Dynamic Face Video Segmentation via Reinforcement Learning

Yujiang Wang[1,2] **Jie Shen**[1,2] **Mingzhi Dong**[3] **Yang Wu**[4] **Shiyang Cheng**[2] **Maja Pantic**[1,2]
[1] Imperial College London [2]Samsung AI Center Cambridge
[3] University College London [4]Nara Institute of Science and Technology
{yujiang.wang14, jie.shen07}@imperial.ac.uk mingzhi.dong.13@ucl.ac.uk
yangwu@rsc.naist.jp {shiyang.c, maja.pantic}@samsung.com

Abstract

For real-time semantic video segmentation, most recent works utilise a dynamic framework with a key scheduler to make online key/non-key decisions. Some works used a fixed key scheduling policy, while others proposed adaptive key scheduling methods based on heuristic strategies, both of which may lead to suboptimal global performance. To overcome this limitation, we propose to model the online key decision process in dynamic video segmentation as a deep reinforcement learning problem, and to learn an efficient and effective scheduling policy from expert information about decision history and from the process of maximising global return. Moreover, we study the application of dynamic video segmentation on face videos, a field that has not been investigated before. By evaluating on the 300VW dataset, we show that the performance of our reinforcement key scheduler outperforms that of various baseline approaches, and our method could also achieve real-time processing speed. To the best of our knowledge, this is the first work to use reinforcement learning for online key-frame decision in dynamic video segmentation, and also the first work on its application on face videos.

1 Introduction

In computer vision, semantic segmentation is a computationally intensive task which performs per-pixel classification on images. Following the pioneering work of Fully Convolutional Networks (FCN) [1], tremendous progress has been made in recent years with the propositions of various deep segmentation methods [2–12]. To achieve accurate result, these image segmentation models usually employ heavy-weight deep architectures and additional steps such as spatial pyramid pooling [5, 10, 2] and multi-scaled paths of inputs/features [13, 11, 14, 2, 15, 8, 16], which further increase the computational workload. For real-time applications such as autonomous driving, video surveillance, and facial analysis [17], it is impractical to apply such methods on a per-frame basis, which will result in high latency intolerable to those applications. Therefore, acceleration becomes a necessity for these models to be applied in real-time video segmentation.

Various methods [18–26] have been proposed to accelerate video segmentation. Because adjacent frames in a video often share a large proportion of similar pixels, most of these works utilise a dynamic framework which separates frames into key and non-key frames and produce their segmentation masks differently. As illustrated in Fig. 1 (left), a deep image segmentation model $\mathcal{N}$ is divided into a heavy feature extraction part $\mathcal{N}_{feat}$ and a light task-related part $\mathcal{N}_{task}$. To produce segmentation masks, key frames would go through both $\mathcal{N}_{feat}$ and $\mathcal{N}_{task}$, while a fast feature interpolation method is used to obtain features for the non-key frames by warping $\mathcal{N}_{feat}$'s output on the last key frame (LKF), thus to avoid the heavy cost of running $\mathcal{N}_{feat}$ on every frame. On top of that, a key scheduler is used to predict whether an incoming frame should be a key or non-key frame.

Preprint. Under review.

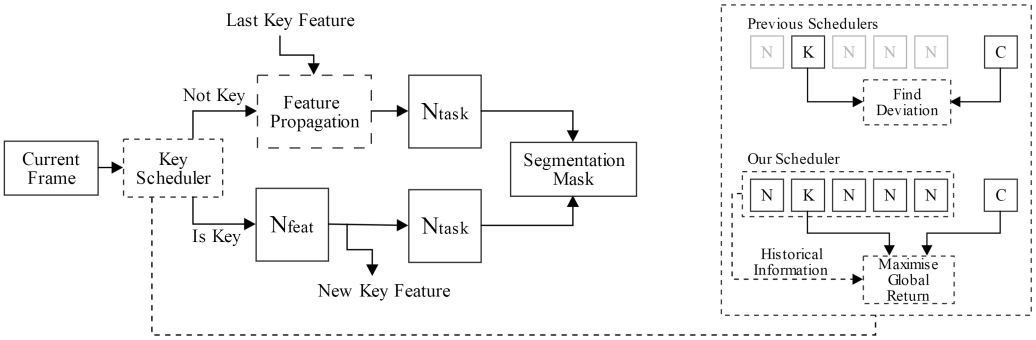

Figure 1: Left: The dynamic video segmentation framework in which a key scheduler is used to make online key/non-key predictions. Right: a comparison between previous key schedulers and ours. Previous works only consider deviation between current frame (C) and the last key frame (K), while our scheduler takes into account C, K and historical information from non-key frames (N), aiming to maximise the global return.

As an essential part of dynamic video segmentation, decisions made by the key scheduler could significantly affect the overall performance [21, 20, 27] of the video segmentation framework. However, this topic is somewhat underexplored by the community. Several recent works [23, 19, 24, 22] simply used a fixed key scheduler, which is usually suboptimal as it does not take into account the video content. Some other works have proposed to use adaptive schedulers [20, 21, 27] that make the key/non-key decisions based on whether the deviation between two frames surpass a certain threshold. Trained to heuristically predict similarity between image pairs, such schedulers lack awareness of the global video context, which may also lead to suboptimal performance in the long run.

To overcome this limitation, we propose to apply reinforcement learning techniques to expose the key scheduler to the global video context. Leveraging additional expert information about decision history, our scheduler is trained to learn key-decision policies that maximise the long-term returns in each episode, as shown in Fig. 1 (right).

We further study the application of dynamic video segmentation in real-time face videos. Comparing to semantic image/video segmentation, semantic segmentation for faces is a less investigated field [28–37], and there are even fewer works on face segmentation in videos [17, 38]. These works either used engineering-based features [30, 31, 33–36], or employed outdated image segmentation models like FCN [1] on a per-frame basis [32, 38, 17] without a dynamic acceleration mechanism. Therefore, we propose real-time face segmentation system based on the dynamic segmentation framework with our key scheduler trained using reinforcement learning.

In particular, we adopt the Deeplab-V3+ model [7] with MobileNet-V2 [39] backbone for the image segmentation model $\mathcal{N}$, and we use optical flows extracted by the FlowNet2-s architecture [40] to interpolate key-frame features to non-key ones [19]. We conduct experiments on the 300 Videos in the Wild (300VW) [41] dataset, where the face segmentation annotations are obtained as mentioned in [17]. By comparing with several baseline approaches, we show that our reinforcement key schedulers can make more effective key-frame decisions at the cost of little resource. We also show that our final system could achieve real-time performances for face segmentation task.

2 Related works

Semantic image segmentation Fully Convolutional Networks (FCN) [1] is the first work to use fully convolutional layers and skip connections to obtain pixel-level predictions for image segmentation. Successive works have made various improvements, including the usage of dilated convolutions [15, 2, 10, 42, 43], encoder-decoder architecture [3, 8, 7], Conditional Random Fields (CRF) for post-processing [44, 15, 2], spatial pyramid pooling to capture multi-scale features [5, 2, 10] and Neural Architecture Search (NAS) [45] to search for the best-performing architectures [46, 6]. Nonetheless, such models usually require intensive computational resources, and applying them in real-time frame-by-frame video segmentation can lead to undesirably high latency. To address this, several

light-weight architectures were proposed [12, 11]. Unfortunately, due to the absence of effective interpolation from previous frames, their solutions could not produce temporal-consistent results.

Dynamic video segmentation Clockwork ConvNet [18] promoted the idea of dynamic segmentation by fixing part of the network to avoid unnecessary computations. Deep Feature Flow (DFF) [19] proposed to accelerate video recognition by leveraging optical flow (extracted by FlowNet [47, 40] or SpyNet [48]) to warp key-frame features. Similar idea can also be found in [20, 24–26]. Jain and Gonzalez [22] proposed to use block motion vectors in compressed videos to achieve fast feature interpolation, however, it is difficult to adapt this method to online scenarios. Mahasseni *et al.*[49] employed a single convolution layer with uniform filters as the interpolation model. Nevertheless, Li *et al.*[21] argued that using such convolution filters does not reflect the varying motion across frames, and thus proposed to use a spatially-variant convolution for propagation. On the other hand, NAS [45] was utilised by [23] to explore the best architecture for the interpolation model.

Although various feature propagation techniques have been explored, research on key scheduling policy is limited. Most existing works adopted fixed key schedulers [23, 19, 24, 22], which is inefficient for real-time segmentation. Mahasseni *et al.*[49] suggested a budget-aware, LSTM-based key selection strategy trained with reinforcement learning, however it is only applicable for *offline* scenarios. Inspired by DFF [19], DVSNet [20] used an adaptive key decision network which takes as input the optical flow features between key-current image pairs, and computes the similarity score between current interpolated segmentation mask (if non-key) and the prediction from image segmentation model $\mathcal{N}$ (if key). If this score is lower than a threshold, it will be a key and vice versa. Li *et al.*[21] introduced a dynamic key scheduler which was trained to predict the deviation degree between two frames by the deviations of their low-level features. Similarly, [27] proposed to adaptively determine key frames based on the number of positions where temporal features are inconsistent. All these schedulers were trained to learn the deviation degree between two frames and lacked the understanding of global context. On the contrary, our key scheduler is assisted by reinforcement techniques to derive a temporal-consistent policy for maximising overall performance.

Semantic face segmentation The study of semantic face segmentation received far less attention than that of image/video segmentation. Early works on this topic were mostly engineering-feature based [30, 31, 33–36] and were designed for static image. Saito *et al.*[38] employed graphic cut algorithm to refine the segmentation probabilistic maps obtained from a FCN trained with augmented data. In [32], a semi-supervised data collection approach was proposed to generate a large number of labelled facial images with random occlusion for FCN training. Recently, Wang *et al.*[17] integrated Conv-LSTM [50] with FCN [1] to extract face masks from video sequence. Despite its improved accuracy against vanilla FCN, its run-time speed did not improve. Furthermore, we can simply replace FCN with other segmentation models [7, 6, 3] to achieve better performance. None of the aforementioned works has considered the video dynamics, and their performances are overshadowed by those of new FCN variants [3, 7, 2, 6]. To bridge this gap, we propose to combine the DFF framework [19] with the advanced Deeplab-V3+ segmentation approach [7] and the FlowNet2-s model [40], and integrate our proposed reinforcement-based key scheduler to devise an effective and efficient real-time face video segmentation framework.

Reinforcement learning In model-free Reinforcement Learning (RL), an agent receives a state $\mathbf{s}_t$ at each time step t from the environment, and learns a policy $\pi_\theta(a_j|\mathbf{s}_t)$ with parameters θ that guides the agent to take an action $a_j \in \mathcal{A}$ to maximise the cumulative rewards $J = \sum_{t=1}^{\infty} \gamma^{t-1} r_t$. RL has demonstrated impressive performance on various fields such as robotics and complicated strategy games [51–55]. In this paper, we show that RL can be seamlessly applied to online key decision problem in real-time video segmentation, which can be seen as a Markov Decision Process (MDP). Among various RL approaches, we chose the policy gradient with reinforcement [56] to learn π_θ, where gradient ascend was used for maximising the objective function $J_\pi(\theta)$.

3 Methodology

3.1 System Overview

Our target is to develop an efficient and effective key scheduling policy $\pi_\theta(\mathbf{a}|\mathbf{s})$ for the dynamic video segmentation system. To this end, a feature propagation framework is essential, thus we adopted the Deep Feature Flow [19] where the optical flow is calculated by a light-weight flow estimation model

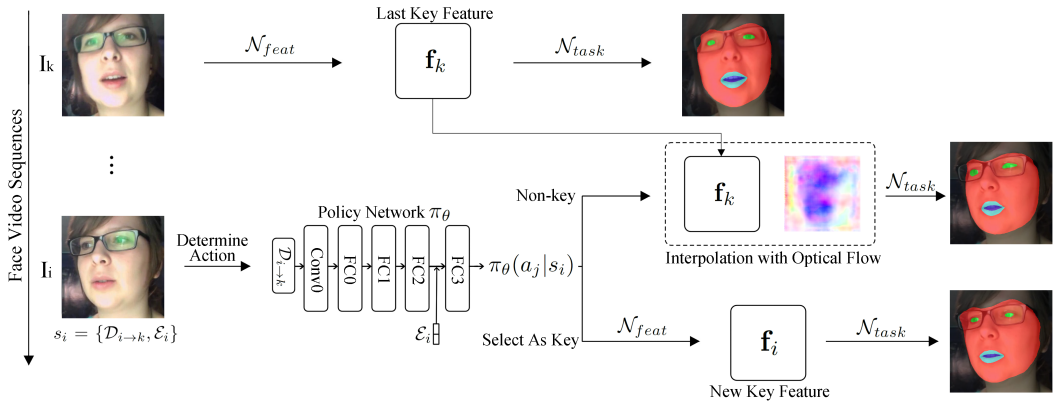

Figure 2: An overview of our system. $\mathbf{I}_k$ is the last key frame (key decision process not shown) with feature $\mathbf{f}_k$ extracted by $\mathcal{N}_{feat}$. For an incoming frame $\mathbf{I}_i$, its input state $\mathbf{s}_i$ includes two components: the deviation information $\mathcal{D}_{i\to k}$ between $\mathbf{I}_i$ and $\mathbf{I}_k$, and the expert information $\mathcal{E}_i$ about decision history. $\mathcal{D}_{i\to k}$ is fed into Conv0 layer of policy network π_θ, while $\mathcal{E}_i$ is concatenated to the output of FC2 layer. Basing on $\mathbf{s}_i$, π_θ gives probabilities output $\pi_\theta(a_j|\mathbf{s}_i)$ regarding taking key or non-key actions. For a non-key action, the optical flow between $\mathbf{I}_i$ and $\mathbf{I}_k$ will be used to warp $\mathbf{f}_k$ to $\mathbf{f}_i$, while for a key action, $\mathbf{I}_i$ will go through $\mathcal{N}_{feat}$ to obtain a new key feature $\mathbf{f}_i$.

$\mathcal{F}$ such as FlowNet [47, 40] or SpyNet [48]. Specifically, an image segmentation model $\mathcal{N}$ can be divided into a time-consuming feature extraction module $\mathcal{N}_{feat}$ and a task specified module $\mathcal{N}_{task}$. We denote the last key frame as $\mathbf{I}_k$ and its features extracted by $\mathcal{N}_{feat}$ as $\mathbf{f}_k$, i.e., $\mathbf{f}_k = \mathcal{N}_{feat}(\mathbf{I}_k)$. For an incoming frame $\mathbf{I}_i$, if it is a key frame, the feature is $\mathbf{f}_i = \mathcal{N}_{feat}(\mathbf{I}_i)$ and the segmentation mask is $\mathbf{y}_i = \mathcal{N}_{task}(\mathbf{f}_i)$; if not, instead of using the resource-intensive module $\mathcal{N}_{feat}$ for feature extraction, its feature $\mathbf{f}_i$ will be propagated by a feature interpolation function $\mathcal{W}$, which involves the flow field $\mathbf{M}_{i\to k}$ from $\mathbf{I}_i$ to $\mathbf{I}_k$, the scale field $\mathbf{S}_{i\to k}$ from $\mathbf{I}_i$ to $\mathbf{I}_k$, and key frame feature $\mathbf{f}_k$, hence the predicted mask becomes $\mathbf{y}_i = \mathcal{N}_{task}(\mathbf{f}_i)$. Please check [19] for more details on the feature propagation process.

On top of the DFF framework, we design a light-weight policy network π_θ to make online key predictions. The state $\mathbf{s}_i$ at frame $\mathbf{I}_i$ consists of two parts, the deviation information $\mathcal{D}_{i\to k}$ which describes the differences between $\mathbf{I}_k$ and $\mathbf{I}_i$, and the expert information $\mathcal{E}_i$ regarding key decision history (see Section 3.2 for details), i.e., $\mathbf{s}_i = \{\mathcal{D}_{i\to k}, \mathcal{E}_i\}$. Feeding $\mathbf{s}_i$ as input, the policy network outputs the action probabilities $\pi_\theta(a_j|\mathbf{s}_i)$ where $a_j \in \{a_0, a_1\}$ and $\pi_\theta(a_0|\mathbf{s}_i) + \pi_\theta(a_1|\mathbf{s}_i) = 1.0$ (we define a_0 for non-key action and a_1 for the key one). For an incoming frame $\mathbf{I}_i$, if $\pi_\theta(a_1|\mathbf{s}_t) > \tau$ where τ is a threshold, it will be identified as a key frame, vice versa. In general, key action a_1 will lead to a segmentation mask with better quality than the ones given by action a_0.

In this work, we utilise the FlowNet2-s model [40] as the optical flow estimation function $\mathcal{F}$. DVSNet [20] has shown that the high-level features from FlowNet models contain sufficient information about the deviations between two frames, and it can also be easily fetched along with optical flow without additional cost. Therefore, we adopt the features of FlowNet2-s model for $\mathcal{D}_{i\to k}$. It is worthwhile to notice that by varying $\mathcal{D}_{i\to k}$ properly, our key scheduler can be easily integrated into other dynamic segmentation frameworks [24, 21, 23, 22, 27] which do not use optical flow. Fig. 2 gives an overview of our system.

3.2 Training Policy Network

Network structure Our policy network comprises of one convolution layer and four fully connected (FC) layers. The FlowNet2-s feature $\mathcal{D}_{i\to k}$ is fed into the first convolution layer Conv0 with 96 channels, followed by FC layers (FC0, FC1 and FC2) with output size being 1024, 1024 and 128 respectively. Two additional channels containing expert information about decision history $\mathcal{E}_i$ are concatenated to the output of FC2 layer. The first channel records the Key All Ratio (KAR), which is the ratio between the key frame and every other frames in decision history, while the second channel contains the Last Key Distance (LKD), which is the interval between the current and the last key frame. KAR provides information on the frequency of historical key selection, and LKD gives

awareness about the length of continuous non-key decisions. Hence, the insertion of KAR and LKD extends the output dimension of FC2 to 130, while FC3 layer summarises all these information and gives action probabilities $\pi_\theta(a_j|\mathbf{s}_i)$ where $a_j \in \{a_0, a_1\}$, a_0 and a_1 stand for non-key and key action correspondingly.

Reward definition We use the widely-used mean Intersection-over-Union (mIoU) as the metric to evaluate the segmentation masks. We denote the mIoU of $\mathbf{y}_i$ from a non-key action a_0 as $U^i_{a_0}$, the mIoU from key action a_1 as $U^i_{a_1}$, and the reward r_i at frame $\mathbf{I}_i$ is defined in Eq. 1. Such definition encourages the scheduler to choose key action on the frames that would result in larger improvement over non-key action, this also reduces the variances of mIoUs across the video.

$$r_i = \begin{cases} 0, & a_j = a_0. \\ U^i_{a_1} - U^i_{a_0}, & a_j = a_1. \end{cases} \tag{1}$$

Constraining key selection frequency The constraints on key selection frequency are necessary in our task. Since a key action will generally lead to a better reward than a non-key one, the policy network inclines to make all-key decisions if no constraint is imposed on the frequency of key selection. In this paper, we propose a *stop immediately exceeding the limitation* approach. Particularly, for one episode consisting of $M+1$ frames $\{\mathbf{I}_t, \mathbf{I}_{t+1}, ..., \mathbf{I}_{t+M}\}$, the agent starts from $\mathbf{I}_t$ and explores continuously towards $\mathbf{I}_{t+M}$. At each time step, if the KAR in decision history has already surpassed a limit η, the agent will stop immediately and thus this episode ends, otherwise, it will continue until reaching the last frame $\mathbf{I}_{t+M}$. By using this strategy, a policy network should limit the use of key decision to avoid an over-early stopping, and also learn to allocate the limited key budgets on the frames with higher rewards. By varying the KAR limit η, we could train π_θ with different key decision frequencies.

Episode settings Real-time videos usually contains enormous number of high-dimensional frames, thus it is impractical to include all of them in one episode, due to the high computational complexity and possible huge variations across frames. For simplicity, we limit the length of one episode $\{\mathbf{I}_t, \mathbf{I}_{t+1}, ..., \mathbf{I}_{t+M}\}$ to 270 frames (9 seconds in our dataset), which should cover the most circumstances. We vary the starting frame $\mathbf{I}_t$ during training to learn the global policies across videos. For each episode, we let the agent run K times (with the aforementioned key constraint strategy) to obtain K trials to reduce variances. The return of each episode can be expressed as $J(\theta) = \frac{1}{K}\sum_{v=1}^{K}\sum_{u=t}^{t+p_v}\gamma^{u-t}r^v_u$, where t is the starting frame index of the episode, and p_v denotes the total step number at the v^{th} trail (since agent may stop before M steps), and r^v_u refers to the reward of frame u in v^{th} trail. $J(\theta)$ is also the main objective function to optimise.

Auxiliary losses In addition to optimise the cumulative reward $J(\theta)$, we apply two auxiliary losses for regularisation. Following the works of [57, 58], we employ the entropy loss $\mathcal{H}(\pi_\theta(\mathbf{a}|\mathbf{s}))$ to promote the policy that retains high-entropy action posteriors so as to avoid over-confident actions. We also add a L2-norm loss $\mathcal{L}_2(\theta_W)$ for weight decay. Eq. 2 shows the final objective function $\mathcal{L}$ to optimise using policy gradient with reinforcement method [56].

$$\mathcal{L} = J(\theta) + \lambda_1\mathcal{H}(\pi_\theta(\mathbf{a}|\mathbf{s})) - \lambda_2\mathcal{L}_2(\theta_W) \tag{2}$$

Epsilon-greedy strategy During training, agent may still fall into over-deterministic dilemmas with action posteriors approaching nearly 1, even though the auxiliary entropy loss have been added. To recover from such dilemma, we implement a simple strategy similar to epsilon-greedy algorithm for action sampling, i.e., in the cases that action probabilities $\pi_\theta(a_j|\mathbf{s})$ exceed a threshold ϵ (such as 0.98), instead of taking action a_j with probability $\pi_\theta(a_j|\mathbf{s})$, we use ϵ to stochastically pick action a_j (and $1.0 - \epsilon$ for picking action a_{1-j}).

4 Experiments

4.1 Dataset

We have conducted our experiments on the 300 Videos on the Wild (300VW) dataset [41]. This dataset contains 114 videos (captured at 30 FPS) with an average length of 64 seconds, all of which

are taken in unconstrained environment. Following [17], we have cropped faces out of the video frames and generated the segmentation labels with facial skin, eyes, outer mouth and inner mouth for all the 218,595 frames. For experiment purpose, we divided the videos into three subject-independent parts, namely $A/B/C$ sets with 93 / 9 / 12 videos. For training preliminary networks such as the image segmentation model $\mathcal{N}$ and the flow estimation model $\mathcal{F}$, we used sets A, B and C for training, validation and testing purposes respectively, while for the training of key scheduler, we used set B for training and validation, and retained set C for testing. In detail, for training $\mathcal{N}$, we randomly picked 18,570 / 1,740 / 2,400 frames from sets $A/B/C$ for training/validation/testing purposes. As for the training of flow estimation model $\mathcal{F}$, we randomly generate 51,480 / 4,836 / 6,671 *key-current* image pairs with a varying gap between 1 to 30 frames from sets $A/B/C$ for training/validation/testing purposes. We intentionally excluded set A for policy network learning, since this set has already been used to train $\mathcal{N}$ and $\mathcal{F}$, instead, we used the full B set (16,568 frames) for training and validating RL model, and evaluated it on the full C set (22,580 frames).

4.2 Experimental Setup

Evaluation metric We employed the commonly used mean Intersection-over-Union (mIoU) as the evaluation metric. It is worth mentioning that we excluded the background class from mIoU calculation for a fairer evaluation, i.e., the mIoU metric averages the IoUs of facial skin, eyes, outer and inner mouths without the background class, which may lead to a comparatively low mIoU value than including background.

Training preliminary networks To enable a key scheduler to work in a Deep Feature Flow framework, preliminary networks such as the image segmentation model $\mathcal{N}$ and the flow estimation function $\mathcal{F}$ are required. We borrowed the state-of-the-art Deeplab-V3+ architecture [7] for model $\mathcal{N}$, and we selected MobileNet-V2 [39] as its backbone structure considering the balance between performance and running speed. Regarding the implementation of flow estimation function $\mathcal{F}$, we adopted the FlowNet2-s architecture [40]. For training $\mathcal{N}$, we initialised the weights using the pre-trained model provided in [7] and fine-tuned the model. We set the output stride and decoder output stride to 16 and 4, respectively. We divided $\mathcal{N}$ into $\mathcal{N}_{feat}$ and $\mathcal{N}_{task}$, where the output of $\mathcal{N}_{feat}$ is the posterior for each image pixel, we then fine-tuned the FlowNet2-s model $\mathcal{F}$ as suggested in [19] by freezing $\mathcal{N}_{feat}$ and $\mathcal{N}_{task}$. Also, we use the pre-trained weights provided in [40] as the starting point of training $\mathcal{F}$. The input sizes for $\mathcal{N}$ and $\mathcal{F}$ are both set to 513*513.

RL settings For state $\mathbf{s}_i = \{\mathcal{D}_{i\to k}, \mathcal{E}_i\}$, following DVSNet [20], we leveraged the features from the Conv6 layer of the FlowNet2-s model as the deviation information $\mathcal{D}_{i\to k}$, and we obtained the expert information $\mathcal{E}_i = \{KAR, LKD\}$ from the last 90 decisions. During the training of policy network, $\mathcal{N}_{feat}$, $\mathcal{N}_{task}$ and $\mathcal{F}$ were frozen to avoid unnecessary computations. We chose RMSProp [59] as the optimiser and set the initial learning rate to 0.001. The parameters λ_1, λ_2 in Eq. 2 were set to 0.14 and 0.001 respectively. We empirically decided the discount factor γ to be 1.0, as the per frame performance was equally important in our task. The value of epsilon ϵ in epsilon-greedy strategy was set to 0.98. During training, we set the threshold value τ for determining the key action to 0.5.

The maximum length of each episode was set to 270 frames (9 seconds), and we repeated a relatively large number of 32 trials for each episode, while the returns of these trials were concatenated with a normalisation step to stabilise gradients. A mini-batch size of 8 was used for back-propagation in π_θ. Starting from random weights, we trained each model for 2,400 episodes and validated the performances of checkpoints on the same set to find the best one, which was further evaluated on the test set. We also varied the KAR limit η to obtain policy networks with different key decision tendencies.

Baseline comparison For a fair comparison, we applied the same $\mathcal{N}_{feat}$, $\mathcal{N}_{task}$ and $\mathcal{F}$ models for all approaches and only evaluated the key scheduling part. Besides, we used the plots of average key intervals versus mIoUs to measure the overall performance.

We compared our reinforcement key scheduler with three different baseline approaches: (1) The adaptive key decision model in DVSNet [20]; (2) The adaptive key scheduler using flow magnitude difference in [20]; (3) Deep Feature Flow with fixed key scheduler [19]. For implementation of DVSNet, we used the codes provided by the authors, and trained it with 32,049 image pairs generated from set B. By varying the key score threshold, we obtained its performance curve. Regarding the computation of flow magnitude, we refer the readers to [20]. Note that we changed its difference

Table 1: The performance of various image segmentation models and FowNet2-s. mIoU does not include the background class. FPS is evaluated on a NVidia 1080 Ti GPU for one face image of 513*513 size.

Models	Methods	mIoU(%)	FPS
FCN (VGG16)	Per Frame	55.71	45.5
Deeplab-V2 (VGG16)	Per Frame	58.66	3.44
Deeplab-V3+ (Xception-65)	Per Frame	60.69	24.4
Deeplab-V3+ (MobileNet-V2)	Per Frame	60.1	58.8
FlowNet2-s ($\mathcal{N}_{feat}$: Deeplab-V3+ with MobileNet-V2)	Feature Propagation	56.56	153.8

threshold to draw the results. For DFF with fixed scheduler, we simply changed the key interval value to get different results. To obtain the final results of our method, we trained six models with η value set to 0.04, 0.06, 0.1, 0.12, 0.14 and 0.25, respectively. We then varied the threshold τ to obtain more data points for drawing the performance curve.

Implementation We implemented our method in Tensorflow [60] framework. Experiments were run on a cluster with eight NVidia 1080 Ti GPUs, and we evaluated the running speed on a desktop with one NVidia 1080 Ti GPU. It took approximately 2.5 days to train one model with RL.

4.3 Results

Preliminary networks We evaluated four image segmentation models: FCN [1] with VGG16 [61] architecture, Deeplab-V2 [2] of VGG16 version, Deeplab-V3+ [7] with Xception-65 [62] backbone, and Deeplab-V3+ with MobileNet-V2 [39] backbone. As can be seen from Table 1, Deeplab-V3+ with MobileNet-V2 backbone struck a better balance between the speed (58.8 FPS) and the accuracy (60.1% in mIoU), therefore we selected it for our segmentation model $\mathcal{N}$. Its feature extraction part $\mathcal{N}_{feat}$ was used to extract key frame feature in *key-current* images pairs during the training of FlowNet2-s [40] model $\mathcal{F}$, whose performance was evaluated by the interpolation results on current frames. From Table 1 we can discover that the interpolation speed with $\mathcal{F}$ is generally much faster than those segmentation models at the cost of a slight drop in mIoU (from 60.1% to 56.56%). Under live video scenario, the loss of accuracy can be effectively remedied by a good key scheduler.

RL training visualisation In the upper row of Fig. 3, we demonstrate the average return during RL training with different KAR limits η (0.04, 0.06, 0.14). It can be seen that even though we select the starting frames of each episode randomly, those return curves still exhibit a generally increasing trend despite several fluctuations. This validates the effectiveness of our solutions for reducing variances and stabilising gradients, and it also verifies that the policy π_θ is improving towards more rewarding key actions. Besides, as the value of η increases and allows for more key actions, the maximum return that each curve achieves also becomes intuitively higher.

We also visualised the influences of two expert information KAR and LDK by plotting their weights in π_θ during RL training. As shown in the bottom row of Fig. 3, we plotted the weights of two channels in π_θ that received KAR and LDK as input and contributed to the key posteriors $\pi_\theta(a_1|\mathbf{s})$. From these plots we can observe that the weights of the LDK channel show a globally rising trend, while that of the KAR channel decrease continuously. These trends indicate that the information of KAR and LDK become increasingly important in key action decisions as training proceeds, since a large LDK value (or a small KAR) will encourage the scheduler to take key action. This is an intuitive result with the key constraint strategy we have applied. Furthermore, we can imply from the plots that the key scheduler tends to rely more on LDK channel than KAR channel to make key decision with a lower η like 0.04, conversely, KAR becomes more significant with a higher η value such as 0.14.

Performance evaluation The results of our key scheduler with RL and three baseline schedulers are shown in Fig. 4. We can see that the performances of all methods are similar for key intervals less than 20. This is to be expected as the performance degradation on non-key frames can be compensated by dense key selections. Our method starts to show superior performance when the key interval increases beyond 25, where our mIoUs are consistently higher than that of other methods and decreases slower as the key interval increases. It should be noted that, in the case of face videos,

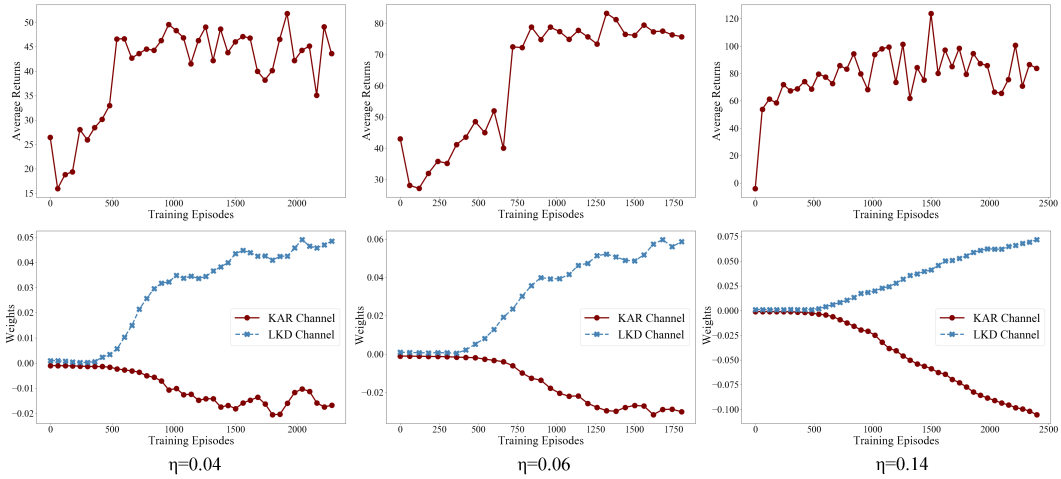

Figure 3: The upper row plots the average return curves during RL training with η value set to 0.04, 0.06 and 0.14. The bottom row illustrates the variations of the weights of KAR and LDK channels contributing to the key posteriors $\pi_\theta(a_1|\mathbf{s})$. The plots in the same column are from the same training session.

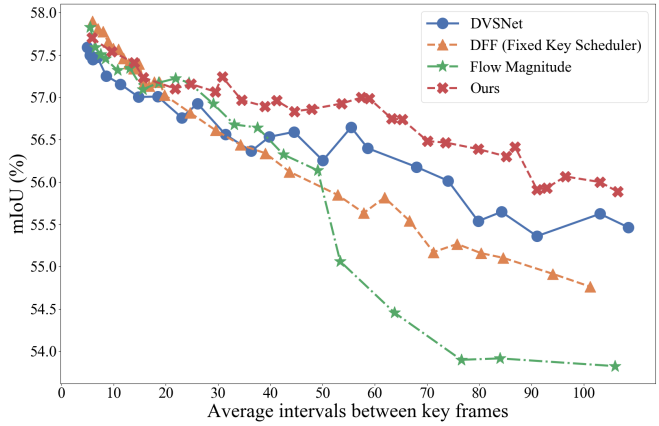

Figure 4: Comparison between ours and baseline key schedulers. No background class in mIoU.

selecting key frames by a small interval ($<< 20$) does not significantly affect the performance, which is not the same as in autonomous driving scenarios [19, 20, 24]. This could be attributed to the fact that variations between consecutive frames in face videos are generally less than those in autonomous driving. As a result, we can gain more efficiency benefit when using key scheduling policy with relatively large interval for dynamic segmentation of face video.

Running speed The average run-time of our key scheduler is 1.1 ms per frame. On average, segmentation takes 7.6 ms and 23.6 ms for non-key and key frame, respectively. With 10% / 20% / 30% key frames, the average frame rate of our system is 108 / 93 / 81 FPS, respectively, which are all faster than real-time as well as the speed of the original $\mathcal{N}$ (58.8 FPS).

5 Conclusions

In this paper, we propose to learn an efficient and effective key scheduler via reinforcement learning for dynamic face video segmentation. By the utilisation of expert information and appropriately-designed training strategies, our key scheduler achieves more effective key decisions than baseline methods on most average key interval sections. This is the first work to apply dynamic segmentation techniques with RL on real-time face videos, and it can be enlightening to future works on real-time face segmentation and on dynamic video segmentation.

References

[1] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.

[2] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2018.

[3] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[4] Z. Wu, C. Shen, and A. Van Den Hengel, "Wider or deeper: Revisiting the resnet model for visual recognition," *Pattern Recognition*, vol. 90, pp. 119–133, 2019.

[5] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2881–2890, 2017.

[6] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. Yuille, and L. Fei-Fei, "Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation," in *CVPR*, 2019.

[7] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *ECCV*, 2018.

[8] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1925–1934, 2017.

[9] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," *arXiv preprint arXiv:1606.02147*, 2016.

[10] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[11] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "Icnet for real-time semantic segmentation on high-resolution images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 405–420, 2018.

[12] V. Nekrasov, C. Shen, and I. Reid, "Light-weight refinenet for real-time semantic segmentation," *arXiv preprint arXiv:1810.03272*, 2018.

[13] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille, "Attention to scale: Scale-aware semantic image segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3640–3649, 2016.

[14] G. Lin, C. Shen, A. Van Den Hengel, and I. Reid, "Efficient piecewise training of deep structured models for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3194–3203, 2016.

[15] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *arXiv preprint arXiv:1412.7062*, 2014.

[16] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe, "Full-resolution residual networks for semantic segmentation in street scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4151–4160, 2017.

[17] Y. Wang, B. Luo, J. Shen, and M. Pantic, "Face mask extraction in video sequence," *International Journal of Computer Vision*, pp. 1–17, 2018.

[18] E. Shelhamer, K. Rakelly, J. Hoffman, and T. Darrell, "Clockwork convnets for video semantic segmentation," in *Computer Vision–ECCV 2016 Workshops*, pp. 852–868, Springer, 2016.

[19] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei, "Deep feature flow for video recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2349–2358, 2017.

[20] Y.-S. Xu, T.-J. Fu, H.-K. Yang, and C.-Y. Lee, "Dynamic video segmentation network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6556–6565, 2018.

[21] Y. Li, J. Shi, and D. Lin, "Low-latency video semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5997–6005, 2018.

[22] S. Jain and J. E. Gonzalez, "Inter-bmv: Interpolation with block motion vectors for fast semantic segmentation on video," *arXiv preprint arXiv:1810.04047*, 2018.

[23] V. Nekrasov, H. Chen, C. Shen, and I. Reid, "Architecture search of dynamic cells for semantic video segmentation," *arXiv preprint arXiv:1904.02371*, 2019.

[24] S. Jain, X. Wang, and J. Gonzalez, "Accel: A corrective fusion network for efficient semantic segmentation on video," *arXiv preprint arXiv:1807.06667*, 2018.

[25] D. Nilsson and C. Sminchisescu, "Semantic video segmentation by gated recurrent flow propagation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6819–6828, 2018.

[26] R. Gadde, V. Jampani, and P. V. Gehler, "Semantic video cnns through representation warping," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4453–4462, 2017.

[27] X. Zhu, J. Dai, L. Yuan, and Y. Wei, "Towards high performance video object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7210–7218, 2018.

[28] U. Güçlü, Y. Güçlütürk, M. Madadi, S. Escalera, X. Baró, J. González, R. van Lier, and M. A. van Gerven, "End-to-end semantic face segmentation with conditional random fields as convolutional, recurrent and adversarial networks," *arXiv preprint arXiv:1703.03305*, 2017.

[29] Y. Zhou, X. Hu, and B. Zhang, "Interlinked convolutional neural networks for face parsing," in *International Symposium on Neural Networks*, pp. 222–231, Springer, 2015.

[30] A. Kae, K. Sohn, H. Lee, and E. Learned-Miller, "Augmenting crfs with boltzmann machine shape priors for image labeling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2019–2026, 2013.

[31] B. M. Smith, L. Zhang, J. Brandt, Z. Lin, and J. Yang, "Exemplar-based face parsing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3484–3491, 2013.

[32] Y. Nirkin, I. Masi, A. T. Tuan, T. Hassner, and G. Medioni, "On face segmentation, face swapping, and face perception," in *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, pp. 98–105, IEEE, 2018.

[33] J. Warrell and S. J. Prince, "Labelfaces: Parsing facial features by multiclass labeling with an epitome prior," in *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pp. 2481–2484, IEEE, 2009.

[34] C. Scheffler and J.-M. Odobez, "Joint adaptive colour modelling and skin, hair and clothing segmentation using coherent probabilistic index maps," in *British Machine Vision Association-British Machine Vision Conference*, no. EPFL-CONF-192633, 2011.

[35] Y. Yacoob and L. S. Davis, "Detection and analysis of hair," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 7, pp. 1164–1169, 2006.

[36] K.-c. Lee, D. Anguelov, B. Sumengen, and S. B. Gokturk, "Markov random field models for hair and face segmentation," in *Automatic Face & Gesture Recognition, 2008. FG'08. 8th IEEE International Conference on*, pp. 1–6, IEEE, 2008.

[37] G. Ghiasi, C. C. Fowlkes, and C. Irvine, "Using segmentation to predict the absence of occluded parts.," in *BMVC*, pp. 22–1, 2015.

[38] S. Saito, T. Li, and H. Li, "Real-time facial segmentation and performance capture from rgb input," in *European Conference on Computer Vision*, pp. 244–261, Springer, 2016.

[39] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *CVPR*, 2018.

[40] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2462–2470, 2017.

[41] J. Shen, S. Zafeiriou, G. G. Chrysos, J. Kossaifi, G. Tzimiropoulos, and M. Pantic, "The first facial landmark tracking in-the-wild challenge: Benchmark and results," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 50–58, 2015.

[42] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[43] F. Yu, V. Koltun, and T. Funkhouser, "Dilated residual networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 472–480, 2017.

[44] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1529–1537, 2015.

[45] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[46] L.-C. Chen, M. D. Collins, Y. Zhu, G. Papandreou, B. Zoph, F. Schroff, H. Adam, and J. Shlens, "Searching for efficient multi-scale architectures for dense image prediction," in *NIPS*, 2018.

[47] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei, "Flow-guided feature aggregation for video object detection," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 408–417, 2017.

[48] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4161–4170, 2017.

[49] B. Mahasseni, S. Todorovic, and A. Fern, "Budget-aware deep semantic video segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1029–1038, 2017.

[50] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, pp. 802–810, 2015.

[51] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[52] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[53] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[54] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser, *et al.*, "Starcraft ii: A new challenge for reinforcement learning," *arXiv preprint arXiv:1708.04782*, 2017.

[55] V. d. N. Silva and L. Chaimowicz, "Moba: a new arena for game ai," *arXiv preprint arXiv:1705.10443*, 2017.

[56] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[57] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, pp. 1928–1937, 2016.

[58] K. Pang, M. Dong, Y. Wu, and T. Hospedales, "Meta-learning transferable active learning policies by deep reinforcement learning," *arXiv preprint arXiv:1806.04798*, 2018.

[59] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[60] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[61] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[62] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.