Yolo-v3 with a probability of 27% and Tiny-Yolo for the rest of the times. This model achieves 0.49 F_1 score and 0.47 F_1 score in the Hollywood scene dataset and the pursuit of happyness respectively. Frame-rate approaches 89 FPS.

TKD with our key frame selection method: Initially, we set τ (the training prevention factor) to 2 (We observe that the transition between two scenes takes at least 2 frames); along with setting the minimum random selection to 5%. In the Hollywood dataset, our method selects around 26% of frames and the F_1 score achieves 0.76 (IOU=0.5). In the pursuit of happyness movie, our method selects around 24% of frames and the F_1 score reaches to 0.67 (IOU=0.5). On average, the system achieves a frame-rate of 91 FPS sequentially and 220 FPS with running inference and knowledge distillation in parallel.

Table 1 lists the experimental results we observed with these variants. These experiments show, the TKD, while maintaining a similar frame-rate as other methods, it can achieve higher recognition accuracy. To further validate this claim, we conduct one additional experiment on a single-shot movie [28], TKD selects 21% and random procedure selects 27% of the total frames for re-training. They reach comparable F1-score (TKD:0.807, Random:0.812), but our TKD method uses 10400 frames less than the random one.

Method	IOU=0.5	
	mAP	F-1 score
RetinaNet-50 [23]	0.45	0.44
FasterRCNN [35]	0.52	0.50
Tiny-Yolo [34]	0.38	0.33
Tiny-Yolo (73%) + Yolo-v3 (27%)	0.44	0.45
TKD	**0.56**	**0.55**
Oracle (Teacher)		
Yolo-v3 [34]	0.60	0.62

Table 2: Compression of accuracy (IoU=0.5) over Youtube object dataset.

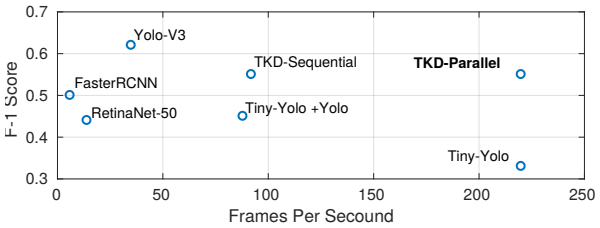


Figure 4: Accuracy and speed in Youtube-Objects Dataset.

5.2. Overall Performance

Table 2 shows mean average precision (mAP) and F_1 score for five different object detection models as well as our TKD method over the Youtube object dataset [32]. For the student models without oracles supervision, we train them to the best performance we could achieve. Not surprisingly, larger or deeper models with larger numbers of parameters perform better than shallower models, while smaller models run faster than larger ones. However, TKD achieves a high detection accuracy compare to RetinaNet, FasterRCNN, Tiny-Yolo, the combination of Tiny-Yolo and Yolo-v3 (same configuration which is described in Sec. 5.1). TKD's detection performance also approaches the performance of the oracle model (Yolo-v3). In this experiment, 25% of frames have been selected for training using the proposed key frames selection method.

To illustrate the accuracy-speed trade-off, we further plot them in Figure 4, where we can see that the TKD archives higher accuracy compare to other shallow methods while still operating far above the real-time speeds with a 91 FPS. The oracle model has a better detection accuracy, but it runs much slower than the TKD.

5.3. Further Study and Discussions

In this section, we provide further insight into the loss function design, the general knowledge distillation idea, and suggest an application of the proposed method.

Loss function: we studied the λ effect over the number of true positives and false positives generated by TKD. All tests are done over an episode from The office [10]. We choose this video since it was recorded in one indoor environment, with a consistent objects distribution. Table 3 shows the student model's detection accuracy varies with the different choices of λ. At $\lambda = 0$, we observed a lower

Figure 5: Computational costs for loss functions.

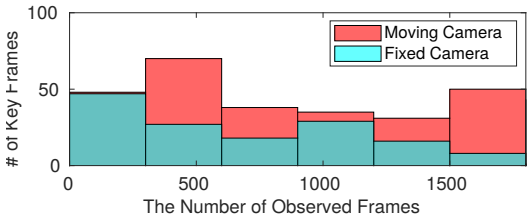


Figure 6: Key frames histogram.

ssd		0	0.2	0.4	0.6	0.8	1
IOU	**AP**	0.47	0.72	0.82	**0.83**	0.79	0.8
0.5	**F-1**	0.36	0.649	**0.676**	0.656	0.634	0.643
#TP		3353	8570	8371	7806	7274	7438
#FP		215	2952	1522	1129	814	841

Table 3: Parameter study of λ over the TKD.

number of false positives since a fewer number of frames (5%) selected by the key frame selection module. With a low λ (except at 0), we observe an increase in false positives as the model tries to generate more boxes and loss function doesn't punish hardly enough onto the student model for generating false positives. With a high λ, we observe drops in the true positive rates since we are forcing the student to learn noises which are likely introduced by the oracle model. Consequently, 0.4 is empirically the best choice here, and we set it as the λ value for all the experiments.

To validate our loss design, we further compare its performance with the one from Mehta et al. [27], where the proposed loss is based on Non-Maximum Suppression algorithm. It is computationally more expensive in comparison with our approach. Figure 5 depicts that, an increasing number of targets from each frame will result in the increasing of execution time for calculating the loss function in [27]. Our loss design has an almost constant execution time, while the proposed loss function by [27] is linearly growing.

Temporal knowledge distillation: Here, we take a closer look at the key selection module. Figure 3a shows its performance over two video clips from the Hollywood scene dataset. Red crosses are frames selected by our proposed method as key frames. At peaks, we have a scene change and logically these points would be the best candidate for training. Following this insight, we observe our model has a lag on detecting these points. Here, we argue that training over these frames is not the best one for improving the student model's accuracy. The scene detection method can identify these points yet table 1 shows it achieves lower accuracy. Figure 3a shows the TKD after detecting a change in loss start stabilizing the model by selecting most of the frames (parts A & C) and for the rest select less number of frames (parts B & D).

The proposed key frame selection method leads to improved performance comparing with [30]'s. Figure 6 shows that the number of selected key frames is adjusted based on the domain change. With the fixed camera case in which the domain does not change, the number of selected frames decreases along observing more frames (validated over the UCF Crime dataset [38]). Indeed, for the case of a moving camera, more key frames are selected to adjust the TKD to the specific domain. Here, the method presented in [30] relies on a static strategy of selecting frames which are chosen manually at the beginning.

For further evaluation, we applied TKD on one episode of the office TV series. Then, we test the trained student model over another episode without any re-training at the inference time. We observed an increase of precision by 6% comparing to the case in which we use the original student model without applying TKD. The result demonstrates the domain adaption capability of our method. Furthermore, it maintains a high recall over other domains which indicates that unseen objects have a chance to be detected. With the method presented in [30], the model loses its generality over unseen objects due to the practice of optimizing the overall model with the new frames.

6. Conclusion and Future Work

In this paper, we propose a novel approach to distill temporal knowledge of an accurate but slow object detection model to a tinier model yielding a light and accurate object detection paradigm for robotic applications, called TKD. We conducted experiments on the Hollywood scene dataset, Youtube object dataset, the pursuit of happyness movie and the office TV series, and empirically validate that TKD maintains a high inference efficiency while achieving a high recognition accuracy. The accuracy even approaches the original oracle model for the object detection task.

The promising experimental results we observed suggest several potential lines of future work: 1) the frame selection procedure could be further optimized to be more selective while maintaining the recognition accuracy; 2) we plan to test our TKD model with an oracle model that follows the two-stage object detection manner; 3) TKD performance can future improve by adopting temporal features in video.

Acknowledgment: The National Science Foundation under the Robust Intelligence Program (1750082), and the IoT Innovation (I-square) fund provided by ASU Fulton Schools of Engineering, GPU and FPGA donations from NVIDIA and Xilinx, are gratefully acknowledged.

References

[1] E. Bank-Tavakoli, S. A. Ghasemzadeh, M. Kamal, A. Afzali-Kusha, and M. Pedram. Polar: A pipelined/overlapped fpga-based lstm accelerator. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2019.

[2] S. Breuers, L. Beyer, U. Rafi, and B. Leibel. Detection-tracking for efficient person analysis: The detta pipeline. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 48–53. IEEE, 2018.

[3] A. Carrio, S. Vemprala, A. Ripoll, S. Saripalli, and P. Campoy. Drone detection using depth maps. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1034–1037. IEEE, 2018.

[4] B. Castellano. Pyscenedetect, 2018.

[5] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker. Learning efficient object detection models with knowledge distillation. In *Advances in Neural Information Processing Systems*, pages 742–751, 2017.

[6] Y. Chen, W. Li, C. Sakaridis, D. Dai, and L. Van Gool. Domain adaptive faster r-cnn for object detection in the wild.

[7] C. W. Clifford, M. A. Webster, G. B. Stanley, A. A. Stocker, A. Kohn, T. O. Sharpee, and O. Schwartz. Visual adaptation: Neural, psychological and computational aspects. *Vision research*, 47(25):3125–3131, 2007.

[8] D. Dai and L. Van Gool. Dark model adaptation: Semantic image segmentation from daytime to nighttime. *arXiv preprint arXiv:1810.02575*, 2018.

[9] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016.

[10] G. Daniels. The Office, 2013.

[11] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[12] M. Farhadi, M. Ghasemi, and Y. Yang. A novel design of adaptive and hierarchical convolutional neural networks using partial reconfiguration on fpga. In *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–7. IEEE, 2019.

[13] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. Dssd: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017.

[14] Y. Gong, L. Liu, M. Yang, and L. Bourdev. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*, 2014.

[15] S. Gupta, J. Hoffman, and J. Malik. Cross modal distillation for supervision transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2827–2836, 2016.

[16] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[17] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, pages 1135–1143, 2015.

[18] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *stat*, 1050:9, 2015.

[19] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[20] M. Khayatian, Y. Lou, M. Mehrabian, and A. Shirvastava. Crossroads+: A time-aware approach for intersection management of connected autonomous vehicles. *ACM Transactions on Cyber-Physical Systems*, 4(2):20, 2019.

[21] M. Khayatian, M. Mehrabian, and A. Shrivastava. Rim: Robust intersection management for connected autonomous vehicles. In *2018 IEEE Real-Time Systems Symposium (RTSS)*, pages 35–44. IEEE, 2018.

[22] C. Lea, R. Vidal, and G. D. Hager. Learning convolutional action primitives for fine-grained action recognition. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1642–1649. IEEE, 2016.

[23] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[24] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014*, pages 740–755. Springer, 2014.

[25] H. Lu, L. Zhang, Z. Cao, W. Wei, K. Xian, C. Shen, and A. van den Hengel. When unsupervised domain adaptation meets tensor representations.

[26] M. Marszałek, I. Laptev, and C. Schmid. Actions in context. In *IEEE Conference on Computer Vision & Pattern Recognition*, 2009.

[27] R. Mehta and C. Ozturk. Object detection at 200 frames per second. *arXiv preprint arXiv:1805.06361*, 2018.

[28] S. Mokri. Fish and cat, 2013.

[29] G. Muccino. The pursuit of happyness, 2008.

[30] R. T. Mullapudi, S. Chen, K. Zhang, D. Ramanan, and K. Fatahalian. Online model distillation for efficient video inference. *arXiv preprint arXiv:1812.02699*, 2018.

[31] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[32] A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari. Learning object class detectors from weakly annotated video. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3282–3289. IEEE, 2012.

[33] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

[34] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[35] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[36] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[37] J.-C. Su and S. Maji. Adapting models to signal degradation using distillation. *arXiv preprint arXiv:1604.00433*, 2016.

[38] W. Sultani, C. Chen, and M. Shah. Real-world anomaly detection in surveillance videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6479–6488, 2018.

[39] N. H. Tonekaboni, S. Kulkarni, and L. Ramaswamy. Edge-based anomalous sensor placement detection for participatory sensing of urban heat islands. In *2018 IEEE International Smart Cities Conference (ISC2)*, pages 1–8. IEEE, 2018.

[40] N. H. Tonekaboni, L. Ramaswamy, D. Mishra, A. Grundstein, S. Kulkarni, and Y. Yin. Scouts: A smart community centric urban heat monitoring framework. In *Proceedings of the 1st ACM SIGSPATIAL Workshop on Advances on Resilient and Intelligent Cities*, pages 27–30. ACM, 2018.

[41] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[42] M. A. Webster. Visual adaptation. *Annual review of vision science*, 1:547–567, 2015.

[43] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2074–2082, 2016.

[44] S. Yaghoubi and G. Fainekos. Worst-case satisfaction of stl specifications using feedforward neural network controllers: a lagrange multipliers approach. *ACM Transactions on Embedded Computing Systems (TECS)*, 18(5s):107, 2019.

[45] X. Ye, Z. Lin, J.-Y. Lee, J. Zhang, S. Zheng, and Y. Yang. Gaple: Generalizable approaching policy learning for robotic object searching in indoor environment. *IEEE Robotics and Automation Letters*, 4(4):4003–4010, 2019.

[46] X. Ye, Z. Lin, H. Li, S. Zheng, and Y. Yang. Active object perceiver: Recognition-guided policy learning for object searching on mobile robots. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6857–6863. IEEE, 2018.

[47] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014.