# Optimization Planning for 3D ConvNets

Zhaofan Qiu [1] Ting Yao [1] Chong-Wah Ngo [2] Tao Mei [1]

## Abstract

It is not trivial to optimally learn a 3D Convolutional Neural Networks (3D ConvNets) due to high complexity and various options of the training scheme. The most common hand-tuning process starts from learning 3D ConvNets using short video clips and then is followed by learning long-term temporal dependency using lengthy clips, while gradually decaying the learning rate from high to low as training progresses. The fact that such process comes along with several heuristic settings motivates the study to seek an optimal "path" to automate the entire training. In this paper, we decompose the path into a series of training "states" and specify the hyper-parameters, e.g., learning rate and the length of input clips, in each state. The estimation of the knee point on the performance-epoch curve triggers the transition from one state to another. We perform dynamic programming over all the candidate states to plan the optimal permutation of states, i.e., optimization path. Furthermore, we devise a new 3D ConvNets with a unique design of dual-head classifier to improve spatial and temporal discrimination. Extensive experiments on seven public video recognition benchmarks demonstrate the advantages of our proposal. With the optimization planning, our 3D ConvNets achieves superior results when comparing to the state-of-the-art recognition methods. More remarkably, we obtain the top-1 accuracy of 80.5% and 82.7% on Kinetics-400 and Kinetics-600 datasets, respectively.

## 1. Introduction

The recent advances in 3D Convolutional Neural Networks (3D ConvNets) have successfully pushed the limits and improved the state-of-the-art of video recognition. For instance, an ensemble of LGD-3D networks (Qiu et al., 2019) achieves 17.88% in terms of average error in trimmed video classification task of ActivityNet Challenge 2019, which is dramatically lower than the error (29.3%) attained by the former I3D networks (Carreira & Zisserman, 2017). The result basically indicates the advantage and great potential of 3D ConvNets for improving video recognition. Despite these impressive progresses, learning effective 3D ConvNets for video recognition remains challenging, due to large variations and complexities of video content. Existing works of 3D ConvNets (Tran et al., 2015; Carreira & Zisserman, 2017; Tran et al., 2018; Wang et al., 2018c; Qiu et al., 2017b; 2019; Feichtenhofer et al., 2019; Feichtenhofer, 2020; Li et al., 2020b; 2021; Qiu et al., 2021) predominately focus on the designs of network architectures but seldom explore how to train a 3D ConvNets in a principled way.

The difficulty in training 3D ConvNets originates from the high flexibility of the training scheme. Compared to the training of 2D ConvNets (Ge et al., 2019; Lang et al., 2019; Yaida, 2019), the involvement of temporal dimension in 3D ConvNets brings two new problems of *how many frames should be sampled from the video* and *how to sample these frames*. First, the length of video clip is a tradeoff to control the balance between training efficiency and long-range temporal modeling for learning 3D ConvNets. On one hand, training with short clips (16 frames) (Tran et al., 2015; Qiu et al., 2017b) generally leads to fast convergence with large mini-batch, and also alleviates the overfitting problem through data augmentation brought by sampling short clips. On the other hand, recent works (Varol et al., 2018; Wang et al., 2018c; Qiu et al., 2019) have proven better ability in capturing long-range dependency when training with long clips (over 100 frames) at the expense of training time. The second issue is the sampling strategy. Uniform sampling (Fan et al., 2019; Jiang et al., 2019; Martínez et al., 2019) offers the network a fast-forward overview of the entire video, while consecutive sampling (Tran et al., 2015; Qiu et al., 2017b; Varol et al., 2018; Wang et al., 2018c) encodes the continuous changes across frames and captures the spatio-temporal relation better. Given these complex choices of training scheme, learning a powerful 3D ConvNets often requires significant engineering efforts of human experts to determine the optimal strategy on each dataset. That motivates us to automate training strategy for 3D ConvNets.

[1]JD AI Research, Beijing, China [2]School of Computing and Information Systems, Singapore Management University, Singapore. Correspondence to: Ting Yao <tingyao.ustc@gmail.com>.

*Proceedings of the $38^{th}$ International Conference on Machine Learning*, PMLR 139, 2021. Copyright 2021 by the author(s).

In the paper, we propose optimization planning mechanism which seeks the optimal training strategy of 3D ConvNets adaptively. To this end, our optimization planning studies three problems: *1) choose between consecutive or uniform sampling; 2) when to increase the length of input clip; 3) when to decrease the learning rate.* Specifically, we decompose the training process into several training states. Each state is assigned with the fixed hyper-parameters, including sampling strategy, length of input clip and learning rate. The transition between states represents the change of hyper-parameters during training. Therefore, the training process can be decided by the permutation of different states and the number of epochs for each state. Here, we build a candidate transition graph to define the valid transitions between states. The search of the best optimization strategy is then equivalent to seeking an optimal path from the initial state to the final state on the graph, which can be solved by dynamic programming. In order to determine the best epoch for each state in such process, we propose a knee point estimation method via fitting the performance-epoch curve. In general, our optimization planning is viewed as a training scheme controller and is readily applicable to train other neural networks in stages with multiple hyper-parameters.

To the best of our knowledge, our work is the first to address the issue of optimization planning for 3D ConvNets training. The issue also leads to the elegant view of how the order and epochs for different hyper-parameters should be planned adaptively. We uniquely formulate the problem as seeking an optimal training path and devise a new 3D ConvNets with dual-head classifier. Extensive experiments on seven datasets demonstrate the effectiveness of our proposal, and with optimization planning, our 3D ConvNets achieves superior results than several state-of-the-art techniques.

## 2. Related Work

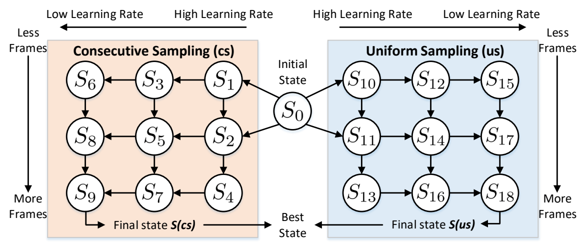
The early works using Convolutional Neural Networks for video recognition are mostly extended from 2D ConvNets for image classification (Karpathy et al., 2014; Simonyan & Zisserman, 2014; Feichtenhofer et al., 2016; Wang et al., 2016; Qiu et al., 2017a). These approaches often treat a video as a sequence of frames or optical flow images, and the pixel-level temporal evolution across consecutive frames are seldom explored. To alleviate this issue, 3D ConvNets in Ji et al. (2013) is devised to directly learn spatio-temporal representation from a short video clip via 3D convolution. Tran *et al.* design a widely-adopted 3D ConvNets in Tran et al. (2015), namely C3D, consisting of 3D convolutions and 3D poolings optimized on the large-scale Sports1M (Karpathy et al., 2014) dataset. Despite having encouraging performances, the training of 3D ConvNets is computationally expensive and the model size suffers from a massive growth. Later in Qiu et al. (2017b); Tran et al. (2018); Xie et al. (2018), the decomposed 3D convolution is proposed to simulate one 3D convolution with one 2D spatial convolution plus one 1D temporal convolution. Recently, more advanced techniques are presented for 3D ConvNets, including inflating 2D convolutions (Carreira & Zisserman, 2017), non-local pooling (Wang et al., 2018c) and local-and-global diffusion (Qiu et al., 2019). These newly designed 3D ConvNets further show good transferability to several downstream tasks (Qiu et al., 2017c; Li et al., 2018; 2019; Long et al., 2019a;b; 2020).

Our work expands the research horizons of 3D ConvNets and focuses on improving 3D ConvNets training by adaptively planning the optimization process. The related works for 2D ConvNets training (Chee & Toulis, 2018; Lang et al., 2019; Yaida, 2019) automate the training strategy via only changing the learning rate adaptively. Our problem is much more challenging especially when temporal dimension is additionally considered and involved in the training scheme of 3D ConvNets. For enhancing 3D ConvNets training, the recent works (Wang et al., 2018c; Qiu et al., 2019) first train 3D ConvNets with short input clips and then fine-tune the network with lengthy clips, which balances training efficiency and long-range temporal modeling. The multigrid method (Wu et al., 2020) and ours both delve into the network training of 3D ConvNets, but along two distinct dimensions. Multigrid proposes to cyclically change spatial resolution and temporal duration of the input clips for a more efficient optimization of 3D ConvNets and the training strategy is still hand-designed. In contrast, ours studies the training of 3D ConvNets with multiple stages, but automatically schedules the change of hyper-parameters through optimization planning.

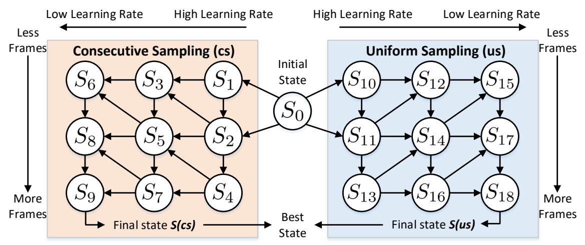
## 3. Optimization Planning

### 3.1. Problem Formulation

The goal of optimization planning is to automate the learning strategy of 3D ConvNets. Formally, the optimization process of 3D ConvNets can be represented as an optimization path $\mathcal{P} = \langle S_0, S_1, ..., S_N \rangle$, which consists of one initial state $S_0$ and $N$ intermediate states. Each intermediate state is assigned with the fixed hyper-parameters, and the training is performed with these $N$ different settings one by one. The training epoch on each setting is decided by $\mathcal{T} = \{t_1, t_2, ..., t_N\}$, in which $t_i$ denotes the number of epochs when moving from $S_{i-1}$ to $S_i$. The hyper-parameters include $sampling\_strategy \in \{cs,\ us\}$, $length\_of\_input\_clip \in \{l_1, l_2, ..., l_{N_l}\}$ and $learning\_rate \in \{r_1, r_2, ..., r_{N_r}\}$, where $cs$ and $us$ denotes consecutive sampling and uniform sampling from input videos, respectively. As a result, there are $2 \times N_l \times N_r$ valid types of training states.



(a) Basic transition graph



(b) Extended transition graph

*Figure 1.* Examples of two kinds of transition graphs. The circles denote the candidate states and the arrows represent the candidate transitions. The ultimate model is the one with higher accuracy of the two final states.

The objective function of optimization planning is to seek the optimal strategy $\{\mathcal{P}, \mathcal{T}\}$ by maximizing the performance of the final state $S_N$ as

$$\underset{\mathcal{P},\mathcal{T}}{maximize}\, \mathcal{V}(S_N), \tag{1}$$

where $\mathcal{V}(\cdot)$ is the performance metric, i.e., mean accuracy on validation set in our case.

### 3.2. Optimization Path

To plan the optimal permutation of training states, we first choose a final state $S_N$, which is usually with low learning rate and lengthy input clip. Then, the problem of seeking an optimal optimization path to $S_N$ is naturally decomposed to the subproblem of finding the optimization path to an intermediate state $S_i$ and the state transition from $S_i$ to $S_N$. As such, the problem can be solved by dynamic programming. Formally, the solution of optimization path $\mathcal{P}(S_N)$ can be given in a recursive form:

$$\mathcal{P}(S_N) = \langle \mathcal{P}(S_{i^*}), S_N \rangle\,, \quad i^* = \underset{i}{argmax}\,\{\mathcal{V}(S_i \rightarrow S_N)\}\,. \tag{2}$$

When executing the transfer from the state $S_i$ to the state $S_N$, we fine-tune the 3D ConvNets at the state $S_i$ by using the hyper-parameters at the state $S_N$. We then evaluate such fine-tuned model on the validation set to measure the priority of this transition, i.e., $\mathcal{V}(S_i \rightarrow S_N)$. We choose the state $S_{i^*}$, which achieves the highest priority of transition to the state $S_N$, as the preceding state of $S_N$. In other words, the optimal path for $S_N$ derives from the best-performing preceding state $S_{i^*}$. Here, we propose to pre-define all the valid transitions in a directed acyclic graph and determine the best optimization path of each state one by one in the topological order. Figure 1(a) shows one example of the pre-defined transition graph. In the example, we set the number of candidate input clip lengths $N_l = 3$ and the number of candidate learning rates $N_r = 3$. Hence, there are $2 \times 3 \times 3 = 18$ candidate states. Next, we capitalize on the following principles to determine the possible transitions, i.e., the connections between states:

(1) The transitions between the states with different sampling strategies are forbidden. $S_9$ and $S_{18}$ is the final state for consecutive and uniform sampling, respectively.

(2) The training only starts from high learning rate and short input video clips.

(3) The intermediate state can be only transferred to a new state, where either the learning rate is decreased or the length of input clip is increased in the new state.

Please note that, some very specific learning rate strategies, e.g., schedules with restart or warmup, show that increasing the learning rate properly may benefit training. Nevertheless, there is still no clear picture of when to increase the learning rate, and thus it is very difficult to automate these schedules. In the works of adaptively changing the learning rate for 2D ConvNets training (Ge et al., 2019; Lang et al., 2019; Yaida, 2019), such cyclic schedules are also not taken into account. Similarly in this work, we only consider the schedule of decreasing learning rate in the transition graph.

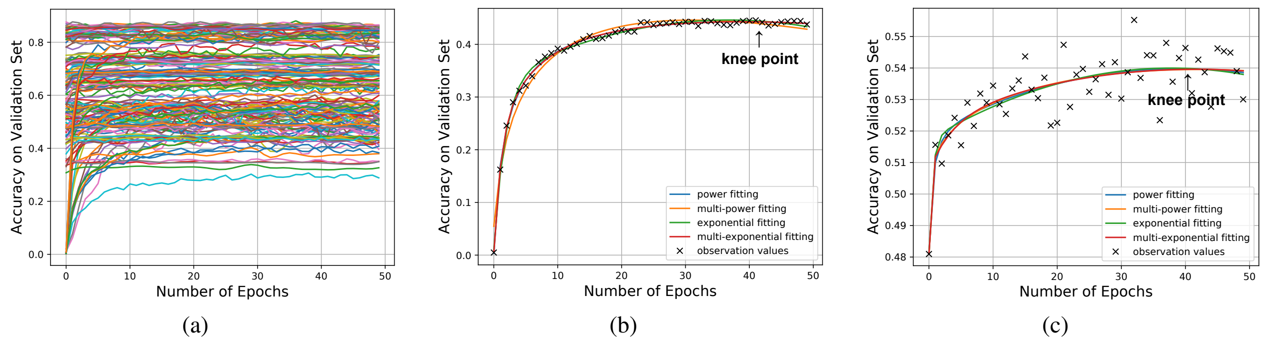
The aforementioned principles can simplify the transition graph and reduce the time cost when solving Equ.(2). We take this graph as **basic transition graph**. Furthermore, we also build an **extended transition graph** by enabling to simultaneously decrease the input clip length and the learning rate, as illustrated in Figure 1(b). In such graph, the training strategies are more flexible.

### 3.3. State Transition

One state transition from $S_i$ to $S_j$ is defined as a training step that starts to optimize the model at $S_i$ by using the hyper-parameters at $S_j$. Then the question is when this training step completes. Here, we derive the spirit from SASA (Lang et al., 2019) that trains the network with constant hyper-parameters until it reaches a stationary condition. SASA presents to adaptively evaluate the convergence of stochastic gradient descent by Yaida's condition (Yaida, 2019) during training. However, in practice, the thoroughly optimized network does not always perform well on validation set due to overfitting problem. Therefore, we take both convergence and overfitting into account, and propose to estimate the knee point on the performance-epoch curve evaluated on the validation set. That performs more steadily

*Table 1.* The comparisons of four fitting functions in terms of RMSE and R-Square.

| Fitting Function $f_\alpha(t)$ | Constraints | RMSE | R-Square |
|---|---|---|---|
| **power:** $\alpha_1 + \alpha_2(t+1)^{\alpha_3} + \alpha_4 t + \alpha_5 t^2$ | $\alpha_2, \alpha_3, \alpha_5 < 0$ | $1.010 \times 10^{-3}$ | 0.356 |
| **multi-power:** $\alpha_1 + \alpha_2(t+1)^{\alpha_3} + \alpha_4(t+1)^{\alpha_5} + \alpha_6 t + \alpha_7 t^2$ | $\alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_7 < 0$ | $1.030 \times 10^{-3}$ | 0.320 |
| **exponential:** $\alpha_1 + \alpha_2 e^{\alpha_3 t} + \alpha_4 t + \alpha_5 t^2$ | $\alpha_2, \alpha_3, \alpha_5 < 0$ | $\mathbf{1.007 \times 10^{-3}}$ | **0.360** |
| **multi-exponential:** $\alpha_1 + \alpha_2 e^{\alpha_3 t} + \alpha_4 e^{\alpha_5 t} + \alpha_6 t + \alpha_7 t^2$ | $\alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_7 < 0$ | $1.063 \times 10^{-3}$ | 0.350 |



(a) (b) (c)

*Figure 2.* The examples of (a) the collected performance-epoch curves, (b) the fitting results for training model from scratch, and (c) the fitting results for fine-tuning model.

across various datasets. Specifically, we measure the accuracy $y_t$ by evaluating the intermediate model after $t$-th training epoch on validation set. To estimate the knee point given a limited number of observations $y_t$, we fit the curve by a continuous function $f_\alpha(t)$ as

$$y_t = f_\alpha(t) + z_t, \;\; z_t \sim \mathcal{N}(0, \sigma^2), \tag{3}$$

where $z_t$ is the stochastic factor following a normal distribution, and $\alpha$ denotes the parameters of function $f$. Here, we choose $f_\alpha(t)$ as a unimodal function to ensure that there is only one maximum value. The curve fitting can be formulated as the optimization of parameters $\alpha$ by minimizing the distance between the observed performance and the estimated performance:

$$\underset{\alpha}{minimize} \sum_{0}^{t} \|y_t - f_\alpha(t)\|^2, \quad s.t. \;\; f_\alpha(t) \text{ is unimodal.} \tag{4}$$

We exploit Trust Region Reflective algorithm (Branch et al., 1999) to solve this problem and the algorithm is robust for arbitrary form of function $f_\alpha(t)$. To adaptively stop the iteration, we estimate the knee point epoch $t^*$ by solving Equ.(4) after each training epoch. If the current epoch $t$ is larger than $t^* + T$, we will stop the iteration and choose $t^*$ as the best epoch number. $T$ is a delay parameter which allows the model to have a $T$-epoch attempt even if $t > t^*$. We simply fix the delay parameter $T$ to 10 in all the experiments.

Next, the essential issue is the form of fitting function $f_\alpha(t)$. We separate the function into two parts $f_\alpha(t) = g_\alpha(t) + h_\alpha(t)$, where $g_\alpha(t)$ is an increasing bounded function to simulate the convergence of the model, and $h_\alpha(t)$ is a concave function to model the influence of overfitting. Table 1 shows four examples of fitting function $f_\alpha(t)$. In the four functions, we fix $h_\alpha(t)$ as a quadratic function and exploit **power**, **multi-power**, **exponential** and **multi-exponential** function as $g_\alpha(t)$, respectively. Please note that, for each function, some constraints are given to guarantee the properties of $g_\alpha(t)$ and $h_\alpha(t)$. We empirically validate the functions by pre-collecting 162 performance-epoch curves (Figure 2(a)) from the training processes of different networks on various datasets and employing the four functions to fit the curves through solving Equ.(4). Table 1 compares the average Root Mean Square Error (**RMSE**) and **R-square** when using different functions. Figure 2(b) and Figure 2(c) further depict a fitting example in the context of model training from scratch and model fine-tuning, respectively. The general observation is that, all the four functions can nicely fit the performance-epoch curve and do not make a major difference on the final performance. Hence, we simply choose the best-performing exponential function in the rest of the paper.

## 4. 3D ConvNets Architecture

In this section, we present the proposed **Dual-head Global-contextual Pseudo-3D** (DG-P3D) network and Figure 3 shows an overview. In particular, the network is originated from the residual network (He et al., 2016) and further extended to 3D manner with three designs, i.e., pseudo-3D convolution, global context and dual-head classifier.

**Pseudo-3D convolution.** To achieve a good tradeoff between accuracy and computational cost, pseudo-3D convolution proposed in Qiu et al. (2017b) decomposes 3D learning into 2D convolutions in spatial space plus 1D operations in

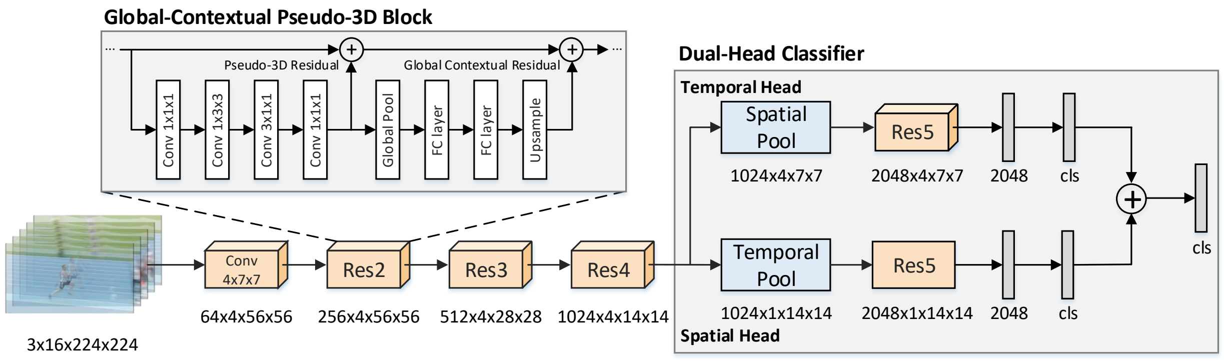


*Figure 3.* An overview of our proposed Dual-head Global-contextual Pseudo-3D (DG-P3D) network. Here, we take the 16-frame input as an example and the size of output feature map is also given for each layer.

temporal dimension. The similar idea of decomposing 3D convolution is also presented in R(2+1)D (Tran et al., 2018) and S3D (Xie et al., 2018). In this paper, we choose the mixed P3D architecture with the highest performance in Qiu et al. (2017b), which interleaves three types of P3D blocks.

**Global context.** The recent works on non-local networks (Wang et al., 2018c; Cao et al., 2019; Qiu et al., 2019) highlight the drawback of performing convolutions, in which each operation processes only a local window of neighboring pixels and lacks a holistic view of field. To alleviate this limitation, we delve into global context to learn the global residual from the global-pooled representation and then broadcast to each position in the feature map.

**Dual-head classifier.** 3D ConvNets are expected to have both spatial and temporal discrimination. For example, the SlowFast network (Feichtenhofer et al., 2019) contains an individual pathway for visual appearance and temporal dynamics, respectively. Here, we uniquely propose a simpler way that builds a dual-head classifier at the top of the network instead of the two-path structure in the SlowFast network. In between, the temporal head with large temporal dimension focuses on modeling the temporal evolution, and the spatial head with large spatial resolution emphasizes the spatial discrimination. The predictions from two heads are linearly fused. As such, our design costs less computations and is easier to implement.

## 5. Experiments

### 5.1. Datasets

The experiments are conducted on HMDB51, UCF101, ActivityNet, SS-V1/V2, Kinetics-400 and Kinetics-600 datasets. Table 2 details the information and settings on these datasets. The HMDB51 (Kuehne et al., 2011), UCF101 (Soomro et al., 2012), Kinetics-400 (Carreira & Zisserman, 2017) and Kinetics-600 (Carreira et al., 2018) are the most popular video benchmarks for action recognition on trimmed video clips. The Something-Something V1 (SS-V1) dataset is firstly constructed in Goyal et al. (2017) to learn fine-grained human-object interactions, and then extended to Something-Something V2 (SS-V2) recently. The ActivityNet (Caba Heilbron et al., 2015) dataset is an untrimmed video benchmark for activity recognition. The latest released version of the dataset (v1.3) is exploited. In our experiments, we only use the video-level label of ActivityNet and disable the temporal annotations. Note that the labels for test sets are not publicly available, and thus the performances of ActivityNet, SS-V1, SS-V2, Kinetics-400 and Kinetics-600 are all reported on the validation set. For optimization planning, the original training set of each dataset is split into two parts for learning the network weights and validating the performance, respectively. We construct this internal validation set with the same size as the original validation/test set. Note that the original validation/test set is never exploited in the optimization planning.

*Table 2.* The number of videos, the number of video categories, and the detailed settings for optimization planning, respectively, on HMDB51, UCF101, ActivityNet, SS-V1, SS-V2, Kinetics-400, and Kinetics-600 datasets.

| Dataset | #videos | #classes | $l_1$ | $l_2$ | $l_3$ | $r_1$ | $r_2$ | $r_3$ | dropout |
|---|---|---|---|---|---|---|---|---|---|
| HMDB51 | 6K | 51 | 16 | 32 | 64 | 0.01 | 0.001 | 0.0001 | 0.9 |
| UCF101 | 13K | 101 | 16 | 32 | 64 | 0.01 | 0.001 | 0.0001 | 0.9 |
| ActivityNet | 20K | 200 | 16 | 32 | 128 | 0.01 | 0.001 | 0.0001 | 0.9 |
| SS-V1 | 108K | 174 | 16 | 32 | – | 0.04 | 0.004 | 0.0004 | 0.5 |
| SS-V2 | 220K | 174 | 16 | 32 | – | 0.04 | 0.004 | 0.0004 | 0.5 |
| Kinetics-400 | 300K | 400 | 16 | 32 | 128 | 0.04 | 0.004 | 0.0004 | 0.5 |
| Kinetics-600 | 480K | 600 | 16 | 32 | 128 | 0.04 | 0.004 | 0.0004 | 0.5 |

### 5.2. Implementation Details

For **optimization planning**, we set the number of the choices for both input clip length $N_l$ and learning rate $N_r$ as 3, and utilize the extended transition graph introduced in Section 3.2. The candidate values of input clip length $\{l_1, l_2, l_3\}$ and learning rate $\{r_1, r_2, r_3\}$ for each dataset are summarized in Table 2. Specifically, on SS-V1, SS-V2,

*Table 3.* The comparisons between optimization planning (OP) and hand-tuned strategies (HS) with DG-P3D on Kinetics-400 dataset. The number in the bracket denotes the best number of epoches, which is achieved by grid-search for hand-tuned strategies and adaptively determined by our optimization planning.

| Method | Sampling | Clip length | Learning rate strategy 3-step | Learning rate strategy cosine |
|---|---|---|---|---|
| HS | *consecutive* | $l_1 \to l_3$ | 77.3 (256) | 77.6 (192) |
| | | $l_2 \to l_3$ | 77.8 (320) | **78.0** (256) |
| | | $l_1 \to l_2 \to l_3$ | 77.5 (320) | 77.9 (256) |
| | *uniform* | $l_1 \to l_3$ | 76.5 (192) | 76.9 (128) |
| | | $l_2 \to l_3$ | 76.8 (256) | 76.9 (256) |
| | | $l_1 \to l_2 \to l_3$ | 76.8 (192) | 77.1 (192) |
| OP | **78.9** (220) | | | |

*Table 4.* The comparisons between optimization planning (OP) and hand-tuned strategies (HS) with different 3D ConvNets on Kinetics-400 dataset. All the backbone networks are ResNet-50.

| Network | Training strategy | |
|---|---|---|
| SlowFast (Feichtenhofer et al., 2019) | cosine decay 77.0 | multigrid 77.6 |
| | HS | OP |
| I3D (Carreira & Zisserman, 2017) | 75.6 | 76.4 (↑0.8) |
| P3D (Qiu et al., 2017b) | 75.2 | 76.0 (↑0.8) |
| R(2+1)D (Xie et al., 2018) | 75.1 | 76.2 (↑1.1) |
| LGD-3D (Qiu et al., 2019) | 76.3 | 77.3 (↑1.0) |
| | HS | OP |
| G-P3D | 77.1 | 77.9 (↑0.8) |
| DG-P3D | 78.0 | **78.9** (↑0.9) |

Kinetics-400 and Kinetics-600 datasets, the base learning rate is set as $0.04$ and the dropout ratio is fixed as 0.5. For HMDB51, UCF101 and ActivityNet, we set lower base learning rate and higher dropout ratio due to limited training samples. The maximum clip length is 64 for HMDB51 and UCF101, and is increased to 128 for ActivityNet, Kinetics-400 and Kinetics-600. Considering that the video clips in SS-V1 and SS-V2 are usually shorter than 64 frames, we only use two settings, i.e., 16-frame and 32-frame.

The **network training** is implemented on PyTorch framework and the mini-batch stochastic gradient descent is employed to tune the network. The resolution of the input clip is fixed as $224 \times 224$, which is randomly cropped from the video clip resized with the short size in $[256, 340]$. The clip is randomly flipped along horizontal direction for data augmentation except for SS-V1 and SS-V2 in view of the direction-related categories. Following the settings in Wang et al. (2018c); Qiu et al. (2019), for network training with long clips (64-frame and 128-frame), we freeze the parameters of all Batch Normalization layers except for the first one since the batch size is too small for batch normalization.

There are two **inference strategies** for the evaluations. The first one roughly predicts the video label on a $224 \times 224$ single center crop from the centric one clip resized with the short size 256. This strategy is only used when planning the optimization for the purpose of efficiency. Once the optimization path is fixed, we train 3D ConvNets with the path and evaluate the learnt 3D ConvNets by using the second strategy, i.e., the three-crop strategy as in Feichtenhofer et al. (2019), which crops three $256 \times 256$ regions from each video clip. The video-level prediction score is achieved by averaging all scores from ten uniform sampled clips.

## 5.3. Evaluation of Optimization Planning

We firstly verify the effectiveness of our proposed optimization planning for 3D ConvNets and compare the hand-tuned strategies. To find the most powerful hand-tuned strategy, we capitalize on the popular practices in the literature, and grid-search the training settings through four dimensions, i.e., input clip length, learning rate decay, sampling strategy and training epochs. Specifically, for input clip length, we follow the common training scheme that first learns the network with short clips and then fine-tunes the network on lengthy clips, and experiment with three strategies $l_1 \to l_3$, $l_2 \to l_3$ and $l_1 \to l_2 \to l_3$. For each input clip length, we train the network with the same number of epochs. For learning rate decay, we choose two mostly utilized strategies, i.e., 3-step learning rate decay (Wang et al., 2018c; Qiu et al., 2019) and cosine decay (Feichtenhofer et al., 2019). The optimal training epoch for each strategy is determined by grid-searching from $[128, 192, 256, 320, 384]$ epochs.

Table 3 shows the comparisons between optimization planning and hand-tuned strategies with DG-P3D architecture on Kinetics-400 dataset. The backbone network is derived from ResNet-50 pre-trained on ImageNet dataset. The results constantly indicate that optimization planning exhibits better performance than hand-tuned strategies. In particular, training DG-P3D with optimization planning leads to a performance boost against the network learnt with best-performing hand-tuned strategy by 0.9%. That basically verifies the advantage of dynamically determining the training strategy. Moreover, Table 4 summarizes the two training strategies on six different 3D ConvNets, i.e., I3D, P3D, R(2+1)D, LGD-3D, G-P3D and our DG-P3D. In between, G-P3D extends the P3D network by employing global context, and DG-P3D further devises dual-head classifier. Here, we also include SlowFast network (Feichtenhofer et al., 2019) trained with either cosine learning rate decay or multigrid method (Wu et al., 2020). Overall, optimization planning makes the absolute improvement over hand-tuned strategies by 0.8%$\sim$1.1% across six 3D ConvNets, demonstrating its generalizability to different 3D architectures. With the same optimization planning strategy, DG-P3D network achieves 1.0% improvement over G-P3D, which validates the design of dual-head classifier. Though both multigrid method and optimization planning involve the network training in stages with multiple hyper-parameters, they are different in the way that multigrid pre-defines the change of hyper-parameters in

*Table 5.* The comparisons between optimization planning and hand-tuned strategy with DG-P3D on HMDB51 (split 1), UCF101 (split1), ActivityNet, SS-V1, SS-V2 and Kinetics-400 datasets. The backbone is ResNet-50 pre-trained on ImageNet. The time cost for grid search/optimization planning is reported with 8 NVidia Titan V GPUs in parallel.

| **Strategy** | | **HMDB51** | **UCF101** | **ActivityNet** | **SS-V1** | **SS-V2** | **Kinetics-400** |
|---|---|---|---|---|---|---|---|
| Hand-tuned Strategy | *top-1* | 57.7 | 88.2 | 75.2 | 51.4 | 63.9 | 78.0 |
| | *time cost* | 83h | 158h | 166h | 540h | 1072h | 4057h |
| Optimization Planning | *top-1* | **58.4** (↑0.7) | **89.1** (↑0.9) | **76.5** (↑1.3) | **52.8** (↑1.4) | **65.5** (↑1.6) | **78.9** (↑0.9) |
| | *time cost* | 6h | 13h | 38h | 67h | 142h | 288h |

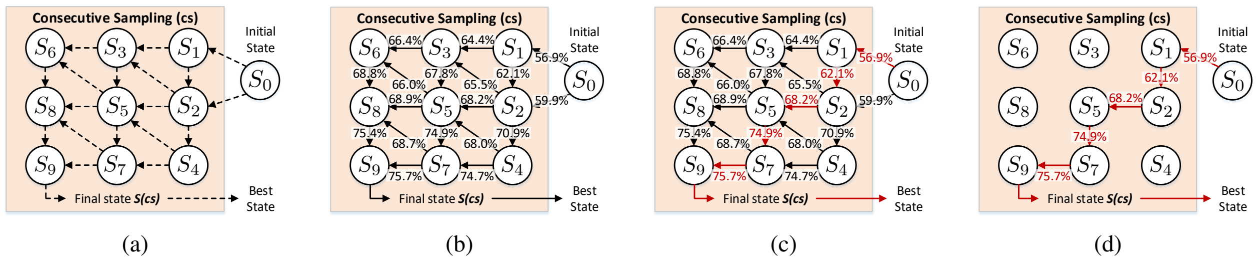


*Figure 4.* An example of optimization planning procedure with consecutive sampling strategy on Kinetics-400 dataset. The procedure includes (a) the extended transition graph, (b) exploring all the candidate transitions, (c) seeking the optimal path by maximizing the performance of the final state, (d) forming the optimization path. The clip-level accuracy is also given for each explored transition.

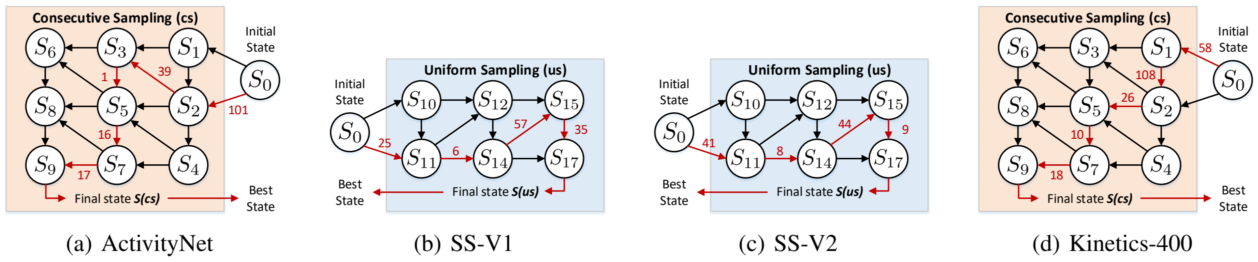


*Figure 5.* The optimization path produced by optimization planning on (a) ActivityNet, (b) SS-V1, (c) SS-V2, (d) Kinetics-400. The red edge represents the state transition in the optimization path, and the black edges denote the transitions that have been explored but not selected in the final optimization path. The optimal number of training epochs is also given for each transition in the path.

each stage, and optimization planning adaptively schedules such change. As indicated by our results, DG-P3D with optimization planning leads the accuracy by 1.3%, against SlowFast network with multigrid strategy.

Taking our DG-P3D as 3D ConvNets, Table 5 further details the comparisons between optimization planning and the hand-tuned strategy across six different datasets. The accuracy of the hand-tuned strategy is reported on the best training scheme by grid search on each dataset. Such best hand-tuned strategy can be considered as a well-tuned DG-P3D model without optimization planning. The time cost of optimization planning contains the training time of exploring all the possible transitions, and that of hand-tuned strategy is measured by grid-searching the candidate training strategies. Compared to the hand-tuned strategy, optimization planning shows consistent improvements across different datasets, and requires much less time than the exhaustive grid search, due to adaptive determination of training scheme.

## 5.4. Qualitative Analysis of Optimization Planning

Figure 4 illustrates the process of optimization planning with consecutive sampling strategy on Kinetics-400 dataset. The candidate transitions between states are explored in topological order, and the optimal path of each state derives from the best-performing preceding state. We also examine how optimization path impacts the performance and experiment with some variant paths on Kinetics-400, which are built by either inserting an additional state or skipping an intermediate state in our adopted optimization path. For fair comparisons, the numbers of epochs in these variant paths are re-determined by the algorithm in Section 3.3. The results indicate that inserting and skipping one state result in an accuracy decrease of 0.2%$\sim$0.6% and 0.3%$\sim$1.0%, respectively, and verify the impact of optimization planning.

Next, Figure 5 depicts the optimization paths on different datasets. An interesting observation is that SS-V1/2 tend to select uniform sampling while Kinetics-400 prefers consecutive sampling. We speculate that this may be the result

*Table 6.* Comparisons with the state-of-the-art methods on (a) HMDB51 (3 splits) & UCF101 (3 splits) and (b) ActivityNet with RGB input.

(a) HMDB51 (H51) & UCF101 (U101)

| Method | Backbone | H51 | U101 |
|---|---|---|---|
| I3D (Carreira & Zisserman, 2017) | BN-Inception | 74.5 | 95.4 |
| ARTNet (Wang et al., 2018a) | BN-Inception | 70.9 | 94.3 |
| ResNeXt (Hara et al., 2018) | ResNeXt-101 | 70.2 | 94.5 |
| R(2+1)D (Tran et al., 2018) | ResNet-34 | 74.5 | 96.8 |
| S3D-G (Xie et al., 2018) | BN-Inception | 75.9 | 96.8 |
| STM (Jiang et al., 2019) | ResNet-50 | 72.2 | 96.2 |
| LGD-3D (Qiu et al., 2019) | ResNet-101 | 75.7 | 97.0 |
| **DG-P3D** | ResNet-50 | 79.2 | 97.6 |
| | ResNet-101 | **80.4** | **97.9** |

(b) ActivityNet

| Method | Backbone | +K400 | Top-1 |
|---|---|---|---|
| TSN (Wang et al., 2018b) | BN-Inception | | 72.9 |
| RRA (Zhu et al., 2018) | ResNet-152 | | 78.8 |
| MARL (Wu et al., 2019) | ResNet-152 | | 79.8 |
| TSN (Wang et al., 2018b) | BN-Inception | ✓ | 78.9 |
| MARL (Wu et al., 2019) | SEResNeXt152 | ✓ | 85.7 |
| **DG-P3D** | ResNet-50 | | 76.5 |
| | ResNet-50 | ✓ | 85.9 |
| | ResNet-101 | | 77.8 |
| | ResNet-101 | ✓ | **86.8** |

*Table 7.* Performance comparisons with the state-of-the-art methods on SS-V1 and SS-V2 with RGB input. All the backbone networks are ResNet-50.

| Method | Pre-train | SS-V1 | | SS-V2 | |
|---|---|---|---|---|---|
| | | Top-1 | Top-5 | Top-1 | Top-5 |
| NL I3D+GCN (Wang & Gupta, 2018) | Kinetics | 46.1 | 76.8 | – | – |
| TSM (Lin et al., 2019) | Kinetics | 47.2 | 77.1 | 63.4 | 88.5 |
| bLVNet-TAM (Fan et al., 2019) | ImageNet | 48.4 | 78.8 | 61.7 | 88.1 |
| ABM-C-in (Zhu et al., 2019) | ImageNet | 49.8 | – | 61.2 | – |
| I3D+RSTG (Nicolicioiu et al., 2019) | Kinetics | 49.2 | 78.8 | – | – |
| GST (Luo & Yuille, 2019) | ImageNet | 48.6 | 77.9 | 62.6 | 87.9 |
| STDFB (Martínez et al., 2019) | ImageNet | 50.1 | 79.5 | – | – |
| STM (Jiang et al., 2019) | ImageNet | 50.7 | 80.4 | 64.2 | 89.8 |
| TEA (Li et al., 2020b) | ImageNet | 51.9 | 80.3 | – | – |
| ASS (Li et al., 2020a) | ImageNet | 51.4 | – | 63.5 | – |
| **DG-P3D** | ImageNet | **52.8** | **81.8** | **65.5** | **90.3** |

of different emphases of the two sampling strategies. In general, the most special on uniform sampling is to capture the completeness of a video with only a small number of sampled frames. In contrast, consecutive sampling emphasizes the continuity in a video but may only focus on a part of the video content. The SS-V1/V2 datasets consist of fine-grained interactions and the differentiation between these interactions relies more on the completeness of an action. For example, it is almost impossible to distinguish the videos of the category “Pushing something so that it falls off the table” from those of “Pushing something so that it almost falls off but doesn’t,” if only based on part of the video content. In other words, uniform sampling offers the completeness of a video and benefits the recognition on SS-V1/V2. Instead, the videos in Kinetics-400 are usually with static scenes or slow motion. Hence, the completeness may not be essential in this case, but consecutive sampling encodes the continuous changes across frames and thus captures the spatio-temporal relation better.

## 5.5. Comparisons with State-of-the-Art

We compare with several state-of-the-art techniques on **HMDB51**, **UCF101** and **ActivityNet** datasets. The performance comparisons are summarized in Table 6. The backbone of DG-P3D is either ResNet-50 or ResNet-101 pre-trained on ImageNet. Please note that most recent works employ Kinetics-400 pre-training to improve the accuracy.

*Table 8.* Comparisons with state-of-the-art methods on Kinetics-400 & Kinetics-600 with RGB input. The computational complexity is measured in GFLOPs × views and the views represent the number of clips sampled from the full video during inference. * In view that it is not that fair to directly compare irCSN pre-trained on IG65M (65M web videos) and other methods, here we report the performance of irCSN pre-trained on Sports1M.

| Method | Backbone | GFLOPs×views | Kinetics-400 (top-1/5) | Kinetics-600 (top-1/5) |
|---|---|---|---|---|
| I3D | BN-Inception | 108×N/A | 72.1/90.3 | – |
| R(2+1)D | custom | 152×115 | 74.3/91.4 | – |
| S3D-G | BN-Inception | 66.4×N/A | 74.7/93.4 | – |
| NL I3D | ResNet-101 | 359×30 | 77.7/93.3 | – |
| LGD-3D | ResNet-101 | 195×N/A | 79.4/94.4 | 81.5/95.6 |
| X3D-XL | custom | 48.4×30 | 79.1/93.9 | 81.9/95.5 |
| irCSN* | custom | 96.7×30 | 79.0/93.5 | – |
| SlowFast | ResNet-50 | 65.7×30 | 77.0/92.6 | 79.9/94.5 |
| | ResNet-101 | 213×30 | 78.9/93.5 | 81.1/95.1 |
| | ResNet-101+NL | 234×30 | 79.8/93.9 | 81.8/95.1 |
| **DG-P3D** | ResNet-50 | 123×30 | 78.9/93.9 | 81.6/95.6 |
| | ResNet-101 | 218×30 | **80.5/94.6** | **82.7/95.8** |

Here, we also choose the two-step strategy that first trains DG-P3D on Kinetics-400 (K400) and then fine-tunes the network on the target dataset. The two steps are both trained with optimization planning. Overall, DG-P3D achieves the highest performances on all the three datasets, i.e., 80.4% on HMDB51, 97.9% on UCF101 and 86.8% on ActivityNet. In particular, DG-P3D outperforms the other 3D ConvNets of I3D, R(2+1)D, S3D-G and LGD-3D by 5.9%, 5.9%, 4.5% and 4.7% on HMDB51, respectively. The results again verify the merit of the learnt 3D ConvNets. For ActivityNet, most baselines utilize the temporal annotation to locate the foreground segment in the untrimmed videos. In our experiments, we only use the video-level annotations and our DG-P3D still surpasses the best competitor MARL by 1.1%.

Then, we turn to evaluate DG-P3D with optimization planning on four large-scale datasets, i.e., **SS-V1**, **SS-V2**, **Kinetics-400** and **Kinetics-600**. To reduce the cost, the optimization path found on Kinetics-400 is directly utilized as the path for Kinetics-600. The top-1 and top-5 accuracies on the four datasets are reported in Table 7 and Table 8. Specifically, DG-P3D achieves the highest top-1 accuracy of 52.8% on SS-V1 and 65.5% on SS-V2. DG-P3D is

*Table 9.* Comparisons with state-of-the-art methods on Kinetics-400 & Kinetics-600 datasets with the input of flow modality, and the fusion of frame and flow modalities.

| **Method** | **Backbone** | **Kinetics-400** (top-1/5) | | **Kinetics-600** (top-1/5) | |
|---|---|---|---|---|---|
| | | Flow | Fusion | Flow | Fusion |
| I3D | BN-Inception | 65.3/86.2 | 75.7/92.0 | – | – |
| R(2+1)D | custom | 68.5/88.1 | 75.4/91.9 | – | – |
| S3D-G | BN-Inception | 68.0/87.6 | 77.2/93.0 | – | – |
| LGD-3D | ResNet-101 | 72.3/90.9 | 81.3/95.2 | 75.0/92.4 | 83.1/96.2 |
| **DG-P3D** | ResNet-50 | 72.0/90.9 | 80.4/94.7 | 74.8/93.2 | 82.7/95.9 |
| | ResNet-101 | **73.2/91.2** | **82.6/96.1** | **76.7/93.4** | **84.3/96.6** |

superior to TEA and STM, which reports the best known results, by 0.9% and 1.3%, respectively. On Kinetics-400, the top-1 accuracy of DG-P3D reaches 80.5%, which makes the improvements over the recent 3D ConvNets irCSN (Tran et al., 2019), X3D-XL (Feichtenhofer, 2020), LGD-3D (Qiu et al., 2019), SlowFast (Feichtenhofer et al., 2019) by 1.5%, 1.4%, 1.1% and 0.7%, respectively. The similar performance trends are also observed on Kinetics-600. DG-P3D achieves 82.7% top-1 accuracy, which leads to the performance boost of 0.8%, over the best competitor X3D-XL. For fair comparisons with the two-stream baselines on Kinetics datasets, we additionally consider the flow modality by the two-direction optical flow image extracted by TV-L1 algorithm (Zach et al., 2007) and directly use the optimal path on RGB modality as that on flow modality. Table 9 summarizes the performance comparisons. Overall, the two-stream DG-P3D achieves 82.6%/84.3% top-1 accuracy, which leads the performance by 1.3%/1.2%, against two-stream LGD-3D on Kinetics-400 and Kinetics600, respectively. The results also validate the use of the learnt optimization path on frame modality to flow modality and two-stream structures.

## 6. Conclusion

We have presented optimization planning which aims to automate the training scheme of 3D ConvNets. Particularly, a training process is decided by a sequence of training states, namely optimization path, plus the number of training epochs for each state. We specify the hyper-parameters in each state and the permutation of states determines the changes of hyper-parameters. Technically, we propose a dynamic programming method to seek the best optimization path in the candidate transition graph and each state transition is stopped adaptively by estimating the knee point on the performance-epoch curve. Furthermore, we devise a new 3D ConvNets, i.e., DG-P3D, with a unique design of the dual-head classifier. The results on seven video benchmarks, which are different in terms of data scale, target categories and video duration, validate our proposal. Notably, DG-P3D with optimization planning obtains superior performances on all the seven datasets.

**Acknowledgments.** This work was supported by the National Key R&D Program of China under Grant No. 2020AAA0108600.

## References

Branch, M. A., Coleman, T. F., and Li, Y. A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. *SIAM Journal on Scientific Computing*, 21:1–23, 1999.

Caba Heilbron, F., Escorcia, V., Ghanem, B., and Carlos Niebles, J. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, 2015.

Cao, Y., Xu, J., Lin, S., Wei, F., and Hu, H. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *ICCV Workshop*, 2019.

Carreira, J. and Zisserman, A. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.

Carreira, J., Noland, E., Banki-Horvath, A., Hillier, C., and Zisserman, A. A short note about kinetics-600. *ArXiv*, abs/1808.01340, 2018.

Chee, J. and Toulis, P. Convergence diagnostics for stochastic gradient descent with constant learning rate. In *AISTATS*, 2018.

Fan, Q., Chen, C.-F., Kuehne, H., Pistoia, M., and Cox, D. More is less: Learning efficient video representations by big-little network and depthwise temporal aggregation. In *NeurIPS*, 2019.

Feichtenhofer, C. X3d: Expanding architectures for efficient video recognition. In *CVPR*, 2020.

Feichtenhofer, C., Pinz, A., and Zisserman, A. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016.

Feichtenhofer, C., Fan, H., Malik, J., and He, K. Slowfast networks for video recognition. In *ICCV*, 2019.

Ge, R., Kakade, S. M., Kidambi, R., and Netrapalli, P. The step decay schedule: A near optimal, geometrically decaying learning rate procedure. *ArXiv*, abs/1904.12838, 2019.

Goyal, R., Kahou, S. E., Michalski, V., Materzynska, J., Westphal, S., Kim, H., Haenel, V., Fründ, I., Yianilos, P., Mueller-Freitag, M., Hoppe, F., Thurau, C., Bax, I., and Memisevic, R. The "something something" video database for learning and evaluating visual common sense. In *ICCV*, 2017.

Hara, K., Kataoka, H., and Satoh, Y. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet. In *CVPR*, 2018.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

Ji, S., Xu, W., Yang, M., and Yu, K. 3d convolutional neural networks for human action recognition. *IEEE Trans. on PAMI*, 35(1):221–231, 2013.

Jiang, B., Wang, M., Gan, W., Wu, W., and Yan, J. Stm: Spatiotemporal and motion encoding for action recognition. In *ICCV*, 2019.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., and Serre, T. HMDB: a large video database for human motion recognition. In *ICCV*, 2011.

Lang, H., Xiao, L., and Zhang, P. Using statistics to automate stochastic optimization. In *NeurIPS*, 2019.

Li, D., Qiu, Z., Dai, Q., Yao, T., and Mei, T. Recurrent tubelet proposal and recognition networks for action detection. In *ECCV*, 2018.

Li, D., Yao, T., Qiu, Z., Li, H., and Mei, T. Long short-term relation networks for video action detection. In *ACM MM*, 2019.

Li, D., Qiu, Z., Pan, Y., Yao, T., Li, H., and Mei, T. Representing videos as discriminative sub-graphs for action recognition. In *CVPR*, 2021.

Li, H., Zheng, W.-S., Tao, Y., Hu, H., and Lai, J.-H. Adaptive interaction modeling via graph operations search. In *CVPR*, 2020a.

Li, Y., Ji, B., Shi, X., Zhang, J., Kang, B., and Wang, L. Tea: Temporal excitation and aggregation for action recognition. In *CVPR*, 2020b.

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

Long, F., Yao, T., Qiu, Z., Tian, X., Luo, J., and Mei, T. Gaussian temporal awareness networks for action localization. In *CVPR*, 2019a.

Long, F., Yao, T., Qiu, Z., Tian, X., Mei, T., and Luo, J. Coarse-to-fine localization of temporal action proposals. *IEEE Trans. on MM*, 22(6):1577–1590, 2019b.

Long, F., Yao, T., Qiu, Z., Tian, X., Luo, J., and Mei, T. Learning to localize actions from moments. In *ECCV*, 2020.

Luo, C. and Yuille, A. L. Grouped spatial-temporal aggregation for efficient action recognition. In *ICCV*, 2019.

Martínez, B., Modolo, D., Xiong, Y., and Tighe, J. Action recognition with spatial-temporal discriminative filter banks. In *ICCV*, 2019.

Nicolicioiu, A. L., Duta, I., and Leordeanu, M. Recurrent space-time graph neural networks. In *NeurIPS*, 2019.

Qiu, Z., Yao, T., and Mei, T. Deep quantization: Encoding convolutional activations with deep generative model. In *CVPR*, 2017a.

Qiu, Z., Yao, T., and Mei, T. Learning spatio-temporal representation with pseudo-3d residual networks. In *ICCV*, 2017b.

Qiu, Z., Yao, T., and Mei, T. Learning deep spatio-temporal dependence for semantic video segmentation. *IEEE Trans. on MM*, 20(4):939–949, 2017c.

Qiu, Z., Yao, T., Ngo, C.-W., Tian, X., and Mei, T. Learning spatio-temporal representation with local and global diffusion. In *CVPR*, 2019.

Qiu, Z., Yao, T., Ngo, C.-W., Zhang, X.-P., Wu, D., and Mei, T. Boosting video representation learning with multi-faceted integration. In *CVPR*, 2021.

Simonyan, K. and Zisserman, A. Two-stream convolutional networks for action recognition in videos. In *NeurIPS*, 2014.

Soomro, K., Zamir, A. R., and Shah, M. UCF101: A dataset of 101 human action classes from videos in the wild. *CRCV-TR-12-01*, 2012.

Tran, D., Bourdev, L., Fergus, R., Torresani, L., and Paluri, M. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015.

Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., and Paluri, M. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, 2018.

Tran, D., Wang, H., Torresani, L., and Feiszli, M. Video classification with channel-separated convolutional networks. In *ICCV*, 2019.

Varol, G., Laptev, I., and Schmid, C. Long-term temporal convolutions for action recognition. *IEEE Trans. on PAMI*, 40(6):1510–1517, 2018.

Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., and Van Gool, L. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016.

Wang, L., Li, W., Li, W., and Gool, L. V. Appearance-and-relation networks for video classification. In *CVPR*, 2018a.

Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., and Van Gool, L. Temporal segment networks for action recognition in videos. *IEEE Trans. on PAMI*, 2018b.

Wang, X. and Gupta, A. Videos as space-time region graphs. In *ECCV*, 2018.

Wang, X., Girshick, R., Gupta, A., and He, K. Non-local neural networks. In *CVPR*, 2018c.

Wu, C.-Y., Girshick, R., He, K., Feichtenhofer, C., and Krahenbuhl, P. A multigrid method for efficiently training video models. In *CVPR*, 2020.

Wu, W., He, D., Tan, X., Chen, S., and Wen, S. Multi-agent reinforcement learning based frame sampling for effective untrimmed video recognition. In *ICCV*, 2019.

Xie, S., Sun, C., Huang, J., Tu, Z., and Murphy, K. Re-thinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *ECCV*, 2018.

Yaida, S. Fluctuation-dissipation relations for stochastic gradient descent. In *ICLR*, 2019.

Zach, C., Pock, T., and Bischof, H. A duality based approach for realtime tv-l1 optical flow. *Pattern Recognition*, 2007.

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

Zhu, X., Xu, C., Hui, L., Lu, C., and Tao, D. Approximated bilinear modules for temporal modeling. In *ICCV*, 2019.