Efficient Parallel Inflated 3D Convolution Architecture for Action Recognition

YUKUN HUANG, YONGCAI GUO, AND CHAO GAO

College of Optoelectronic Engineering, Chongqing University, Chongqing 400044, China

Corresponding author: Chao Gao (chaogaocqu@126.com)

This work was supported in part by the National key Research and Development Program of China under Grant 2016YFF0102802, and in part by the Hardware Grant from NVIDIA Corporation.

ABSTRACT
Deep neural networks have received increasing attention in human action recognition. Previous research has established that utilizing 3D convolution is a reasonable approach to learn spatio-temporal representation. Nevertheless, constructing effective 3D ConvNets usually need an expensive pre-training process that performing on a huge-scale video dataset. To avoid this burdensome situation, one major issue is to determine whether the pre-trained parameters of 2D convolution networks can be directly bootstrapped into 3D. In this paper, we devise a 2D-Inflated operation and a parallel 3D ConvNet architecture to solve this problem. The 2D-Inflated operation is used for converting pre-trained 2D ConvNets into 3D ConvNets, which avoiding video data pre-training. We further explore the optimal quantity of 3D ConvNet in the parallel architecture, and the results suggest that 6-nets architecture is an excellent solution for recognition. Another contribution of our study is two practical and valid skills, accumulated gradient descent and video sequence decomposition. Either of those techniques can promote the improvement of performance. The recognition results of UCF101 and HMDB51 reveal that, without the video data pre-training, our 3D ConvNets still can achieve competitive performance to the other generic and recent methods of using 3D ConvNets in the RGB image domain.

INDEX TERMS
Action recognition, image sequence analysis, 3D convolution network, 2D-inflated, parallel feature extraction, accumulated gradient descent.

I. INTRODUCTION

Deep learning achieves significant success in image processing and have been extensively used for action recognition. 3D convolution is an important research hotspot in this area. Recently, in order to eliminate the weaknesses of 2D CNN in capturing temporal feature and motion information, many investigators have constructed various 3D CNN structure. These previous studies of 3D ConvNets have established that video data pre-training is extremely attractive for improving recognition capability. However, the pre-training process of 3D convolution networks not only requires massive video data, but also demands enormous hardware resources. The recent research also suggests that Kinetics [1], or other equivalent scale video datasets, such as Sports1M [2], is desirable for pre-training 3D ConvNets [3]. Furthermore, redesigned 3D CNN are unable to receive the benefits from ImageNet pre-trained 2D CNN, because it is difficult to adapt the 2D parameters to a new 3D kernel. In the study [4], after a Kinetics pre-training, I3D that based on the recycle of 2D architecture achieves remarkable performance. This research is both exciting and dispiriting, revealed that the expensive video data pre-training is one of critical aspect for improving accuracy. According to such results, one may consider that the parameters of 2D CNN learned from image domain are not useful for 3D CNN in video tasks.

In this paper, we challenge this situation and revisit the parallel extraction pattern in temporal reasoning by exploring five parallel 3D ConvNet architectures and two input structural design. Despite it is a common practice for 3D ConvNets to improve accuracy by video data pretraining [3]–[8], here we construct 3D ConvNets by reusing the Imagenet trained 2D models, which have shown surprising effects in the field of image classification. We demonstrate that these 2D ConvNets still can provide enough effective pre-trained parameters for the
corresponding 3D ConvNets. This allows us escape from the expensive video data pre-training.

Inspired by previous studies, our parallel 3D ConvNet architecture primarily makes improvements on two structural hierarchies. The first aspect concentrates on the construction of 3D filters. It is responsible for converting pre-trained filters of 2D ConvNets into 3D structure, which is called 2D-Inflated operation in this work. The motivation of such operation stems from the studies that utilize 2D ConvNets for video processing [9], [10]. In this case, the inflated 3D kernels can be viewed as the organized combination of the same 2D kernels. More precisely, we grows the identical 2D filters along the temporal dimension until a certain 3D size. When performing 3D filters on video data, it seems like employing parallel 2D filters on frame sequence simultaneously. Besides, the inflated 3D ConvNets can further learn abstract spatio-temporal representation through the video sequence training process. The second focuses on the quantity of 3D ConvNets involved in spatio-temporal feature extraction. This hierarchy is used for analyzing temporal characteristic within long-term content. The inspiration is partly characterized by the framework proposed in [11]. Moreover, we extend such extraction pattern to a scalable form for a wider and deeper evaluation. Stated differently, this system can implement the extraction architecture with different amounts of 3D ConvNet branch. During the exploration, a counter-intuitive phenomenon is that equipping more 3D ConvNet branch leads to the performance degradation if there is no reduction in the temporal information entered by each branch. The empirical evidences demonstrate that the amount of 3D ConvNet is inversely proportional to the input frame number of each branch. The overall diagram of our architecture is shown in Figure 1.

Our contribution in general is the goal of training the 3D ConvNets based models for effective video analysis at a lower cost. In most practices, the video training process consumes more computing resources than image training, and this condition also causes the training is limited to a small batch size. As a result, we introduce the accumulated gradient descent for the purpose of enhancing the video training effect under a limited condition. We confirm that the accumulated gradient descent outperforms traditional stochastic gradient descent through comparative experiments. We conduct our researches on commonly used action recognition benchmarks, HMDB51 [12] and UCF101 [13].

The next section is the overview of the existing methods for action recognition. Section III gives a detail description of proposed approach in this paper. Section IV reports the performance of models on both datasets during the exploration and analyzes the experimental results. Section V concludes and discusses the findings of this article.

II. RELATED WORK

Compared to traditional hand-crafted methods that depends on low-level characteristics [14]–[17], deep learning is an end-to-end training process and extract high-level abstract features. Successful deep learning architectures in action recognition can be divided into three categories: 1) The 2D ConvNets based two-stream networks( [9], [11], [18], [19]); 2) The architectures that combine 2D ConvNets with LSTM( [20]–[23]); 3) The 3D ConvNets with various structures of 3D filters( [3], [5], [6], [24]–[26]). The following part will introduce such three categories in detail.

A. TWO-STREAM DEEP NETWORKS

Two-stream architecture is a practical approach that first proposed by Simonyan and Zisserman [18]. This structure is mainly composed of the spatial stream and the temporal stream that operates on RGB images and optical-flow images, respectively. The final prediction is a fusion result that is made through the features extracted from both image domain. Temporal Segment Network(TSN) introduces video segment method and deep residual modules to this architecture [11]. This provides TSN the ability of capturing independent temporal feature from three separated video segments, which can contribute to comprehend long-term content. Trajectory-Pooled Deep-Convolutional Descriptors(TDD) [10] is a successful way of combining two-stream network and traditional trajectory features [27]. The study [19] creatively adopted the Spatial Pyramid Network (SPyNet) to extract the optical-flow images, which provided an additional stream for prediction. It constructed a 3-stream (RGB stream, TV-L1 stream, SPyNet stream) framework for action recognition by using ImageNet pre-trained 2D ResNet152. Two-stream networks are commonly built upon successful image classification models in ImageNet. It is needless to say that the pre-trained models can be directly applied in the two-stream architecture. However, it seems difficult for 2D convolution to extract spatio-temporal information between image sequences.

B. 2D CONVNETS WITH LSTM

The well-designed structure of recurrent networks is an ideal way for processing sequence data. Therefore, Jeff Donahue et al. proposed a Long-term Recurrent Convolutional Neural Network(LRCN) which adds a recurrent layer(LSTM) to 2D CNNs [20]. The role of LSTM is to process the feature sequences extracted by 2D ConvNets and the LRCN can be applied in a variety of video tasks(action recognition, image description, and video description). Yue-Hei Ng et al. [21] proposed the fusion architecture of Conv Pooling and LSTM to handle full length videos. And the Deep Bi-Directional LSTM(DB-LSTM) is also be used with CNN features to process lengthy videos [22]. This type of model is trained with all the frames of video, and the prediction is made by the last temporal output. The broad receptive field of LSTM can provide the refined temporal structure, which is capable of encoding temporal ordering and long range content. But this requires a hard work for modeling the complicated LSTM.

C. 3D CONVNET ARCHITECTURES

Similar to processing static images with 2D convolution, the 3D convolution can directly extract spatio-temporal...
feature from video data. The C3D [6] adopts eight convolution layers of $3 \times 3 \times 3$ kernel size in the architecture, which is compact and efficient structure to model appearance and motion simultaneously. The parameters of C3D are pre-trained on Sports1M and I380K to pursue better performance. The residual network is also introduced into 3D ConvNets [25], but also requires the pre-training on Sports1M. The study [3] explored the effects of pre-training in different video datasets. The experimental results obviously illustrate that the sufficient scale of pre-trained datasets is as large as Kinetics [1]. The recent researches also attempt to exploit new 3D convolution modules for processing video data. Temporal 3D ConvNets (T3D) [5] imbedding the 3D Temporal Transition Layer (TTL) into 3D DenseBlock layer. The TTL module is composed of three parallel kernels in different size. This renders the model ability of capturing hierarchical spatio-temporal features from video clips. T3D is still pre-trained on Sports-1M then finetuned on HMDB51 and UCF101. The R(2+1)D [8] facilitates 3D convolution block into a spatial 2D convolution($1 \times d \times d$) followed by a temporal 1D convolution($t \times 1 \times 1$). The decomposition of the 3-dimensional convolution enhances the nonlinear relationship between the spatial dimension and the temporal dimension. The R(2+1)D is pre-trained on both Sports-1M and Kinetics, and the results suggest that pre-training in Kinetics can achieve better performance. Another novel study investigated a hybrid design pattern of 3D convolution and 2D convolution, and it is an inspiring study that demonstrate the replacing some bottom 3D convolutions with 2D can achieve improvement in both accuracy and speed [28]. Furthermore, the 3D ConvNet also is able to be applied in the other computer vision tasks. The 3D ConvNet combined with ConvLSTM can be used for Gesture Recognition [29]. The Region Convolutional 3D Network (R-C3D) [30] is a remarkable attempt that combined Faster R-CNN [31] with C3D for activity detection. Such attempt achieves significantly success in activity detection.

Although 3D convolution has an advantage over 2D convolution in modeling spatio-temporal information, it is a tough work to make 3D ConvNet accomplish sufficient pre-training. The reported pre-training in Kinetics need at least 8 GPUs [3]. Our work avoids such expensive pre-training by an effective 2D-Inflated operation, which make 2 GPUs enough to complete the training of 3D ConvNets. We adopt three 2D networks (Resnet101, Resnet152, Densenet169) that widely used in image classification as our basic model for transforming. The following study will express that a generic architecture of video analysis is equally effective for above three networks.

### III. THE PROPOSED APPROACH

In this section, we will introduce our approach in detail. It starts with a general introduction of our model and 2D-Inflated operation. The following part provides a brief presentation of 2D networks adopted in our approach. After that is the description of residual fully connected layer that is responsible for the feature fusion. Finally, we present the particulars of our training strategies.

#### A. THE PARALLEL-BRANCH FEATURE EXTRACTION

As illustrated in Fig. 1, the parallel architecture is designed to be scalable for exploration. We can change the input sequence structure and the number of 3D ConvNet branch in architecture. Each 3D ConvNet branch independently processes the individual video block that is gathered from each equally divided video segment. The deep features extracted by each branch are concatenated in chronological order, then are passed into the residual fully connected layer (Resfc).
Those 3D ConvNet branches can be regarded as the descriptors for local temporal motion information, and the residual fully connected layer is responsible to reasoning the global spatio-temporal characteristic. Additionally, the 3D ConvNet of our framework is one and shares its weights with the other branches. The extraction process of different branches is independent and performed in parallel. The equation (6) also indicates this operation that the parameter updates of all branches are gathered together on one 3D ConvNet. In experiments, we explore five kinds of architectures in order to find the optimal input structure and the appropriate number of 3D ConvNet branch architecture. The detail results are reported in Section IV that 6-nets structure with separated input is an ideal solution in practice.

B. 2D-INFLATED OPERATION

Inspired by the inflated operation proposed by I3D [4], we present the 2D-Inflated operation for bootstrapping the ImageNet pre-trained parameters. Our approach not only create the structure of 3D ConvNets, but also initialize the 3D filters with pre-trained parameters. It is proved to be a low-cost way for constructing effective 3D ConvNets by the following experiments. Let \( k_{m}^{l} \) denotes the pre-trained 2D filter of the \( m^{th} \) channel in the \( l^{th} \) layer. Then \( K_{m}^{l} \) denotes the corresponding filter in 3D ConvNet. The 2D-Inflated operation can be described as:

\[
K_{m}^{l} = [k_{m}^{l}, k_{m}^{l}, k_{m}^{l}] \\
K^{l} = C_{l}(K_{1}^{l}, K_{1}^{l}, \ldots, K_{m-1}^{l})
\]

(1)

where \( K^{l} \) is the 3D kernel of the \( l^{th} \) layer, and \( C_{l} \) denotes the operation of combining all filters into one kernel. Specifically, each 3D filter is composed by three 2D filters. Those 2D filters are duplicated from the same channel of the corresponding layer in ImageNet pre-trained 2D ConvNet. Afterwards, every cubic filter of \( l^{th} \) is aggregated into a 3D kernel. And the pooling kernels can be simply converted from square to cubic.

Nevertheless, our implement of inflated operation is quite different from this in I3D [4] and the study [3]. In our work, we consider the inflated operation more as a domain adaption task which transfers the classification knowledge from static image domain to video sequence domain. This makes differences in the implementation of inflated operation, for instance, the inflated kernel size, the 2D parameters duplication operation, the reduction of temporal resolution and so on. This innovation also indicates that the similarity and balance between temporal dimension and spatial dimension are not necessary in 3D ConvNets. Specifically, Both the I3D [4] and our work aim to benefit from the ImageNet pre-trained 2D ConvNets, but I3D considered the inflated operation as an implicit pre-training process. Besides, I3D believes that 3D convolution filters should have the same response on the ImageNet dataset as 2D convolution. While in our inflated approach, we regard the inflated operation more as a domain adaption task (from image to video), thus the same response is not our pursuit. We treat the 3D ConvNet as a three-dimensional space container that arranges multiple identical 2D ConvNet planes in parallel, and the 2D network is regarded as the slice from shallow to deep directions of 3D network, which can independently perform on each frame.

Moreover, the study [3] only inflates the structure of 2D ConvNets regardless of the reuse of ImageNet trained parameters and aims to demonstrate that the Kinetics pre-training can improve video training performance, just like the performance improvement in image tasks after the ImageNet pre-training process. However, our inflated approach aims to avoid the expensive Kinetics pre-training process, and convert the ImageNet trained parameters to 3D models. From the Table 1, we can see that study [3] just inflates the structure of 3D ConvNets and processes temporal information similar as that in spatial domain. Our approach adopts a different 3D convolutional structure and stride strategy, and it is obvious that our method has a smaller temporal reduction scale. Moreover, we require less GPU resources than study [3] in video training, and our pre-training process is also more economics than study [3]. The more detailed comparison is described in Table 1.

C. CONSTRUCT VIDEO BLOCK FOR BRANCH INPUT

As stated aforementioned, the reduction of temporal resolution in our 3D ConvNets is rather lower than other methods (I3D [4], Study [3]). The video block technology is extremely relevant to this characteristic in our inflated approach. Because of the lower temporal reduction, we can adopt a small quantity of frames in each video block. The detailed process of video block technology is: we divide a video into 6 equal segments, and randomly extract 6 frames from each segment. The sampled 6-frame sequence is called video block in our work, which contains not only spatial information, but also temporal motion information. The 6-frame sequence is the minimum number of frames to meet the information demand for deep layer of our inflated models, and we believe such sequence has enough motion information for representing the corresponding split segment. Stated differently, we both take advantage of the redundancy in segments and also reduce the redundancy in input video blocks. Specifically, when a single video sample is divided into enough video segments, we can see that just a few consecutive frames of the video segment are capable of representing the content of each segment. Meanwhile, the fewer frames we collect from a segment, the less redundancy the input video block will have. So the more segments we divide, the more branches we will introduce, and the less redundancy the video block will have. The detailed results of experimental comparison are provided in Table 2.

D. BASIC NETWORK

1) RESIDUAL NEURAL NETWORK

The residual neural network (Resnet) is first proposed by He et al. [32] and achieves striking success in
image recognition. Instead of directly fitting a desired underlying mapping \( H(x) \), the residual connection recast original mapping into \( F(x) + x \) in order to fit a residual mapping. Such research suggests that an identity skip connection ("shortcut") is the optimal solution [33]. The shortcut connection can be expressed as:

\[
y_{l+1} = x_l + F(x_l, \{W_l\})
\]

\[
x_{l+1} = f(y_l)
\]

where \( x_l \) and \( x_{l+1} \) denote input and output of the \( l \)-th layer, \( F(x_l, \{W_l\}) \) is the residual mapping. \( f \) represents the ReLU function [32]. For deeper network (more than 50 layers), the function \( F \) in the bottleneck is a stack of 3 layers [32].

Our 3D ConvNets (3D Resnet-101, 3D Resnet-152) are similar in spatial convolution structure with their corresponding 2D network. Therefore, the ImageNet pre-trained 2D parameters can be considered as a slice of 3D kernels. We can directly replicate the 2D parameters into 3D along the temporal dimension, which is referred to 2D-Inflated. However, the temporal structure of 3D ConvNets still need to design. The input convolution size is modified into \( 3 \times 3 \times 3 \times 3 \) and the stride in temporal is set as 1. The 3D max-pooling layer is \( 3 \times 3 \times 3 \times 3 \) with the step size is 2. The convolution of building block are \( 1 \times 1 \times 1 \) and \( 3 \times 3 \times 3 \), and the temporal stride is also 1. The temporal length of input video block is reduced by half only through the max-pooling layer.

### 2) DENSELY CONNECTED NETWORK

The dense convolutional network (Densenet) directly connects all layers with each other [34]. In the dense block, each layer receives additional inputs from all preceding layers. Hence, the \( i \)-th layer has \( i \) inputs and its own output is passed into all subsequent layers. The dense connectivity is express as:

\[
x_{l} = H_{I}([x_0, x_1, \ldots, x_{l-1}])
\]

where \([x_0, x_1, \ldots, x_{l-1}]\) refers to the concatenation of the features produced from preceding \( l \) layers. \( H_{I} \) is a composite function that consists of batch normalization (BN), a following ReLU and a \( 3 \times 3 \) convolution. \( x_{l} \) is the output of \( i \)-th layer.

We inflate the Densenet into 3D by a similar way used in 3D Resnet. The input layer is \( 3 \times 3 \times 7 \times 7 \) convolution. The convolution layer in dense block are \( 1 \times 1 \times 1 \) and \( 3 \times 3 \times 3 \). The average pooling layer of transition is \( 1 \times 2 \times 2 \), which only does down-sampling on spatial.

---

**TABLE 1. The comparison of the inflated methods for 3D ConvNets.**

| Basic 3D ConvNet | The Inflated operation used in 3D model [4] | The Inflated operation used in study [3] | 2D-Inflated operation purpose in our work |
|------------------|-------------------------------------------|-------------------------------------------|-------------------------------------------|
| The detailed inflated procedure | This approach inflates all the convolution filters and pooling kernels from \( N \times N \) into \( N \times N \times N \). The stride of the convolution layers remains same in temporal and spatial dimension. The temporal stride of first two max-pooling layer \( (1 \times 2 \times 2) \) is different from the last two max-pooling layer \( (2 \times 2 \times 2) \). | This approach inflates all the convolution filters and pooling kernels from \( N \times N \) into \( N \times N \times N \). The stride of the first convolution layer is \( 1 \times 2 \times 2 \), and the other convolution layers and the only max-pooling layer remain same in temporal and spatial dimension. | Our approach inflates all the convolution filters and pooling kernels from \( N \times N \) into \( 3 \times 3 \times N \). The stride of all convolution is changed to 1 in temporal dimension, but remain same in spatial dimension. The stride of the max-pooling also remain same in both dimension. |
| The comparison in convolution kernels and temporal stride | The kernel size of input convolution layer is \( 7 \times 7 \times 7 \), the other is \( 3 \times 3 \times 3 \), and \( 1 \times 1 \times 1 \). The stride of input layer is \( 2 \times 2 \times 2 \), the other is 1 in both dimension, just same as 2D Inception-V1 model. | The kernel size of input convolution layer is \( 7 \times 7 \times 7 \), the other is \( 3 \times 3 \times 3 \), and \( 1 \times 1 \times 1 \). The stride of input layer is \( 1 \times 2 \times 2 \), the other remain same in both dimension, which means that the stride of first convolution layer of each convolution block is \( 2 \times 2 \times 2 \). | The kernel size of input convolution layer is \( 3 \times 3 \times 7 \), and the padding size is changed to \( 1 \times 3 \times 3 \), the other is \( 3 \times 3 \times 3 \), and \( 1 \times 1 \times 1 \). The temporal stride of all convolution layer is 1, and the spatial stride remain same. |
| The number of Input frames | Consecutive 64 frames | Consecutive 16 frames | 6 separated video blocks of 6 consecutive frames in each, 36 frames. |
| Changes in temporal resolution from shallow to deep | The temporal resolution is reduced by 8 times through the first input convolution layer and the last two max-pooling layer. | The temporal resolution is reduced by 16 times through the only max-pooling layer and three initial convolution layers of each convolution block. | The temporal resolution is reduced by 3 times through the only max-pooling layer. |
| The required computation hardware resources | 64 GPUs for the pre-training in Kinetics, 16 GPUs for the training of HMDB51 and UCF101 | 8 GPUs (TITAN X) for the pre-training in Kinetics and the training of HMDB51 and UCF101. | No need for more video data pre-training. 2 GPUs (TITAN RTX) for training of HMDB51 and UCF101. |

---

VOLUME 8, 2020

45757
to improve the performance. Besides, the residual fully connection can improve the performance and make the training process more stable. The comparison results are presented in the following Section IV. The prediction of action class is produced by another fully connected network(fc). This layer is used as a classifier and the probability of each action is predicted by a softmax function. The final prediction $p$ can be described as:

$$
p = \frac{\exp(y)}{\sum_j \exp(y_j)}$$

(5)

where $M$ denotes the number of action classes, $y$ is the output vector of the classification layer. The loss can back-propagation to all the parallel 3D ConvNets through the shortcut connections, which make it easier to optimize each 3D ConvNet branch by global loss. Formally, we fit the fusion vector $x_t$ through the summation of a global spatio-temporal vector and a concatenated vector of local motion. Thus, the dimension of $x_t$ will increase as more 3D ConvNet branches are used in architecture.

### F. TRAINING STRATEGIES

Since one single video sample in action recognition may contain hundreds of images, the training batch size of video processing will be smaller than image tasks under same hardware condition. Meanwhile, the parameter size of 3D ConvNet is several times of 2D net. This leads to the over-fitting and divergence problem when training 3D nets under the training strategies used for image classification. In this subsection, we explain the accumulated gradient descent method, data augmentation skills, and learning rate adjustment strategy in detail. These methods are together used for solving the above problems.

In the training of deep learning, the stochastic gradient descent(SGD) is the common training algorithm for parameter update in training. However, the video sample contains many image frames, which is difficult to train with large batch size when the computation resources are limited. In our experiments, we adopt the technique of accumulated gradient descent(AGD) which is a common practice in deep learning. In practice, when we use the AGD as the training algorithm, the formula of back-propagation can be described as follows:

$$
\nabla L_{AGD} = \frac{1}{2} \left( \frac{\partial L(p_i, t_i)}{\partial \mathbf{W}} + \frac{\partial L(p_{i-1}, t_{i-1})}{\partial \mathbf{W}} \right)
$$

(6)
where \( p_i, t_i \) are the prediction and the target of the \( i \)th iteration, respectively. \( L \) is the cross-entropy loss function, and 
\[
F(x_{\text{c}}) \text{ is the 3D ConvNet which input the } i \text{th video block. The function } H \text{ denotes the residual fully connection and classification layer. In the equation (6) indicates that the accumulated loss is the averaged loss of two adjacent iterations. The back-propagation of accumulated loss is equal to calculate the averaged gradient tensor of two steps. In experiments, AGD shares the same parameter setting with SGD, but the total count of parameter update reduces to a half of SGD. Although AGD reduces the iteration of weight updates by half, the training process is still stable and can achieve better accuracy. The comparative results of Table 4 provide the convincing evidences for the effectiveness of this method. The more details about the comparisons between AGD and SGD are presented in Section IV.

1) DATA AUGMENTATION

The appropriate data augmentation techniques are necessary for increase the diversity of samples. In training, we use horizontal flipping, multi-scale random cropping, and scale jitting [3] to perform data augmentation. For multi-scale random cropping, we crop the area that defined by the product of the minimum length and scale. The scale is randomly selected from 1.0, 0.875, 0.75, 0.66. Afterwards, all the frames is spatially resized to 224 × 224 pixels. Meanwhile, horizontal flipping operation is performed on each 3 × 224 × 224 input frame with 50% probability. In testing, we just perform image scale transformation and center cropping. We also adopt mean subtraction and variance normalization similar with ImageNet. In temporal dimension, we evenly divide the video sample into \( n \) video segments, and independently collect consecutive \( D \) frames from each segment as input video block. The input video sequence is \( n \times D \) frames in total. However, the experiments suggest that the total number of input frames is no more than 36 in 6-branch architecture. Specifically, if more segments is used, fewer frames should be chosen. For 6-nets parallel 3D ConvNet, we better collect 6 image frames from each video segment.

2) LEARNING RATE ADJUSTMENT

The learning rate adjustment strategy is quite different from the existing strategies of 3D ConvNets. The initial learning rate is set as 0.001. Instead of reducing by every 10 epochs, we decrease the learning rate to 0.78 of the previous epoch. The experiments show that the traditional decline strategy leads to a sustained growth in loss. We suspect that the 2D-Inflated network is more sensitive to large learning rate and the excessive gradient updates directly cause overfitting in training process. Our strategy restrains the overquick parameter update and effectively reduce the transmission of noise from deep to shallow. In addition, we employ a dropout function to against overfitting. Generlly, the dropout ratio is set as 0.2. Although we can increase the dropout ratio to suppress overfitting, the training is affected by higher dropout ratios. In view of this, we adopt an additional strategy that restarts the learning rate after 100 epochs. Concretely, in the training of HMDB51, we commonly increase the learning rate to 0.0001. In the UCF101, we increase the learning rate to the value of between 0.0002 and 0.0003 (commonly 0.0003), as the loss of UCF101 is smaller than it in HMDB51 after 100 epochs. This can awake the training process, and restart valid back-propagation. After the learning rate is reset, the learning rate is reduced to 0.85 of the previous epoch for enhancing training effect.

IV. EXPERIMENTS

In this section, we first briefly introduce the datasets that is used for evaluating our models. Secondly, we perform a serie of comparative experiments for searching optimal architecture. We also prove the effect of accumulated gradient descent and the superiority of residual fully connection in third subsection. The experiments demonstrate that increasing branches can achieve better improvement than increasing frame number. Then, we further report the results of combining iTD feature with three 3D ConvNets(3DResnet101, 3DResnet152, 3DResnet169). Finally, we show the comparison with the state-of-the-art models. All the experiments are implemented on the Pytorch platform.

A. DATASET

As stated above, the benchmark datasets are two commonly used for action recognition, namely HMDB51 and UCF101. HMDB51 contains 6,766 videos from 51 action classes, and UCF101 comprises 13,320 action instances which covers 101 action classes. The videos were temporally trimmed to remove non-action frames, and the average duration of each video is about 7 seconds [3]. Both datasets have three train/test splits (70% training and 30% testing). We first explore our parallel architectures on split 1 of both datasets. Then the proposed architecture is evaluated on three splits to compare with state-of-the-art methods.

B. ARCHITECTURE SEARCHING

In this subsection, we evaluate the performance of five architectures on both datasets to explore the optimal design. The main difference between the various structures is the number of branch networks(or snippets) used in the preceding feature extraction. We start the exploration from a 3DResNet101 with a single branch, which inputs two types of video block(16 frames and 36 frames). Then, we investigate architectures with 2, 4, 6, 9 branches that assigned with different frames in each video block. All the 3DResnet101 is initialized by the 2D-Inflated operation presented in Section III-B, and the dropout rate is set as 0.2. The entire training processes employ the accumulated gradient descent to update the weights of the network. The training parameters include a weight decay of 10\(^{-5}\) and 0.9 for momentum. We choose the batch size according to the maximum value our hardware resources can offer and the model we perform. Besides, the study proposed by Yuxin Wu and...
Y. Huang et al.: Efficient Parallel Inflated 3D Convolution Architecture for Action Recognition

TABLE 2. The accuracy performance of five 3DResnet101 based architectures in HMDB51 and UCF101 under different input conditions (split 1).

| Architecture | Branch Input | Total Input | Params | Batch Size | HMDB51 | UCF101 | GPU |
|---------------|--------------|-------------|--------|------------|--------|--------|-----|
| 1-Net         | 16 frames    | 16 frames   | 89.5M  | 8          | 50.72% | 83.24% | TITAN XP |
| 2-Nets        | 16 frames    | 36 frames   | 8.5M   | 8          | 51.24% | 83.58% | TITAN XP |
| 3-Net         | 16 frames    | 48 frames   | 8.4M   | 8          | 52.10% | -      | TITAN XP + 1080 TI |
| 4-Net         | 16 frames    | 60 frames   | 8.4M   | 12         | 53.50% | 85.22% | TITAN RTX |
| 5-Net         | 16 frames    | 72 frames   | 8.4M   | 12         | 54.51% | -      | 3 TITAN RTX |

FIGURE 3. The loss and accuracy curve of two different architectures on HMDB51((a),(b)) and UCF101((c),(d)).

Kaiming He suggests that a small batch leads to inaccurate estimation of the batch statistics and increases the model error dramatically [36]. Multiple experiments show that our batch size choices (12 in UCF101, 14 in HMDB51) are a suitable solution for a 2 TITAN RTX GPUs platform. Generally, the architectures with less branches demand less computing resource, and we can use larger batch size or less computation hardware to complete the training task. The loss is calculated by the cross-entropy function, and the training process stops after 100 epochs. The learning rate is adjusted according to the strategy described in Section III-F.2. Experimental results are tabulated in Table 2. Experimental results are tabulated in Table 2. Experimental results are tabulated in Table 2. Experimental results are tabulated in Table 2.

As we can see from the results(Table 2 and Figure 3), the 3D ConvNet architecture benefits from the increasing branches only under a particular input structure. In experiments, we maintain the same batch size when we compare the different video block structures in same branch condition in different platform. This kind of settings are used to explore the impact of different number of frames in video block. The experimental evidences show that increasing the number of frames in video block can improve the performance, but there is a limitation for such improvement, and the effect of increasing branches is better than increasing frames. Moreover, increasing frames demands more memory of GPU and training time. The empirical experiments demonstrate that with more branches are involved in the framework, less frames ought to be used in each video block. Although different branch architectures all adopt the same one 3D ConvNet, the processes of parameter update are quite different. The explanation of this improvement can be seen from the Equation-6 that more long-term distinguishing sequences participate in the back-propagation process when more branches are used in the framework. And the counter-intuitive performance degradation caused by the increasing temporal frames is possibly because the redundant frame interferes with the extraction of effective features. Besides, the branch becomes an amplifier of such disturbance. Although it achieves better performance with more branches, this improvement is not infinite. We can conclude from experiments that 6-Nets architecture is a balance choice for video analysis.

C. COMPARATIVE EXPERIMENTS

In this subsection, we conduct a serie of comparative experiments that provide convincing evidences for proving the validity of our training strategies in Section III-F.
First, the experiments compare different input structures that have the same number of frames. Second, we explore the impact on training process of two different gradient descent methods. The other 3D convolution networks are compared in the following subsection. Finally, we combine the iDT descriptor as a complementary method in pursuit of higher accuracy.

1) COMPARISON IN DIFFERENT AGGREGATION CONNECTIONS, INPUT STRUCTURE AND GRADIENT DESCENT METHODS

In this part, we conduct a series of comparison experiments for different aggregation connections, input structure and two kinds of gradient descent methods. Firstly, we compare three kinds of aggregation connections which are proposed in Section III-E. The training processes of these connections are presented in Figure 4, and the accuracy results are presented in Table 3. According to the experimental results, we can see that a further process on combined feature vector is beneficial to improve the performance. The aggregation method by fully connection is 1.24% and 0.64% better than direct connection in HMDB51 and UCF101. Moreover, when we use the modified version of fully connection—the residual fully connection, the performance is further improved and the training process seems to be more stable under the same training condition. Secondly, after the most appropriate architecture is determined in Section IV-B, the optimal input structure is also need to be explored as consecutive frames are commonly used in previous 3D ConvNets. Therefore, we conduct a comparative experiment to investigate the most effective input structure. Meanwhile, different training algorithms are also compared in our experiments. In the experiments, SGD includes a weight decay of 0.001 and 0.9 for momentum and AGD is performed based on the SGD. The parameter setting of AGD is as same as the SGD. The training processes of different training algorithms are shown in Figure 5, while the experimental settings and results are described in Table 4. While according to the regulation of AGD, we will not perform a single optimization step until the iteration reach the number of accumulated step, then the averaged loss of those iterations is back-propagated to the network. According to the results of the Table 4, we can see that the model equipped the separated 36 frames outperforms the consecutive 36 frames by 2.88% on UCF101 and 4.19% on HMDB51. It suggests that the separated structure with multi-time slices information is more conducive to improve performance, which conforms to our intuitive sense. Meanwhile, during the comparison, we find that the performance of AGD is better than training with SGD in both datasets. Moreover, it is worth noting that the performance of SGD with separated input is still better than that of AGD with consecutive input.

2) PERFORMANCE OF DIFFERENT DROPOUT RATES AND OTHER 3D CONVNETS

When the dropout rate is reasonable increased, the performance of network can be improved [35]. Accordingly, we conducted a verification test on 3DResnet101, which is shown in Table 5. The results suggest that dropout rate...
of 0.4 is beneficial to training on HMDB51, but little effect on UCF101. In practice, 3D models inflated from Densenet169 and Resnet152 are trained with dropout rate of 0.4 on HMDB51, 0.2 on UCF101. We also compare the strategies of learning rate in training.

It can be seen from the experiment that restarting the learning rate is beneficial to improve performance. In the training of HMDB51, we commonly increase the learning rate to 0.0001. In UCF101, we increase the learning rate to the value of between 0.0002 and 0.0003 (commonly 0.0003), as the loss of UCF101 is smaller than it in HMDB51 after 100 epochs. The performance of other networks (3DResnet152, 3DDensenet169) are also presented and the deeper nets only obtain a small gain.

### 3) EFFECT OF COMBINING iDT FEATURE

Although integrating three networks (3DResnet101, 3DResnet152, 3DDensenet169) is capable of improving the accuracy, the performance of such ensemble model is not satisfied. In order to chase better performance, we combine iDT feature descriptor with 3DConvNets, just like the same method used in C3D [6]. The results in Table 6 show that any of the iDT combined model can outperform the ensemble model at least 1% on both datasets. This comparison supports the previous conclusion that 3DConvNets and iDT are highly complementary to each other. We further combine ensemble model with iDT feature on three splits. The comparison of each action class on UCF101 is illustrated in Figure 6.

### D. COMPARED WITH STATE-OF-THE-ART

We show the results of our comparison with the previous state-of-the-art methods in Table 7, on UCF101 and HMDB51. The pre-trained datasets of each method are also presented in the table. Our ensemble model (3D-ConvNets) combined with iDT achieves 92.7% on UCF101 and 69.1% on HMDB51, respectively. Such results are the average recognition accuracies over the three splits of UCF101 and HMDB51. Despite our 3D ConvNets are never pre-trained on video datasets (Sports-1M, I380K), our model outperforms C3D ensemble model (3 nets combined with iDT) by 2.3% on UCF101. Compared with 3D ConvNets that conduct pre-training process on Kinetics, our model still achieves a better performance. Such experimental evidences figure out that 3D ConvNet can benefit from not only the structure design of 2D ConvNet, but also the parameters of those pre-trained kernels.

From the Table 7, we can see that our performance in HMDB51 outperforms the other methods, while T3D+TSN result in UCF101 is a bit better than ours. Through the pre-training process in Kinetics and ImageNet, the performance of T3D+TSN [5] is better on UCF101, but our model achieves better accuracy on HMDB51 without any video data pre-training. The possible reasons of this is: First, T3D is pre-trained on Kinetics which is a specialized video dataset for action recognition and the amount of images is about 50 times larger than ImageNet. T3D obtains more priori knowledge from the Kinetics pre-training. Meanwhile, the T3D+TSN approach aims to perform the feature fusion on the T3D features, which is like TSN. It extends the TSN aggregation method that encoding features from 5 non-overlapping clips of each video. This can provide 5 times input information for processing and enhance the comprehension of long-term content, which is far more than our input size. Such technology fusion is quite appropriate for improving performance in UCF101 as videos of this dataset contain more frames than HMDB51. However, the Kinetics pre-training is difficult to achieve for most practical applications, and the T3D+TSN approach trades an increase in processing time (the amount of T3D features) for an improvement in accuracy. Our approach can avoid the costly Kinetics pre-training and demand fewer input frames for processing while realizing close performance in UCF101 and better result in HMDB51.

We note that conducting deep representation learning by 2-stream or 3-stream can achieve more improvement
TABLE 8. The comparison with the best performance models.

| Method | Pre-train | HMDB51 (%) | UCF101 (%) |
|--------|-----------|------------|------------|
| Our Ensemble Model | ImageNet | 69.1 | 92.7 |
| I3D RGB-stream [4] | ImageNet+Kinetics | 74.5 | 95.4 |
| I3D Two-stream [4] | ImageNet+Kinetics | 80.2 | 97.9 |
| S3D-G [28] | ImageNet+Kinetics | 75.9 | 96.8 |
| 2DResNet-152 3-stream (Weighted Average) [19] | ImageNet | 77.45 | 93.47 |
| 2DResNet-152 3-stream (Logistic Regression) [19] | ImageNet | **81.24** | 92.60 |

in performance [19]. The study [19] further adopted the Spatial Pyramid Network (SPyNet) to extract the optical-flow images, which provided an additional stream for prediction. Meanwhile, Kinetics pre-training process is also an effective and useful technique for performance improvement, which is demonstrated by I3D [4], T3D [5] and S3D-G [28]. In Table 8, we compare our approach with the optimal performance of these novel methods, and it can be seen that our model has a certain gap with the best models. The best performance of HMDB51 and UCF101 is 81.24% and 97.9%, respectively. The reasons of this can be explained from two aspects: Firstly, I3D [4] and S3D-G [28] both benefit from the Kinetics video pre-training process. It has been proved by many previous studies that the model transfer strategy from larger datasets is quite effective for improving performance in target dataset. After the Kinetics pre-training process, the performance of I3D RGB-stream achieves a significant improvement in both datasets. Secondly, aggregating an additional stream trained 2DResNet-152 is another solution to improve performance. Utilizing an additional optical-flow domain which is extracted by SPyNet is also quite useful for improving accuracy. Meanwhile, other fusion methods can be used to facilitate the improvement of some datasets. This can be seen from the great improvement in HMDB51 by the 3-stream 2DResNet-152 of Logistic Regression approach.

However, when we follow the study [19] to conduct weighted average, a weighted average scheme of 30% deep models and 70% iDT method decrease the performance in UCF101 and HMDB51 by about 2%. We suspect that the representation learning by the traditional iDT method is not as effective as the deep learning approach. And the results prove that the simple averaged method is the best fusion strategy for our iDT combined model, which also outperforms the simple averaged two-stream 2D Resnet152(Simple Average) of study [19]. Meanwhile, the Kinetics pre-training process is what we aim to avoid in this work. We agree that a video data pre-training is a powerful technique for performance improvement, but it is hard to seek a balance between performance and training speed. This kind of video pre-training is difficult to be achieved in some practical applications. Indeed, the study [19], S3D-G [28] and two-stream I3D [4] are quite enlightening to our future work. While in this work, we mainly aim to demonstrate that there exists a more economical way to construct a 3D ConvNet, and the inflated operation can be conducted under a view of domain adaption task. In the future, we will evaluate if our approach is still effective for the application of two-stream or three-stream framework.

We will provide the source code at https://github.com/ykhuang0817/Inflated_3DConvNets soon.

V. CONCLUSION AND DISCUSSION

We re-examine the problem presented in the first section, “is the parameters of 2D CNN learned from image domain not useful for 3D CNN to perform video tasks?”.
Our experimental evidences confirm that 3D ConvNets for action recognition can also benefit from the Imagenet pre-trained 2D ConvNets, just like expanding a cube from a square. It should be stated that a video data pre-training of 3D ConvNets is still a generic and effective way for improving performance. But the disadvantage is also obvious that the process demands massive video data, far larger than pre-training in static images. Our experiments demonstrate the effectiveness of 2D-Inflated parameters in action recognition, which avoids pre-training 3D ConvNet on large scale video dataset. Moreover, we regard the inflated operation as a domain adaption task, rather than an implicit Imagenet pre-trained task for 3D ConvNets. An implication of this study is the possibility that the 3D ConvNet used in other video understanding tasks can be constructed in a similar way.

Another noteworthy aspect is the input structure of video sequences and the parallel architecture for video analysis. We explore this problem by changing the 3D ConvNet branches and the input structure. The experimental results suggest that input structure of separated video blocks outperforms the traditional consecutive one. We confirm that 6-nets parallel architecture with separated input structure is an optimal solution for action recognition. We also show the improvement of AGD in our experiments. The insights gained from this study may be of assistance to construct 3D ConvNet in other video analysis tasks. In the further work, we will investigate the effectiveness of applying our inflated 3D ConvNets into 2-stream or 3-stream architecture.

REFERENCES

[1] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillicier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman, “The kinetics human action video dataset,” 2017, arXiv:1705.06950. [Online]. Available: http://arxiv.org/abs/1705.06950

[2] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, “Large-scale video classification with convolutional neural networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2014, pp. 1725–1732.

[3] K. Harai, H. Kataoka, and Y. Satoh, “Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and ImageNet?” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 18–22.

[4] J. Carreira and A. Zisserman, “Quo vadis, action recognition? A new model and the kinetics dataset,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 4724–4733.

[5] A. Diba, M. Fawzyaz, V. Sharma, A. Hossein Karami, M. Mahdi Arzani, R. Yousefzadeh, and L. Van Gool, “Temporal 3D ConvNets: New architecture and transfer learning for video classification,” 2017, arXiv:1711.08200. [Online]. Available: http://arxiv.org/abs/1711.08200

[6] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning spatiotemporal features with 3D convolutional networks,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 4489–4497.

[7] Z. Qiu, T. Yao, and T. Mei, “Learning spatio-temporal representation with pseudo-3D residual networks,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 5533–5541.

[8] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, “A closer look at spatiotemporal convolutions for action recognition,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 6450–6459.

[9] C. Feichtenhofer, A. Pinz, and A. Zisserman, “Convolutional two-stream network fusion for video action recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 1933–1941.

[10] L. Wang, Y. Qiao, and X. Tang, “Action recognition with trajectory-pooled deep-convolutional descriptors,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 4305–4314.
[34] G. Huang, Z. Liu, L. V. D. Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.

[35] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[36] Y. Wu and K. He, “Group normalization,” in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 3–19.

[37] L. Sun, K. Jia, D.-Y. Yeung, and B. E. Shi, “Human action recognition using factorized spatio-temporal convolutional networks,” in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 4597–4605.

**YUKUN HUANG** received the bachelor’s degree from Chongqing University, in 2012, where he is currently pursuing the Ph.D. degree with the College of Optoelectronic Engineering under the supervision of Prof. C. Gao and Prof. Y. Guo. His research interests include action recognition and deep learning.

**YONGCAI GUO** received the Ph.D. degree in instrument science and technology from Chongqing University, Chongqing, China, in 1999. She is currently a Professor and the Dean of the College of Optoelectronic Engineering, Chongqing University. Her research interests include digital signal processing, machine learning, optoelectronic technology, and instrument engineering.

**CHAO GAO** received the Ph.D. degree in instrument science and technology from Chongqing University, Chongqing, China, in 2004. He is currently a Professor and the Ph.D. Supervisor of Chongqing University. His research interests are in the areas of signal processing and computer vision, including image processing, video understanding, optical measurement, and control technology.

* * *