IJCV manuscript No.
(will be inserted by the editor)

A Coarse-to-Fine Framework for Resource Efficient Video Recognition

Zuxuan Wu[1], Hengduo Li[2], Yingbin Zheng[3], Caiming Xiong[4], Yu-Gang Jiang[1], Larry S. Davis[2]

Received: date / Accepted: date

Abstract Deep neural networks have demonstrated remarkable recognition results on video classification, however great improvements in accuracies come at the expense of large amounts of computational resources. In this paper, we introduce LiteEval for resource efficient video recognition. LiteEval is a coarse-to-fine framework that dynamically allocates computation on a per-video basis, and can be deployed in both online and offline settings. Operating by default on low-cost features that are computed with images at a coarse scale, LiteEval adaptively determines on-the-fly when to read in more discriminative yet computationally expensive features. This is achieved by the interactions of a coarse RNN and a fine RNN, together with a conditional gating module that automatically learns when to use more computation conditioned on incoming frames. We conduct extensive experiments on three large-scale video benchmarks, FCVID, ActivityNet and Kinetics, and demonstrate, among other things, that LiteEval offers impressive recognition performance while using significantly less computation for both online and offline settings.

Keywords Conditional computation · Video classification · Efficient recognition

This work was supported in part by National Natural Science Foundation of China (#62032006).

Zuxuan Wu
E-mail: zxwu@fudan.edu.cn

Hengduo Li
E-mail: hdli@cs.umd.edu

Yingbin Zheng
E-mail: zyb@videt.cn

Caiming Xiong
E-mail: cxiong@salesforce.com

Yu-Gang Jiang (Corresponding author)
E-mail: ygj@fudan.edu.cn

Larry S. Davis
E-mail: lsd@cs.umd.edu

[1]Fudan University, Shanghai, China
[2]University of Maryland, College Park MD, USA
[3]Videt Lab., Shanghai, China
[4]Salesforce Research, CA, USA

1 Introduction

Recent years have witnessed tremendous progress of Convolutional Neural Networks (CNNs) in a multitude of computer vision tasks like image classification (He et al., 2015; Xie et al., 2017; Hu et al., 2018), object detection (He et al., 2017; Ren et al., 2015), action recognition (Wang et al., 2018a, 2016; Feichtenhofer et al., 2019), *etc*. However, the stunning performance of CNN models are accompanied by increased network depth and model parameters, which limits their deployment in many real-world applications that are oftentimes resource-constrained such as online image/video recognition services, autonomous cars, navigation robots, *etc*. Promising solutions to mitigate this issue include neural network pruning/compression (Chen et al., 2015; Rastegari et al., 2016; Li et al., 2017) and compact/efficient architecture design suitable for embedded devices (Howard et al., 2017; Iandola et al., 2016; Feichtenhofer, 2020; Tran et al., 2015). However, these approaches treat all samples equally, and generate one-size-fits-all models that allocate the same amount of computation regardless of sample complexity.

While lightweight/compact networks usually achieve good results when classifying the majority of samples, much deeper and computationally expensive models, if

not an ensemble of models, are required to achieve competitive performance on challenging benchmarks like ImageNet (Deng et al., 2009) and COCO (Lin et al., 2014)—they are often better at recognizing corner cases that lie in the tail of the data distribution (Liu et al., 2019). It is worth pointing out the computational cost of CNNs is also highly related to the input resolution of the model besides the depth of the model. For instance, when running inference with a ResNet-101 network on spatially downsampled images (112×112), 74% of computation can be saved (measured by giga floating point operations) with reasonable recognition results compared to the standard setting where images with a size of 224×224 are used.

Inspired by these observations, in this paper, we ask the following question: can we design a framework with modules of different complexity operating on different image scales and then can we learn to automatically activate the desired module conditioned on input samples with an aim to save computation while offering reliable recognition results? The intuition behind is that low-cost modules are used by default to classify easy samples (*e.g.*, images with centered objects and front-facing views) using coarse scale inputs, while powerful yet computationally expensive modules are activated only when necessary to examine finer details to cope with hard samples (*e.g.*, images with occlusion and cluttered background). The idea is conceptually similar to human perception mechanisms—we pay more attention to complex scenarios while for most objects we can recognize them easily with a glance.

With this in mind, we explore the problem of adaptively allocating computational resources on a per-input basis for resource efficient video recognition. The reasons that we consider video recognition with constrained resources are two-folds: (1) Video recognition is much more computationally intensive compared to image recognition due to the additional temporal dimension in video data. As a result, developing resource efficient video recognition systems is critical, as computation relates to energy consumption, which should be minimized to be cost-effective and environmental-friendly. (2) There are large intra-class and inter-class variations in video data, and hence computational resources required are expected to differ conditioned on video content. For example, a highly confident prediction can be computed by simply observing a single frame for videos that contain relatively static objects (*e.g.*, "panda" and "giraffe") and scenes (*e.g.*, "forest" and "sea"). On the other hand, for motion/action-intensive videos, more frames are demanded to differentiate fine-grained categories like "drinking beer" from "drinking tea". In addition, even for videos within the same category, computational resources required to make correct predictions are different. For instance, sports videos that are professionally recored usually contain less camera motion compared to user-generated videos produced by hand-held mobile devices or ego-centric cameras.

In this paper, we introduce LiteEval, a coarse-to-fine framework that dynamically allocates computation conditioned on incoming video frames, suitable for both online and offline video classification. LiteEval operates on coarse information for low-cost evaluation by default and takes in fine information that is computationally expensive to obtain only when necessary. More specifically, LiteEval consists of a coarse RNN, a fine RNN and a conditional gating module. The coarse RNN takes in features derived from downsampled image frames (coarse information) with a lightweight CNN model; the fine RNN, on the other hand, examines incoming frames more carefully to obtain finer details (fine information) with a more powerful CNN. The conditional gating module dynamically decides the granularity of information to use. Given a sequence of video frames, LiteEval, at each time step, extracts coarse features from the current downsampled video frame and updates the hidden states of the coarse RNN to model temporal information over time. The conditional gating module then decides whether further examination is needed to obtain detailed information of the current input. If it chooses to use more computation, fine features are then computed with the computationally expensive CNN and fed into the fine RNN for temporal modeling; otherwise, the coarse RNN and the fine RNN are synchronized such that the fine RNN encompasses all information seen so far to make predictions. Then, LiteEval analyzes the next incoming frame. LiteEval processes video frames in a recurrent and efficient manner and can be deployed in both online and offline scenarios. Figure 1 gives an overview of the framework.

We evaluate our approach on three large-scale video datasets for generic video classification (FCVID (Jiang et al., 2018)), "untrimmed" activity recognition (ActivityNet (Heilbron et al., 2015)), and "trimmed" action recognition (Kinetics (Kay et al., 2017)). We consider video recognition under both online and offline settings. We show that, for offline video recognition, LiteEval offers accuracies that are on par with the strong and popular uniform sampling strategy while requiring significantly less computation, and it also achieves better results than efficient video recognition approaches in recent literatures (Yeung et al., 2016; Fan et al., 2018). We also demonstrate LiteEval can be effectively deployed for online video recognition to accommodate different computational budgets. Furthermore, we show that LiteEval is compatible with state-of-the-art video

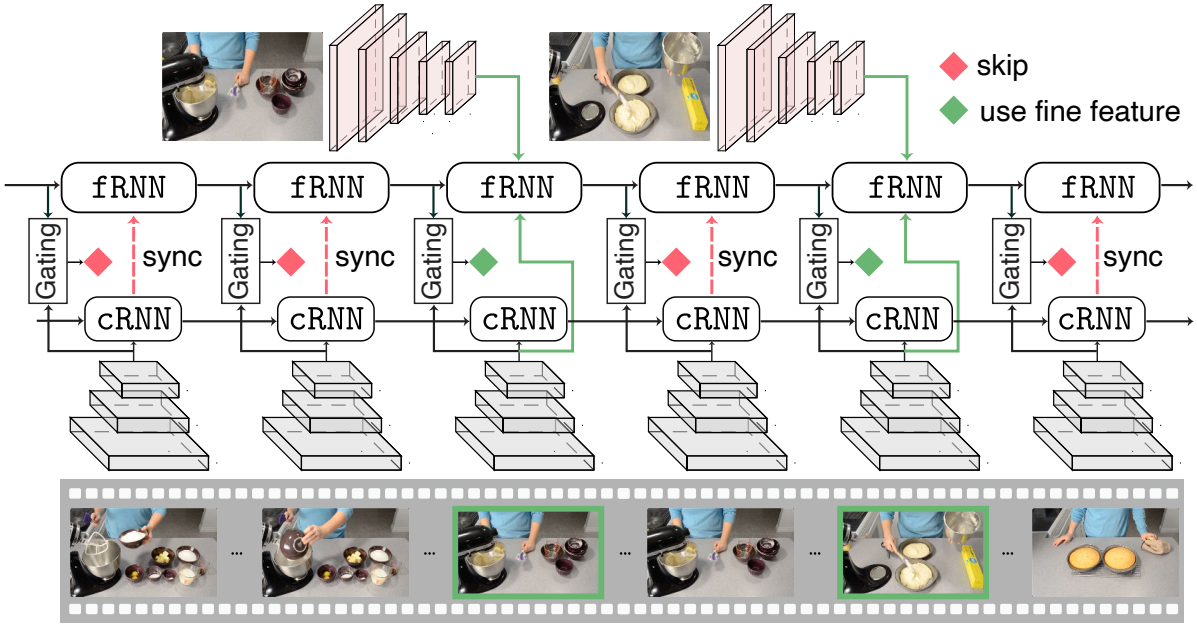


Fig. 1 An overview of the proposed framework. At each time step, coarse features are computed with a lightweight CNN and are then used together with historical information to decide whether more careful examination for the current frame is needed. If further inspection is required, fine features are computed as inputs to the fine RNN; otherwise the two RNNs are synchronized. See texts for more details.

backbones and we provide qualitative evidence that fine feature usage differs for samples in different categories and within the same class.

A preliminary version of this paper appeared in (Wu et al., 2019b). The current paper contains a thorough review of recent literatures on efficient video analysis; more detailed descriptions of the approach; more analysis on online video recognition; new experiments of using different recurrent neural networks in addition to LSTMs; new experiments with modern state-of-the-art 2D and 3D backbones, demonstrating that LiteEval is a generic framework; additional experiments on Kinetics (Kay et al., 2017) and Breakfast (Kuehne et al., 2014).

2 Related Work

Video Classification. Extensive study has been conducted on designing effective and robust models for video classification, where the typical design principles to date focus on equipping deep neural networks for 2D tasks, *e.g.* image classification, with the ability for temporal modeling across different video frames. One line of research applies 2D image models directly to multiple frames and then temporally aggregates frame-level features with pooling (Simonyan and Zisserman, 2014; Wang et al., 2016; Feichtenhofer et al., 2016) and recurrent networks (Ng et al., 2015; Donahue et al., 2015; Li et al., 2016). Another direction expands 2D models to 3D models by replacing 2D convolutions with their 3D counterparts that jointly model temporal and spatial semantics and applies 3D networks to stacked frames (Tran et al., 2015; Kay et al., 2017; Tran et al., 2018; Qiu et al., 2017; Feichtenhofer et al., 2019; Feichtenhofer, 2020). Inputs such as optical flow are also utilized to provide explicit temporal information in existing approaches (Simonyan and Zisserman, 2014; Wang et al., 2016). Despite achieving superior classification performance on standard benchmarks, current state-of-the-art video classification models are extremely resource-demanding mainly due to the computationally heavy model architectures and a large number of uniformly sampled input frames/snippets for all videos, which we posit is an overkill for easy samples. Our approach aims at dynamically allocating computational resources on a per-input basis—lower-resolution inputs and lightweight CNNs are utilized by default and finer information is used only when necessary, aiming to save computational resources while offering reliable classification accuracies.

Conditional Computation. The idea of conditional computation has been widely explored in the image domain. Earlier methods like cascaded classifiers (Viola and Jones, 2004) are proposed to save computation by quickly rejecting easy negative candidates for face detection. Many adaptive computation approaches have also been developed in deep neural networks recently by learning to switch decision branches with different

computational cost, including skipping layers (Wang et al., 2018b; Veit and Belongie, 2018; Wu et al., 2018b) or selecting channels (Bejnordi et al., 2020; Lin et al., 2017) in a large network dynamically, or performing early exiting with auxiliary heads conditioned on inputs (Huang et al., 2018b; Bolukbasi et al., 2017). There are also several recent approaches learning to choose different resolutions (Yang et al., 2020; Uzkent and Ermon, 2020) or select salient regions out of the entire image as input for faster inference (Najibi et al., 2019; Gao et al., 2018). While related to these approaches, our method focuses on the task of video classification and learns to dynamically determine whether to use computationally expensive components like high-resolution inputs and powerful backbone architectures in a network on a per-input basis.

Efficient Video Analysis. While most work focuses on improving video classification performance by designing robust models, some recent attempts have also been made for efficient video analysis (Zhang et al., 2016; Wu et al., 2018a; Fan et al., 2018; Yeung et al., 2016; Wu et al., 2019c; Korbar et al., 2019; Zolfaghari et al., 2018; Tran et al., 2018; Feichtenhofer, 2020). Several lightweight modules for efficient temporal modeling have been introduced such as a relational module in (Zhou et al., 2018) and a temporal shift module in (Lin et al., 2019). Recent advances in efficient 2D CNNs such as group convolutions (Howard et al., 2017; Sandler et al., 2018) are also explored in the video domain (Tran et al., 2018; Chen et al., 2018; Tran et al., 2019). More recently, some lightweight 3D CNNs are proposed to save computation (Tran et al., 2018, 2019; Feichtenhofer, 2020). However, these approaches all use a fixed set of parameters regardless of the complexity of input videos. In contrast, LiteEval is a general dynamic inference framework for efficient video classification, leveraging RNNs to aggregate temporal information and making feature usage decisions over time. Our method is by design complementary to 3D CNNs, as the input features can be easily replaced by the snippet-level feature from 3D CNNs, as will be shown in our experiments.

Apart from designing efficient network architectures, adaptive computation is also explored in some existing work. (Yeung et al., 2016) utilize policy gradient methods to train an agent that selects informative frames and predicts when to stop inference for action detection. A fast forward agent is introduced by Fan *et al.* that decides how many frames to skip at a certain time step (Fan et al., 2018). While being conceptually similar, *i.e.* targeting at skipping redundant frames, our framework is easier to train than policy search methods (Fan et al., 2018; Yeung et al., 2016) as it is fully differentiable. Moreover, our framework is suitable for not only offline inference but also online settings taking video streams as inputs, since the framework does not require access to future frames.

3 Approach

LiteEval consists of a coarse RNN and a fine RNN that work cooperatively, operating on visual information at different scales. It also contains a conditional gating module which learns to switch between different feature scales conditioned on inputs. More specifically, taking in a sequence of video frames, LiteEval aims to learn, at each time step, whether to obtain finer details by computing discriminative features on high resolution inputs. The decision is made based on historical information and a quick glance of the current frame at a coarse scale. This results in a model that operates on low-cost features by default and computes high-cost features only when necessary to take in fine details. The framework is optimized in an end-to-end manner to reduce the overall computational cost while offering reliable recognition results. Below, we elaborate different components of the framework, and introduce the optimization for the framework.

3.1 A Coarse-to-Fine Framework

Coarse RNN. The coarse RNN takes in low-cost features that are computed at a coarse image scale using a lightweight CNN model (see Sec. 4.1 for details), glimpsing over incoming video frames efficiently for an overview of the video. More specifically, at t-th time step, the lightweight CNN model first computes features $\boldsymbol{v}_t^c$ on downsampled video frames. The coarse features are then used in combination with hidden states from the previous time step $\boldsymbol{h}_{t-1}^c$ as inputs to the coarse RNN, outputting hidden states of the current step $\boldsymbol{h}_t^c$:

$$\boldsymbol{h}_t^c = \texttt{cRNN}(\boldsymbol{v}_t^c, \boldsymbol{h}_{t-1}^c). \tag{1}$$

It is worth pointing out the RNN here can be instantiated with different types of recurrent networks such as LSTMs (Hochreiter and Schmidhuber, 1997), GRU (Cho et al., 2014) and SRU (Lei et al., 2017), as will be shown in the experiments. We mainly use LSTMs in our paper, in which $\boldsymbol{h}_t^c$ additionally contains cell states. Without loss of generality, we use $\boldsymbol{h}_t^c$ to denote all hidden states.

Conditional gating module. The coarse RNN observes incoming video frames quickly without consuming too much computation, however this is not sufficient as important details that are needed to differentiate subtle actions are neglected. For instance, finer details are required to separate fine-grained categories like "drinking

tea" and "drinking coffee". To remedy this, LiteEval uses a conditional gating module to dynamically decide whether more discriminative information is needed from the current video frame. In particular, the gating module contains a one-layer MLP that computes the unnormalized probability to extract fine features at a high resolution with a more powerful CNN model:

$$\boldsymbol{b}_t \in \mathbb{R}^2 = \boldsymbol{W}_g^\top [\boldsymbol{v}_t^c, \boldsymbol{h}_{t-1}^f]. \tag{2}$$

Here $\boldsymbol{W}_g$ denotes the weight matrix for the conditional gate, $\boldsymbol{h}_{t-1}^f$ denotes the hidden states for the fine RNN (as will be discussed below) from the previous time step, and $[\,,\,]$ represents the concatenation of features. Since the gating module aims to learn whether to compute features at a finer scale based on $\boldsymbol{b}_t$, this requires making binary decisions, which is non-differentiable and thus hard to optimize in supervised frameworks. Here, we define a Bernoulli random variable B_t to make decisions through sampling from $\boldsymbol{b}_t$. We will introduce how to learn such a parameterized gating function in detail in Section 3.2.

Fine RNN. When the gating module decides to examine the current video frame more carefully (*i.e.*, $B_t = 1$), we then compute more discriminative features with a computationally expensive CNN using high resolution inputs. The features are further input to the fine RNN for temporal modeling. More specifically, fine and coarse features of the current time step, *i.e.*, $\boldsymbol{v}_t^f$ and $\boldsymbol{v}_t^c$ respectively, are concatenated with previous hidden states $\boldsymbol{h}_{t-1}^f$ as inputs to the fine RNN to produce current hidden states:

$$\widetilde{\boldsymbol{h}_t^f} = \texttt{fRNN}([\boldsymbol{v}_t^c, \boldsymbol{v}_t^f],\, \boldsymbol{h}_{t-1}^f) \tag{3}$$

$$\boldsymbol{h}_t^f = (1 - B_t)\boldsymbol{h}_{t-1}^f + B_t \widetilde{\boldsymbol{h}_t^f}. \tag{4}$$

If the gating module decides to skip high resolution inputs (*i.e.*, $B_t = 0$), LiteEval reuses hidden states from the previous time step to save computation.

Synchronizing the `cRNN` with the `fRNN`. The coarse CNN takes in low-cost features for each incoming video frame, and thus it contains all available information seen so far. On the other hand, the fine RNN only contains knowledge from high resolution frames determined by the conditional module. One can consider the `fRNN` stores fine-grained details while the `cRNN` provides context information from all seen frames. For improved recognition performance, we wish to use information seen in both RNN models. Since these two RNN models are asynchronous (the coarse RNN runs faster and is always ahead of the fine RNN, observing more frames), directly concatenating their hidden states is not feasible. Instead, we simply synchronize them by a copy operation. More specifically, at the t-th step, when the gating module opts out of the computation of fine features (*i.e.*, $B_t = 0$ in Equation 4), instead of using $\boldsymbol{h}_{t-1}^f$ directly, we synchronize the hidden states by copying:

$$\boldsymbol{h}_t^f = [\boldsymbol{h}_t^c, \boldsymbol{h}_{t-1}(D^c + 1 : D^f)], \quad \text{if } B_t = 0, \tag{5}$$

where D^c and D^f denote the dimension of $\boldsymbol{h}^c$ and $\boldsymbol{h}^f$, respectively (we assume the $D^c < D^f$). As a result, the hidden states in the fine RNN contains all information seen so far and can be readily used to emit predictions at any time:

$$\boldsymbol{p}_t = \texttt{softmax}(\boldsymbol{W}_p^\top \boldsymbol{h}_t^f), \tag{6}$$

where $\boldsymbol{p}_t$ represents the prediction of the t-th step and $\boldsymbol{W}_p$ is the weights for the classifier.

3.2 Optimization

We denote $\Theta = \{\Theta_{\texttt{cRNN}}, \Theta_{\texttt{fRNN}}, \Theta_g\}$ as all trainable parameters in the framework, where $\Theta_{\texttt{cRNN}}$ and $\Theta_{\texttt{fRNN}}$ are the parameters in the coarse and fine RNNs, respectively and Θ_g are parameters for the conditional gating module [1]. During training, LiteEval uses predictions from the last time step T as the video-level predictions, and it is trained to optimize the following loss function:

$$\begin{aligned}\underset{\Theta}{\text{minimize}}\ \mathbb{E}_{\substack{B_t \sim \texttt{Bernoulli}(\boldsymbol{b}_t;\Theta_g) \\ (\boldsymbol{x},\boldsymbol{y}) \sim D_{train}}} [- \boldsymbol{y}\, \texttt{log}(\boldsymbol{p}_T(\boldsymbol{x};\Theta)) \\ + \lambda(\frac{1}{T}\sum_{t=1}^{T} B_t - \gamma)^2],\end{aligned} \tag{7}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ represent a video and its corresponding one-hot label vector sampled from the training set D_{train}. The first term is a standard cross-entropy loss for classification and the second term constrains the usage of fine features to a predefined target γ, where $\frac{1}{T}\sum_{t=1}^{T} B_t$ is the fraction of the number of times fine features are used over the entire time horizon. In addition, λ controls the trade-off between recognition accuracy and computational cost.

However, directly solving Equation 7 is challenging since determining whether to use fine features from high resolution inputs requires making binary decisions through sampling from a Bernoulli distribution parameterized by Θ_g. One approach to address this is to solve the optimization in Equation 7 in a reinforcement learning framework—defining states and actions and then associating each action taken with a carefully designed reward. The reinforcement learning framework can be trained with policy gradient methods (Sutton and Barto,

[1] We absorb the weights of the classifier $\boldsymbol{W}_p$ into $\Theta_{\texttt{fRNN}}$.

Algorithm 1: Algorithm of LiteEval.

Input: A video stream with T time steps.
Initialize the weights of `cRNN`, `fRNN` and the gating module `G`: $\Theta_{\texttt{cRNN}}, \Theta_{\texttt{fRNN}}, \Theta_g$.
for $t \leftarrow 0$ *to* T **do**
  Compute coarse feature $\boldsymbol{v}_t^c$
  $\boldsymbol{h}_t^c = \texttt{cRNN}(\boldsymbol{v}_t^c, \boldsymbol{h}_{t-1}^c)$
  Compute gating probabilities $\boldsymbol{b}_t$
  if *training* **then**
    $B_t \sim \texttt{Gumbel-Softmax}(\boldsymbol{b}_t)$
  else
    $B_t = \arg\max(\boldsymbol{b}_t)$
  end
  Compute fine feature $\boldsymbol{v}_t^f$ when $B_t = 1$
  Synchronize two RNNs when $B_t = 0$
  $\boldsymbol{p}_t = \texttt{softmax}(\boldsymbol{W}_p^\top \boldsymbol{h}_t^f)$
end

1998) to learn binary decisions. However, reinforcement learning methods are generally difficult to train due to large variance in sampling (Sutton and Barto, 1998). In this work, we leverage a Gumbel-Max trick to make the framework fully differentiable. In particular, given a discrete categorical variable $\hat{B}$ with class probabilities $P(\hat{B} = k) \propto b_k$, where $b_k \in (0, \infty)$ and $k \leq K$ (K represents the total number of categories; $K = 2$ in LiteEval), the Gumbel-Max (Hazan and Jaakkola, 2012; Maddison et al., 2017) trick suggests the sampling from a categorical distribution can be done through:

$$\hat{B} = \arg\max_k(\texttt{log}\, b_k + G_k). \tag{8}$$

Here $G_k = -\texttt{log}\,(-\texttt{log}\,(U_k))$ is the Gumbel noise and U_k denote i.i.d samples drawn from $\texttt{Uniform}\,(0, 1)$. While the arg max operation in Equation 8 is not differentiable, we can use `softmax` as a continuous relaxation of arg max (Maddison et al., 2017; Jang et al., 2017):

$$B_i = \frac{\texttt{exp}((\texttt{log}\, b_i + G_i)/\tau)}{\sum_{j=1}^{K} \texttt{exp}((\texttt{log}\, b_j + G_j)/\tau)} \quad \text{for } i = 1, .., K \tag{9}$$

where τ is a temperature hyper-parameter governing the discreteness in the output vector B. Consider the extreme case when $\tau \to 0$, Equation 9 generates the same samples as Equation 8.

At each time step, LiteEval samples from a Gumbel-Softmax distribution parameterized by the weights of the conditional gating module Θ_g, enabling the learning of binary decisions in a fully differentiable manner. As suggested in (Jang et al., 2017), we also decrease the temperature from a high value to encourage exploration to a smaller positive value. Algorithm 1 summarizes the overall algorithm of our framework.

4 Experiments

4.1 Experimental Setup

Datasets and evaluation metrics. We evaluate our framework on three large-scale video classification benchmarks, *i.e.*, FCVID, ActivityNet and Kinetics. More specifically, FCVID (Fudan-Columbia Video Dataset) (Jiang et al., 2018) consists of 91,223 YouTube videos (with an average duration of 167 seconds) annotated into 239 classes, covering popular topics like "birthday party", "marriage proposal", "making pizza", *etc*. The dataset is divided into a training set with 45,611 videos and a testing set with 45,612 videos. While FCVID contains generic video categories, videos in ActivityNet (Heilbron et al., 2015) are action/activity-focused like "drinking beer", "drinking coffee", "fencing", *etc*. ActivityNet contains $20K$ videos (with an average duration of 117 seconds) belonging to 200 classes. In our evaluation, we adopt the $v1.3$ split which divides the dataset into a training set of 10,024 videos, a validation set of 4,926 videos and a testing set of 5,044 videos. Since labels on the testing set are not publicly available, we report the performance on the validation set. We use FCVID and ActivityNet since videos in both datasets are untrimmed and are relatively long, for which we believe efficient recognition is very important. To demonstrate our framework is also applicable to "trimmed" motion-intensive videos, we additionally evaluate LiteEval on Kinetics (Kay et al., 2017). Kinetics consists of "trimmed" YouTube clips with an average duration of 10 seconds belonging to 400 categories. Following (Feichtenhofer et al., 2019), we use $\sim$240K videos for training and $\sim$20K videos for evaluation.

For offline prediction on ActivityNet and FCVID, we use mean average precision (mAP) to measure the performance following (Heilbron et al., 2015; Jiang et al., 2018). On Kinetics, we report top-1 accuracy following (Kay et al., 2017). For online recognition, we report top-1 accuracy since mAP is a ranking-based metric based on the entire test set and is not suitable online recognition (we do observe similar trends with both metrics). The computational cost is measured by giga floating point operations (GFLOPs), which is a hardware independent metric.

Implementation details. To compute coarse features, we use a lightweight MobileNetv2 (Sandler et al., 2018) model on spatially downsampled inputs (*i.e.*, 112×112). The MobileNetv2 model offers a top-1 accuracy of 52.3% on ImageNet when images are resized to 112×112. For fine feature computation, we use a resolution of 224×224 for input frames to extract features.

We mainly experiment with a ResNet-101 (He et al., 2016) model and compute features from its penultimate layer. The model is pretrained on ImageNet with a top-1 accuracy of 77.4% and it is further finetuned on target datasets for improved performance. More spefically, for both ResNet-101 and MobileNet, we use pretrained models from ImageNet and extract video frames from all the datasets. Then, these CNN backbones are finedtuned on target datasets as image classifcation tasks. To demonstrate that LiteEval is a generic framework and can be used in combination with state-of-the-art video recognition models, we additionally use two more powerful backbones for fine feature computations: (1) a SlowFast-8×8 network with a backbone of ResNet-50 (Feichtenhofer et al., 2019), whose inputs are 8 stacked RGB frames sampled uniformly from 64 frames with a stride of 8. It obtains a top-1 accuracy of 77.0% on Kinetics (Kay et al., 2017) and further finetuned on target datasets; (2) a Dual Path Network (DPN-107) model (Chen et al., 2017), which achieves a top-1 accuracy 80.3% on ImageNet, and is finetuned with temporal segment networks (Wang et al., 2016) for temporal modeling.

Our implementation is based on PyTorch on four NVIDIA V100 GPUs. We use Adam (Yao et al., 2012) as the optimizer and we fix the learning rate to be $1e-4$ and set λ to 2. We use a batch of 128, 256 and 512 for ActivityNet, FCVID and Kinetics, respectively. We mainly instantiate RNNs with LSTMs, and also compare with different types of recurrent networks. On ActivityNet, the coarse RNN and the fine RNN respectively consists of 64 and 512 hidden units. On FCVID and Kinetics, there are $2,048$ hidden units in the fine RNN; the coarse RNN contains 512 and $1,024$ units, respectively. The computational cost for MobileNetv2, ResNet-101, DPN-107, and SlowFast-8×8 is 0.08 GFLOPs, 7.82 GFLOPs, 18.34 GFLOPs, and 65.71 GFLOPs for a single frame [2], respectively. During testing, we sample videos at 1fps for all datasets.

4.2 Main Results

In this section, we first report results of LiteEval using a standard ResNet-101 as the backbone for fine feature computation, offering decent recognition accuracy with moderate computational cost. We then experiment with state-of-the-art video recognition models with better recognition accuracies but are more compute-demanding.

[2] For SlowFast, a frame indicates a snippet with 8 frames.

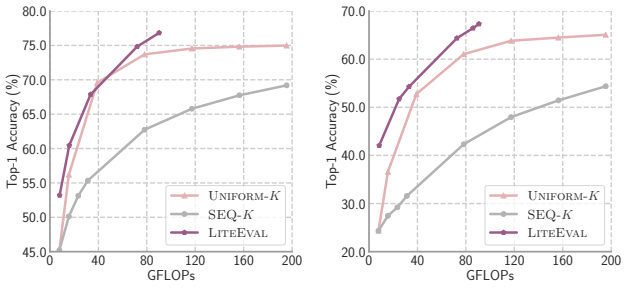
4.2.1 Fine Feature Computation with a ResNet-101

Offline recognition. We first evaluate LiteEval for offline prediction and compare with the following alternative methods: (1) Uniform, which generates prediction scores using the large CNN for 25 uniformly sampled frames and then performs an average-pooling to obtain video-level classification scores; (2) LSTM, which takes in features from the large CNN and produces classification scores using hidden states from the last time step of an LSTM; (3) Uniform+CF, which combines prediction scores from the coarse CNN and the fine CNN model for each frame and then averages 25 frames for prediction; (4) LSTM+CF, which uses concatenated coarse and fine features as inputs to an LSTM to produce predictions; (5) FrameGlimpse (Yeung et al., 2016), which uses an agent trained with REINFORCE (Sutton and Barto, 1998) to choose a small number of frames for efficient recognition; (6) FastForward (Fan et al., 2018), which trains an agent to learn how many steps to jump forward at each time step with REINFORCE (Sutton and Barto, 1998); (7) LiteEval-RL, which is a variant of LiteEval using REINFORCE to learn binary decisions. It is worth pointing out the uniform baseline is very strong and has been adopted by almost all CNN-based approaches due to its simplicity and effectiveness.

Table 1 Results of different approaches for offline video recognition. We compare LiteEval with alternative methods on FCVID and ActivityNet.

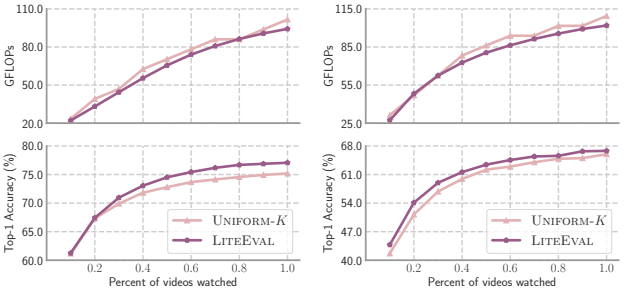
	FCVID		ActivityNet	
Method	mAP	GFLOPs	mAP	GFLOPs
Uniform	80.0%	195.5	70.0%	195.5
LSTM	79.8%	196.0	70.8%	195.8
Uniform+CF	80.1%	197.5	70.1%	197.5
LSTM+CF	80.1%	198.0	71.5%	197.8
FrameGlimpse	71.2%	29.9	60.2%	32.9
FastForward	67.6%	66.2	54.7%	17.2
LiteEval-RL	74.2%	245.9	65.2%	269.3
LiteEval	80.0%	94.3	72.7%	95.1

The results and comparisons of different methods are summarized in Table 1. Compared to the uniform baseline, we observe that LiteEval achieves 51.8% (94.3 *vs.* 195.5) and 51.3% (95.1 *vs.* 195.5) computational savings measured by GFLOPs while offering similar or better accuracies on FCVID and ActivityNet, respectively. This confirms that LiteEval is able to save computation by using low-cost features by default and computing high-cost features as infrequently as possible. We also see that LiteEval uses more computational resources on ActivityNet than on FCVID, which



(a) FCVID (b) ActivityNet

Fig. 2 Computational cost *vs.* recognition accuracy on FCVID and ActivityNet. Results different methods for online prediction.



(a) FCVID (b) ActivityNet

Fig. 3 Percentage of videos watched *vs.* the corresponding computational cost and recognition accuracy on FCVID and ActivityNet for online recognition.

results from the fact that classes in ActivityNet are action-focused whereas FCVID also includes categories that are relatively static with fewer motion. In addition, combining coarse features to fine feature offers slight gains on both datasets. Further, in contrast to FrameGlimpse and FastForward that also learn frame usage policies, LiteEval offers significantly better accuracies although it requires more computation. This is because FrameGlimpse and FastForward have access to future frames (*i.e.*, jumping to a future time step). Instead, LiteEval decides whether to use fine features for the current frame, allowing the framework to be used not only for offline prediction but also in online settings, as will be discussed below. We also compare with LiteEval-RL, which instead of using Gumbel-Softmax learns binary decisions with reinforcement learning. LiteEval achieves better results than LiteEval-RL in terms of both accuracy and computational cost, and it is also easier to optimize.

Online recognition with varying computational budgets. Offline recognition asssumes the whole video is available when making decisions so that multiple snippets/frames can be sampled from the entire video to emit predictions. This is not applicable in online scenerios where frames arrrive sequentially, *i.e.*, the model can only use information seen so far. After LiteEval is trained, it can be readily used for online video recognition. Since the most computationally expensive operation in the framework is to compute fine features, we vary the number of times fine features are computed (represented by K) to accommodate different computational budgets. This forces the model to emit predictions after fine features have been read in for the K-th time, which is conceptually similar to any time prediction (Huang et al., 2018a) that assigns a budget to each testing sample. We report the top-1 recognition accuracy and the mean computational cost on the dataset. We compare with: (1) Uniform-K, which performs a mean-pooling over predictions from K frames that are sampled uniformly from a total of K' frames as the video-level scores. Here, K' is the locaiton where LiteEval emits classification scores after having read in fine features for the K-th time. (2) Seq-K, which averages scores of K consecutive frames.

Figure 2 summarizes the results. From the figure, we see that LiteEval achieves the best trade-off between computational cost and recognition accuracies on both FCVID and ActivityNet for online video recognition. Note that although Uniform-K is a strong baseline, it is not feasible in the online setting since there is no information about the length of the incoming video and the frame rate. In addition, LiteEval achieves better results than the straightforward frame-by-frame computation strategy Seq-K by clear margins. This highlights the effectiveness when LiteEval is deployed online. We also show the computation cost and recognition accuracy with respect to the percentage of videos that are seen in Figure 3. We see that LiteEval achieves better accuracies than the Uniform-K baseline while requiring less computation on both datasets. We also observe more computational resources are used on ActivityNet than FCVID, which is consistent with observations in offline settings.

Learned policies for fine feature usage. We provide analysis on the policies derived by the conditional gating module that dynamically determines whether to compute fine features or not. We visualize, in Figure 4, the distribution of fine feature usage for sampled video classes from FCVID. We observe that the number of times fine features are read in not only varies across different classes but also within the same category. As fine feature computation is a direct indicator of the overall computation, this confirms our hypothesis that computation needed to make correct predictions is different conditioned on input samples. Figure 5 further demonstrates frames selected by LiteEval to compute

Table 2 Results of LiteEval using different backbones on FCVID and ActivityNet. Top: offline recognition. Bottom: online recognition.

		FCVID			ActivityNet		
		mAP	GFLOPs	# Fine Feat	mAP	GFLOPs	# Fine Feat
Offline	DPN-10	81.6%	183.40	10.00 ± 0.00	82.3%	183.40	10.00 ± 0.00
	DPN-25	82.1%	458.50	25.00 ± 0.00	83.0%	458.50	25.00 ± 0.00
	LiteEval	83.3%	157.90	8.00 ± 5.90	84.0%	193.82	9.96 ± 5.20
	SlowFast-10	83.0%	657.10	10.00 ± 0.00	84.2%	657.10	10.00 ± 0.00
	SlowFast-25	83.7%	1642.75	25.00 ± 0.00	85.0%	1642.75	25.00 ± 0.00
	LiteEval	83.9%	430.98	6.39 ± 3.04	85.4%	546.76	8.15 ± 5.42
Online	DPN	74.8%	73.36	4.00 ± 0.00	59.5%	73.36	4.00 ± 0.00
	LiteEval	76.5%	57.78	2.97 ± 0.18	65.7%	55.96	2.99 ± 0.08
	DPN	78.5%	110.04	6.00 ± 0.00	71.8%	110.04	6.00 ± 0.00
	LiteEval	79.4%	90.62	4.64 ± 0.77	71.9%	92.12	4.91 ± 0.37
	DPN	80.6%	165.06	9.00 ± 0.00	80.2%	165.06	9.00 ± 0.00
	LiteEval	81.9%	131.56	6.75 ± 2.51	80.2%	153.39	8.10 ± 2.10

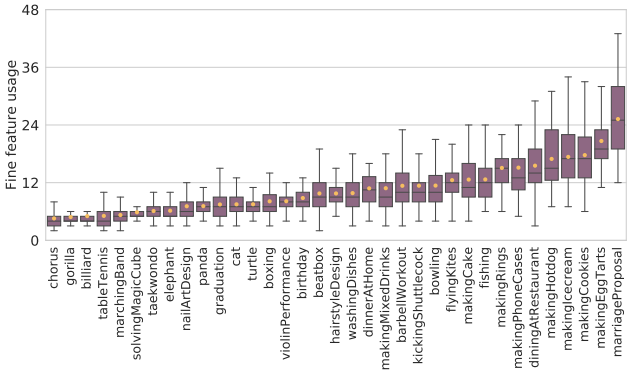


Fig. 4 The distribution of fine feature usage for sampled classes on FCVID. In addition to quartiles and medians, mean usage, denoted as yellow dots, is also presented.

fine features of certain videos. We see that redundant frames without additional information are skipped and those selected frames contain salient information for classifying the class of interest.

4.2.2 Results on State-of-the-Art Video Models

To demonstrate that LiteEval is a generic framework, where different backbones can be readily used. We use (1) a DPN-107 model (Chen et al., 2017) trained with temporal segment networks (Wang et al., 2016) for temporal modeling. During training, it learns to fuse predictions from uniformly sampled frames with a Softmax function; during inference, the DPN-107 model is used similarly as vanilla 2D networks; (2) a SlowFast-8×8 network (Feichtenhofer et al., 2019), which samples 8 frames from a total of 64 frames with a stride of 8 as inputs to 3D CNNs with two pathways. We evaluate both models for offline video prediction; we only evaluate the DPN model for online video prediction, since the SlowFast network takes in stacked frames as inputs that requires a delicate caching design to store frames.

The results are summarized in Table 2. We can see that for offline predictions, LiteEval offers similar or better performance compared to their counterparts requiring less computation for both backbones. More specifically, on FCVID, using the DPN-107 as the backbone, LiteEval offers an mAP of 83.3% which is better than averaging 25 frames by 1.2%, requiring 65% less computation. We observe similar trends with the SlowFast backbone. In addition, similar observations can be made on ActivityNet. For online predictions, we can also see that LiteEval is also better than the uniform baseline with different computational budgets. This confirms that LiteEval is compatible with modern powerful architectures for both online and offline video recognition.

	# Fine feat§	2D/3D	mAP
IDT (Wang and Schmid, 2013)	–	–	68.7%
C3D (Tran et al., 2015)	10×16	3D	67.7%
P3D (Qiu et al., 2016)	20×16	3D	78.9%
RRA (Zhu et al., 2018)	*	2D	83.4%
MARL (Wu et al., 2019a)	25×1	2D	83.8%
LiteEval (DPN)	9.96 ×1	2D	84.0%
LiteEval (SlowFast)	8.15× 8	3D	85.4%

Table 3 State-of-the-art results with different architectures on ActivityNet. *RRA takes in all frames decoded at 4fps; § A×B, represents uniformly sampled A snippets for each video, and there are B frames in each snippet.

Comparisons with state-of-the-art methods on ActivityNet. We now compare with state-of-the-art approaches with powerful backbones for video recognition on ActivityNet. The results are summarized in

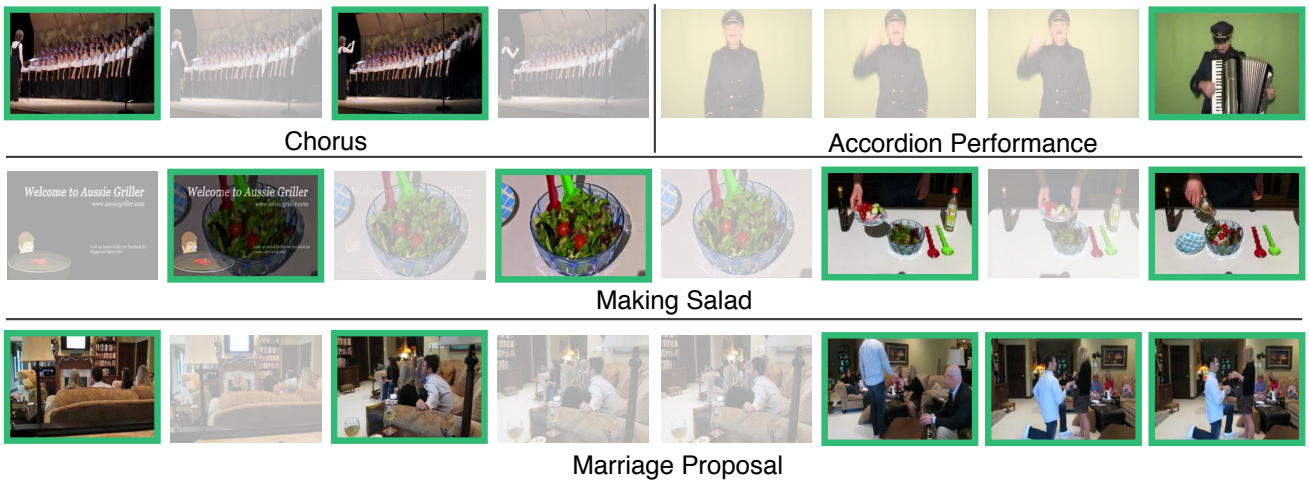


Fig. 5 Frame selected (indicated by green borders) by LiteEval of sampled videos to compute fine features in FCVID.

Table 3. We observe that LiteEval achieves competitive performance compared to state-of-the-art models.

Results on Kinetics with SlowFast. So far we evaluate LiteEval on ActivityNet and FCVID, both of which contain "untrimmed" videos. We also demonstrate that our approach can be used for "trimmed" videos. We report our results on the Kinetics dataset using the SlowFast backbone. During inference, we use a single crop for evaluation and the results are presented in Table 4. We observe that LiteEval achieves similar top-1 accuracy but requires less computation compared to the uniform baseline.

Table 4 Comparisons of different approaches on Kinetics. Here, we use a single center crop for inference.

	# Fine feat	GFLOPs	Top-1 Acc.
SlowFast	3.00 ± 0.00	197.10 ± 0.00	74.3%
SlowFast	5.00 ± 0.00	328.50 ± 0.00	74.9%
SlowFast	10.00 ± 0.00	657.00 ± 0.00	75.2%
LiteEval	4.16 ± 2.20	275.91 ± 144.10	75.0%

Results on Breakfast with Slowfast. We also experimented on the Breakfast dataset (Kuehne et al., 2014) to demonstrate that our apporach is applicable to long videos where long temporal context information is important. The Breakfast dataset (Kuehne et al., 2014) contains 10 human activties about breakfast preparation. There are 1,989 videos in the dataset, collected by 52 persons performing actions in 18 different kitchens. The average length of each video is about 140 seconds. Activities in this dataset are complicated due to large intra-class variations since they contain very similar sub-actions. We adopted the "s1" split to evaluate our approach with a SlowFast model. The results are summarized in Table 5. We can see that LiteEval achieves better performance with fewer computational resources required. This verifies that our approach is general and can be used to model different types of videos. Note that compared to Kinetics, we observe fewer computational savings since the breakfast dataset requires more temporal information.

Table 5 Comparisons of different approaches on Breakfast. Here, we use a single center crop for inference.

	# Fine feat	GFLOPs	Top-1 Acc.
SlowFast	3.00 ± 0.00	197.10 ± 0.00	60.6%
SlowFast	8.00 ± 0.00	525.60 ± 0.00	66.2%
SlowFast	10.00 ± 0.00	657.00 ± 0.00	68.3%
LiteEval	7.35 ± 0.06	497.80 ± 390.10	68.3%

4.3 Discussion

We now provide ablation studies and discussion to justify design choices of different components in the framework.

Different types of recurrent networks. We mainly use LSTMs as an "agent" not only to make classification predictions but also make sequential gating decisions. We also experiment with two types of recurrent networks, *i.e.*, GRU (Cho et al., 2014) and SRU (Lei et al., 2017). We see that SRU and LSTMs are clearly better than GRU, since the capacity of GRU is limited without the cell states. SRU offers better accuracies on FCVID but requires more computation.

Table 6 Results of different types of recurrent networks on ActivityNet.

	DPN		SlowFast	
RNN	mAP	# Fine feat	mAP	# Fine feat
GRU	67.9%	8.78 ± 16.35	73.0%	9.44 ± 16.20
SRU	85.0%	12.48 ± 4.24	84.5%	10.19 ± 3.82
LSTM	84.0%	9.96 ± 5.20	85.4%	8.15 ± 5.42

Fine feature usage. We show, in Table 7, the results of LiteEval when trained with different γ, which controls the fine feature usage in the framework. We see that using a γ of 0.05 achieves the best trade-off between computational savings and accuracies. When an extremely small γ (*e.g.*, 0.01) is set, the results are worse, since it forces the model to compute fine features as infrequently as possible to save computation and could possibly overlook important details. Note that decent results are achieved with smaller γ values (*i.e.*, less or equal than 0.1), which suggests video frames are highly redundant.

Table 7 Results of different γ in LiteEval on FCVID.

γ	mAP	GFLOPs
0.01	78.8%	75.4
0.03	79.7%	82.1
0.10	80.1%	139.0
0.05	80.0%	94.3

The synchronization of the fine RNN with the coarse RNN. We demonstrate the effectiveness of synchronizing the coarse RNN with the fine RNN. We see, from Table 8 that, without updating the hidden states of the `fRNN` with those of the `cRNN`, the performance degrades to 65.7%. This verifies that synchronization by transferring information from the `cRNN` to `fRNN` is important as it provides information from all visual information seen so far.

Table 8 The effectiveness of syncing LSTMs on FCVID.

Method	mAP
w/o. sync	65.7%
LiteEval	80.0%

The number of hidden units in the RNNs. We also investigate the impact of the number of hidden units in the coarse RNN and summarize the results in Table 9. We observe that the performance is worse when using a small LSTM with fewer hidden units due to limited capacity. As mentioned earlier, the most expensive operation in the framework is to extract features from video frames with CNN models, while LSTMs are much more computationally efficient—it only requires 0.06% of computation compared to the extraction of features with a ResNet-101 model. For the fine RNN, we found that using a size of $2,048$ offers the best results.

Table 9 Results of different sizes of LSTMs on FCVID.

The number of units in `cRNN`	mAP
64	76.9%
128	77.3%
256	78.3%
512	80.0%

Runtime. We report the runtime of LiteEval and compare with LSTM, SlowFast and DPN-10. The results are summarized in Table 10. We use a batch size of 16 and then measure runtime by seconds per video on a server with 2 Intel(R) Xeon(R) E5-2658 v4 2.30GHz CPUs and 128G memory using a single NVIDIA TITAN X GPU. We can see that LiteEval is faster compared to alternative approaches.

Method	GFLOPs	Runtime (seconds/video)	mAP
DPN-10	183.4	0.058	81.6
SlowFast-10	657.1	0.047	83.0
LSTM	657.3	0.061	83.1
LiteEval	430.98	0.041	83.9

Table 10 Comparisons between LiteEval with SlowFast-10, DPN-10 and LSTM in terms of mAP, GFLOPs and runtime on FCVID.

5 Conclusion

We introduced LiteEval, a simple yet effective framework for resource-efficient video prediction in both online and offline settings. LiteEval is a conditional computation framework that consists of a coarse RNN and a fine RNN working cooperatively, as well as a gating module. In particular, LiteEval uses low-cost features computed at a coarse scale by default and adaptively determines whether to compute more discriminative features using high resolution inputs to obtain more details. This is achieved by a gating module that can be learned

in a differentiable manner with Gumbel-softmax. The coarse RNN and the fine RNN are also synchronized such that the fine RNN always contain information seen so far and thus can be readily used to make predictions. We conduct extensive experiments on FCVID, ActivityNet and Kinetics for both online and offline video recognition. The results demonstrate the effectiveness of the proposed approach. For future research, we can further prune backbone networks (Wang et al., 2018b; He et al., 2018; Dong and Yang, 2019) to save computation and explore Human Machine Interaction applications (Molchanov et al., 2015; Köpüklü et al., 2019).

References

Bejnordi BE, Blankevoort T, Welling M (2020) Batch-shaping for learning conditional channel gated networks. In: ICLR 4

Bolukbasi T, Wang J, Dekel O, Saligrama V (2017) Adaptive neural networks for fast test-time prediction. In: ICML 4

Chen W, Wilson J, Tyree S, Weinberger K, Chen Y (2015) Compressing neural networks with the hashing trick. In: ICML 1

Chen Y, Li J, Xiao H, Jin X, Yan S, Feng J (2017) Dual path networks. In: NIPS 7, 9

Chen Y, Kalantidis Y, Li J, Yan S, Feng J (2018) Multi-fiber networks for video recognition. In: ECCV 4

Cho K, Van Merriënboer B, Bahdanau D, Bengio Y (2014) On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:14091259 4, 10

Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) Imagenet: A large-scale hierarchical image database. In: CVPR 2

Donahue J, Hendricks LA, Guadarrama S, Rohrbach M, Venugopalan S, Saenko K, Darrell T (2015) Long-term recurrent convolutional networks for visual recognition and description. In: CVPR 3

Dong X, Yang Y (2019) Network pruning via transformable architecture search. In: NeurIPS 12

Fan H, Xu Z, Zhu L, Yan C, Ge J, Yang Y (2018) Watching a small portion could be as good as watching all: Towards efficient video classification. In: IJCAI, DOI 10.24963/ijcai.2018/98 2, 4, 7

Feichtenhofer C (2020) X3d: Expanding architectures for efficient video recognition. In: CVPR 1, 3, 4

Feichtenhofer C, Pinz A, Zisserman A (2016) Convolutional two-stream network fusion for video action recognition. In: CVPR 3

Feichtenhofer C, Fan H, Malik J, He K (2019) Slowfast networks for video recognition. In: ICCV 1, 3, 6, 7, 9

Gao M, Yu R, Li A, Morariu VI, Davis LS (2018) Dynamic zoom-in network for fast object detection in large images. In: CVPR 4

Hazan T, Jaakkola TS (2012) On the partition function and random maximum a-posteriori perturbations. In: ICML 6

He K, Zhang X, Ren S, Sun J (2015) Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: ICCV 1

He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: CVPR 7

He K, Gkioxari G, Dollár P, Girshick R (2017) Mask r-cnn. In: ICCV 1

He Y, Lin J, Liu Z, Wang H, Li LJ, Han S (2018) Amc: Automl for model compression and acceleration on mobile devices. In: ECCV 12

Heilbron FC, Escorcia V, Ghanem B, Niebles JC (2015) Activitynet: A large-scale video benchmark for human activity understanding. In: CVPR 2, 6

Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Computation 4

Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H (2017) Mobilenets: Efficient convolutional neural networks for mobile vision applications. In: CVPR 1, 4

Hu J, Shen L, Sun G (2018) Squeeze-and-excitation networks. In: CVPR 1

Huang G, Chen D, Li T, Wu F, van der Maaten L, Weinberger KQ (2018a) Multi-scale dense convolutional networks for efficient prediction. In: ICLR 8

Huang G, Chen D, Li T, Wu F, van der Maaten L, Weinberger KQ (2018b) Multi-scale dense networks for resource efficient image classification. In: ICLR 4

Iandola FN, Han S, Moskewicz MW, Ashraf K, Dally WJ, Keutzer K (2016) Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size. In: arXiv:1602.07360 1

Jang E, Gu S, Poole B (2017) Categorical reparameterization with gumbel-softmax. In: ICLR 6

Jiang YG, Wu Z, Wang J, Xue X, Chang SF (2018) Exploiting feature and class relationships in video categorization with regularized deep neural networks. IEEE TPAMI 2, 6

Kay W, Carreira J, Simonyan K, Zhang B, Hillier C, Vijayanarasimhan S, Viola F, Green T, Back T, Natsev P, et al. (2017) The kinetics human action video dataset. arXiv preprint arXiv:170506950 2, 3, 6, 7

Köpüklü O, Gunduz A, Kose N, Rigoll G (2019) Real-time hand gesture detection and classification using convolutional neural networks. In: FG 12

Korbar B, Tran D, Torresani L (2019) Scsampler: Sampling salient clips from video for efficient action recognition. In: ICCV 4

Kuehne H, Arslan A, Serre T (2014) The language of actions: Recovering the syntax and semantics of goal-directed human activities. In: CVPR 3, 10

Lei T, Zhang Y, Wang SI, Dai H, Artzi Y (2017) Simple recurrent units for highly parallelizable recurrence. arXiv preprint arXiv:170902755 4, 10

Li H, Kadav A, Durdanovic I, Samet H, Graf HP (2017) Pruning filters for efficient convnets. In: ICLR 1

Li Z, Gavves E, Jain M, Snoek CG (2016) Videolstm convolves, attends and flows for action recognition. arXiv preprint arXiv:160701794 3

Lin J, Rao Y, Lu J, Zhou J (2017) Runtime neural pruning. In: NIPS 4

Lin J, Gan C, Han S (2019) Tsm: Temporal shift module for efficient video understanding. In: ICCV 4

Lin TY, Maire M, Belongie S, Bourdev L, Girshick R, Hays J, Perona P, Ramanan D, Zitnick CL, Dollár P (2014) Microsoft coco: Common objects in context. In: ECCV 2

Liu Z, Miao Z, Zhan X, Wang J, Gong B, Yu SX (2019) Large-scale long-tailed recognition in an open world. In: CVPR 2

Maddison CJ, Mnih A, Teh YW (2017) The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In: ICLR 6

Molchanov P, Gupta S, Kim K, Pulli K (2015) Multi-sensor system for driver's hand-gesture recognition. In: FG 12

Najibi M, Singh B, Davis LS (2019) Autofocus: Efficient multi-scale inference. In: ICCV 4

Ng JYH, Hausknecht M, Vijayanarasimhan S, Vinyals O, Monga R, Toderici G (2015) Beyond short snippets: Deep networks for video classification. In: CVPR 3

Qiu Z, Yao T, Mei T (2016) Deep quantization: Encoding convolutional activations with deep generative model. arXiv preprint arXiv:161109502 9

Qiu Z, Yao T, Mei T (2017) Learning spatio-temporal representation with pseudo-3d residual networks. In: ICCV 3

Rastegari M, Ordonez V, Redmon J, Farhadi A (2016) Xnor-net: Imagenet classification using binary convolutional neural networks. In: ECCV 1

Ren S, He K, Girshick R, Sun J (2015) Faster r-cnn: Towards real-time object detection with region proposal networks. In: NeurIPS 1

Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC (2018) Mobilenetv2: Inverted residuals and linear bottlenecks. In: CVPR 4, 6

Simonyan K, Zisserman A (2014) Two-stream convolutional networks for action recognition in videos. In: NIPS 3

Sutton RS, Barto AG (1998) Reinforcement learning: An introduction. MIT press Cambridge 5, 6, 7

Tran D, Bourdev LD, Fergus R, Torresani L, Paluri M (2015) C3d: Generic features for video analysis. In: ICCV 1, 3, 9

Tran D, Wang H, Torresani L, Ray J, LeCun Y, Paluri M (2018) A closer look at spatiotemporal convolutions for action recognition. In: CVPR 3, 4

Tran D, Wang H, Torresani L, Feiszli M (2019) Video classification with channel-separated convolutional networks. In: ICCV 4

Uzkent B, Ermon S (2020) Learning when and where to zoom with deep reinforcement learning. In: CVPR 4

Veit A, Belongie S (2018) Convolutional networks with adaptive inference graphs. In: ECCV 4

Viola P, Jones MJ (2004) Robust real-time face detection. IJCV 3

Wang H, Schmid C (2013) Action recognition with improved trajectories. In: ICCV 9

Wang L, Xiong Y, Wang Z, Qiao Y, Lin D, Tang X, Van Gool L (2016) Temporal segment networks: Towards good practices for deep action recognition. In: ECCV 1, 3, 7, 9

Wang X, Girshick R, Gupta A, He K (2018a) Non-local neural networks. In: CVPR 1

Wang X, Yu F, Dou ZY, Gonzalez JE (2018b) Skipnet: Learning dynamic routing in convolutional networks. In: ECCV 4, 12

Wu CY, Zaheer M, Hu H, Manmatha R, Smola AJ, Krähenbühl P (2018a) Compressed video action recognition. In: CVPR 4

Wu W, He D, Tan X, Chen S, Wen S (2019a) Multi-agent reinforcement learning based frame sampling for effective untrimmed video recognition. In: ICCV 9

Wu Z, Nagarajan T, Kumar A, Rennie S, Davis LS, Grauman K, Feris R (2018b) Blockdrop: Dynamic inference paths in residual networks. In: CVPR 4

Wu Z, Xiong C, Jiang YG, Davis LS (2019b) Liteeval: A coarse-to-fine framework for resource efficient video recognition. In: NeurIPS 3

Wu Z, Xiong C, Ma CY, Socher R, Davis LS (2019c) Adaframe: Adaptive frame selection for fast video recognition. In: CVPR 4

Xie S, Girshick R, Dollár P, Tu Z, He K (2017) Aggregated residual transformations for deep neural networks. In: CVPR 1

Yang L, Han Y, Chen X, Song S, Dai J, Huang G (2020) Resolution adaptive networks for efficient inference. In: CVPR 4

Yao T, Ngo CW, Zhu S (2012) Predicting domain adaptivity: Redo or recycle? In: ACM Multimedia 7

Yeung S, Russakovsky O, Mori G, Fei-Fei L (2016) End-to-end learning of action detection from frame glimpses in videos. In: CVPR 2, 4, 7

Zhang B, Wang L, Wang Z, Qiao Y, Wang H (2016) Real-time action recognition with enhanced motion vector cnns. In: CVPR 4

Zhou B, Andonian A, Oliva A, Torralba A (2018) Temporal relational reasoning in videos. In: ECCV 4

Zhu C, Tan X, Zhou F, Liu X, Yue K, Ding E, Ma Y (2018) Fine-grained video categorization with redundancy reduction attention. In: ECCV 9

Zolfaghari M, Singh K, Brox T (2018) Eco: Efficient convolutional network for online video understanding. In: ECCV 4