MLP-3D: A MLP-like 3D Architecture with Grouped Time Mixing

Zhaofan Qiu[†], Ting Yao[†], Chong-Wah Ngo[‡] and Tao Mei[†]
[†] JD Explore Academy, Beijing, China [‡] Singapore Management University, Singapore
{zhaofanqiu, tingyao.ustc}@gmail.com, cwngo@smu.edu.sg, tmei@jd.com

Abstract

Convolutional Neural Networks (CNNs) have been regarded as the go-to models for visual recognition. More recently, convolution-free networks, based on multi-head self-attention (MSA) or multi-layer perceptrons (MLPs), become more and more popular. Nevertheless, it is not trivial when utilizing these newly-minted networks for video recognition due to the large variations and complexities in video data. In this paper, we present MLP-3D networks, a novel MLP-like 3D architecture for video recognition. Specifically, the architecture consists of MLP-3D blocks, where each block contains one MLP applied across tokens (i.e., token-mixing MLP) and one MLP applied independently to each token (i.e., channel MLP). By deriving the novel grouped time mixing (GTM) operations, we equip the basic token-mixing MLP with the ability of temporal modeling. GTM divides the input tokens into several temporal groups and linearly maps the tokens in each group with the shared projection matrix. Furthermore, we devise several variants of GTM with different grouping strategies, and compose each variant in different blocks of MLP-3D network by greedy architecture search. Without the dependence on convolutions or attention mechanisms, our MLP-3D networks achieves 68.5%/81.4% top-1 accuracy on Something-Something V2 and Kinetics-400 datasets, respectively. Despite with fewer computations, the results are comparable to state-of-the-art widely-used 3D CNNs and video transformers.

1. Introduction

During the past decade, the advances in Convolutional Neural Networks (CNNs) have successfully pushed the limits and improved the state-of-the-art technologies for image and video understanding [5, 11, 12, 14, 19, 25, 28, 30, 45, 46, 48–51, 57, 64, 69]. Besides achieving the top performances across tasks, the highly optimized implementation of convolution on various hardware makes CNNs continue to dominate computer vision research. Nevertheless, motivated by the success of attention model in Natural Language Processing (NLP) [60], vision transformers [9, 21, 33, 38, 55, 63] become an alternative choice for image recognition by using multi-head self-attention (MSA) and multi-layer perceptrons (MLPs). More recently, the models solely with MLPs (i.e., MLP-like networks) without convolutions or self-attention layers are also shown able to perform well on ImageNet classification, and are more efficient for both training and inference [6, 17, 54, 70].

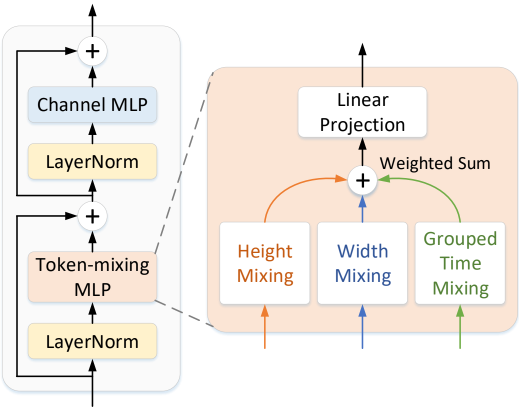


Figure 1. A schematic diagram of MLP-3D block. The block originates from the MLP-mixer layer in [54], and decomposes the original token-mixing MLP into three sub-operations along height, width and time dimensions, respectively. For time dimension, the novel grouped time mixing operation is exploited.

Despite having these impressive progresses on image recognition, devising MLP-like architecture on video data is seldom studied and remains challenging. The video data is more complex due to large variations in motion and rich content in visual details. Capturing useful information from such information-intensive media requires exhaustive computing resources. This property inherently poses difficulties for developing MLP-like 3D architecture from two aspects: 1) how to capture the complex temporal dynamics in videos via MLP-like operations? 2) how to reduce the expensive computations for space-time modeling?

To address the issues, in this work, we start from the design of basic MLP-style operations to model temporal sequence, and next study how to construct an efficient MLP-like architecture on the utilization of these operations. To this end, we propose MLP-3D networks - a novel MLP-like 3D architecture to model spatio-temporal dependency in videos. In MLP-3D networks, an input video clip is divided into overlapped tubelets (i.e., sequences of associated frame patches across time), and each tubelet is mapped into a visual token through a tubelet embedding layer. These tokens are then fed into several stacked MLP-3D blocks, where each block abstracts inter-token information by token-mixing MLP and intra-token information by channel MLP, as shown in Figure 1. The channel MLP, which shares the similar structure as the feed-forward layer in transformer [60], is applied to each token independently. The token-mixing MLP is the weighted summation of three sub-operations applied across different tokens along the height, width and time dimension, respectively. The sub-operations on the spatial dimensions (height and width) follow the recipe of Cycle Fully-Connected Layer (Cycle FC) in [6]. For the time dimension, we devise a novel Grouped Time Mixing (GTM) operation, i.e., a group-based token mixing operation across tokens at different time points. By only mixing the information within each group independently, the computational complexity and the number of parameters are effectively reduced. Furthermore, based on different grouping strategies, we derive four variants of GTM operations and compose each in different MLP-3D blocks by greedy architecture search.

The main contributions of this work are summarized as follows. First, GTM is a novel family of MLP-style operations to model temporal dynamics in an economic and effective way. Second, MLP-3D networks is a new MLP-like 3D architecture by utilizing GTM operations in the decomposed token-mixing MLP. Extensive experiments conducted on Something-Something and Kinetics datasets demonstrate that MLP-3D networks achieve superior or comparable performances to widely-used 3D CNNs (e.g., SlowFast networks [12]) and computationally expensive video transformers (e.g., TimeSformer [3]). Moreover, MLP-3D networks show a great potential in developing the MLP-like architecture for video understanding.

2. Related Work

We group the related works into two categories: deep neural networks for image recognition and video recognition. The first category reviews the research in network design for image classification, and the second surveys a variety of video recognition models.

Image Recognition has received intensive research attention particularly thanks to the success of CNNs in achieving remarkable performance on several benchmarks. Significant amount of efforts are devoted to optimize CNN architectures by hand-tuning [14,18,19,22,25,51,53,67,72]. Later, to automate the design of CNN architectures with less manual intervention, researchers have presented various Network Architecture Search (NAS) approaches, including the proposals of reinforcement learning [35,44,75], architecture evolution [36,40], and differentiable architecture search [7,29,37].

Inspired by the recent advances of attention mechanism [60] in NLP domain, transformer has led to a series of breakthroughs in computer vision area. The pure transformer architectures [9,21,38,63] and the combinations of convolutions and transformers [27,33,52,56,66] become formidable competitors to CNNs. More recently, MLP-based models [6,17,54,70] are built without convolutions or attention mechanisms. Instead, MLP layers are leveraged to aggregate the spatial context over patches.

The current defacto standard for **Video Recognition** is 3D CNNs with 3D convolutions across space and time dimensions. In one of the early works [23], a 3D CNN extending directly from image-based 2D CNN is devised to recognize actions in video clip via 3D convolution. Later, there have been several attempts to improve 3D CNNs. For example, C3D [57], by stacking several 3D convolutions and 3D poolings, demonstrates a state-of-the-art pretraining model for video understanding at the time. I3D [5], which pre-trains an Inception-style network [53] on large-scale dataset, enables the extremely high fine-tuning performances on small datasets. SlowFast networks [12] builds a two-path architecture consisting of a slow path with high sampling rate and a fast path with low sampling rate. In parallel with these architecture design works, the model complexity of 3D CNNs has been reduced by 3D kernel decomposition [46,59,68] and depth-wise 3D convolutions [11,58]. More recently, the transformer-based architectures become a new trend of convolution-free networks on video data [1,3,10,39,41,43,71,73].

Our work also falls into the category of convolution-free architecture for video recognition. Unlike the transformers with attention mechanisms, the token interactions in MLP-3D networks are more efficiently accomplished by fully-connected layers. Moreover, MLP-3D networks expands the research horizons of MLP-like networks to video recognition, and uniquely studies the efficient way of temporal modeling in MLP-like architecture.

3. Our Method

3.1. Overall Architecture

Figure 2 depicts an overview of the proposed MLP-3D Networks. The basic architecture follows the philosophy of CNNs [14,51], where the channel dimension increases while the spatial resolution shrinks with the layer going deeper. The similar design is also exploited in hierarchical transformers [38,63] and MLP-based models [6,70].

Tubelet embedding. Given a video clip with the size of $H \times W \times T \times 3$, where H, W and T denotes the height,

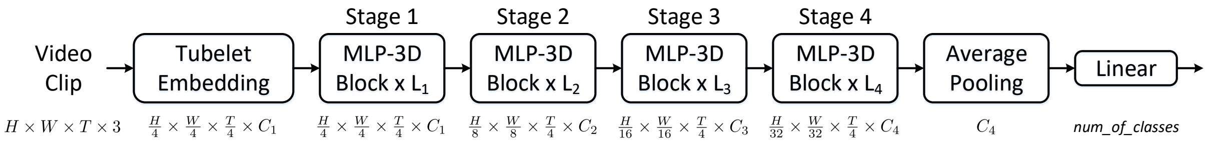

Stage 1
Stage 2
Stage 3
Stage 4
Video Clip
Tubelet Embedding
MLP-3D Block x L1
MLP-3D Block x L2
MLP-3D Block x L3
MLP-3D Block x L4
Average Pooling
Linear
$H \times W \times T \times 3$
$\frac{H}{4} \times \frac{W}{4} \times \frac{T}{4} \times C_1$
$\frac{H}{4} \times \frac{W}{4} \times \frac{T}{4} \times C_1$
$\frac{H}{8} \times \frac{W}{8} \times \frac{T}{4} \times C_2$
$\frac{H}{16} \times \frac{W}{16} \times \frac{T}{4} \times C_3$
$\frac{H}{32} \times \frac{W}{32} \times \frac{T}{4} \times C_4$
C_4
num_of_classes

Figure 2. An overview of our proposed MLP-3D networks. C_i and L_i denotes the number of output channels and the repeated number of MLP-3D blocks in the i-th stage, respectively. The size of output feature map is also given for each block.

width and clip length, respectively, our models first embed the overlapped tubelets with window size $7 \times 7 \times 4$ and stride $4 \times 4 \times 4$. Each tubelet is mapped into a visual token with a higher dimension C_1 by using a shared linear embedding layer. As such, the overall tubelet embedding module produces the features with the shape of $\frac{H}{4} \times \frac{W}{4} \times \frac{T}{4} \times C_1$.

Multi-stage architecture. Then, the sequential MLP-3D blocks illustrated in Figure 2 are applied to the tubelet tokens. The whole MLP-3D network includes four stages, and the feature resolution is maintained within each stage. A stage transition is inserted between two adjacent stages, which increases the number of channels and reduces the spatial resolution. In this way, the number of tokens from the last stage is $\frac{H}{32} \times \frac{W}{32} \times \frac{T}{4}$. The resultant tokens are finally averaged along the space and time dimensions, followed by a fully-connected layer for class prediction.

3.2. MLP-3D Block

The proposed MLP-3D block originates from the MLP-based block in MLP-Mixer [54], which replaces the multi-head self-attention module in transformer by a token-mixing MLP. In detail, MLP-based block consists of two components: channel-MLP and token-mixing MLP. **Channel-MLP** utilizes the similar structure of the feed-forward layer in transformer [60], which contains two linear layers plus a GELU [15] non-linearity in between. **Token-mixing MLP** mixes the information from tokens on different spatial/temporal positions, and characterizes the primary difference among various MLP-based models [6, 17, 54, 70]. Specifically, given the input tokens $\boldsymbol{X}$, the function of MLP-based block can be formulated as

$$\begin{aligned} \boldsymbol{Y} &= \text{Token-mixing-MLP}(\text{LN}(\boldsymbol{X})) + \boldsymbol{X}, \\ \boldsymbol{Z} &= \text{Channel-MLP}(\text{LN}(\boldsymbol{Y})) + \boldsymbol{Y}, \end{aligned} \tag{1}$$

where LN denotes Layer Norm [2]. The output $\boldsymbol{Z}$ serves as the input to the next block until the last one.

Decomposing token mixing. The goal of token-mixing MLP is to capture spatial/temporal patterns by mixing the information of different tokens. Inspired by Vision Permutator [17], MLP-3D block decomposes the token-mixing MLP and encodes the information along one axis at one time. By doing so, the token-mixing MLP can capture long-range dependencies along one dimension and meanwhile preserve precise positional information along the other dimensions. Different from [17] which decomposes the operation by height, width and channel dimensions for image recognition, MLP-3D block chooses the mixing along time axis instead of channel axis for video data. Such design shares the similar spirit as in 3D convolution decomposition [46, 59, 68] and space-time divided attention [1, 3].

Concretely, the output of decomposed token mixing $\hat{\boldsymbol{Y}}$ is calculated by linearly projecting the summation of token mixing along three dimensions:

$$\hat{\boldsymbol{Y}} = \text{FC}(\boldsymbol{X}_H + \boldsymbol{X}_W + \boldsymbol{X}_T), \tag{2}$$

where $\boldsymbol{X}_H$, $\boldsymbol{X}_W$ and $\boldsymbol{X}_T$ are the outputs of height, width and time mixing, respectively. FC denotes a fully-connected layer. Here, we utilize the weighted summation proposed in [17] to aggregate the outputs of different mixing operations. For height/width mixing operation, we choose Cycle FC in [6], which has been proven to be effective on capturing spatial context.

3.3. Grouped Time Mixing (GTM)

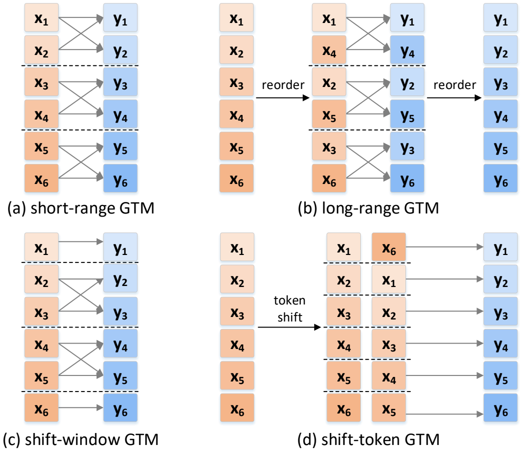
To further improve the efficiency of token-mixing MLP, we propose a novel Grouped Time Mixing (GTM) operation to produce $\boldsymbol{X}_T$ in Eq. (2) by fusing the inter-token information in a grouped manner along time dimension. Formally, we start by analyzing the simplest time mixing, which linearly maps the features of all tokens in different time points, called full time mixing. More specifically, given the reshaped input tokens as $\hat{\boldsymbol{X}} \in \mathbb{R}^{HW \times TC}$, the output of full time mixing is computed as

$$\boldsymbol{X}_T = \hat{\boldsymbol{X}} \cdot \boldsymbol{W}, \tag{3}$$

where $\boldsymbol{W} \in \mathbb{R}^{TC \times TC}$ is the projection matrix. Although the operation can capture large-range dependency along the time axis, it demands geometrical progression of computational complexity $\mathcal{O}(HWT^2C^2)$ and the number of parameters $\mathcal{O}(T^2C^2)$ with the increase of clip length T.

To alleviate this limitation, we devise the Grouped Time Mixing operation, which divides the input tokens into several temporal groups and maps the tokens in each group with the shared projection parameters. As such, the computational complexity and the number of parameters are reduced because the group size is usually much smaller than the clip length. To materialize this idea, we derive four different GTM operations as depicted in Figure 3, which correspond to different constructions of token groups. We detail the comparisons on the operations as follows:

(1) Short-range GTM. The first design evenly separates the tokens into $\frac{T}{S}$ groups, where S is the group size (i.e., the



(a) short-range GTM

(b) long-range GTM

(c) shift-window GTM

(d) shift-token GTM

Figure 3. The illustration of four Grouped Time Mixing (GTM) operations. Each rectangle represents the input ($\boldsymbol{x}_i$) or output ($\boldsymbol{y}_i$) tokens at time point i. The group size is set to 2 as an example.

number of tokens in each group). For each group, the consecutive S tokens are linearly mapped by a shared matrix $\boldsymbol{W}_S \in \mathbb{R}^{SC \times SC}$. In other words, the short-range GTM is equivalent to making the matrix $\boldsymbol{W}$ in Eq. (3) be sparse:

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{W}_S & 0 & \cdots & 0 \\ 0 & \boldsymbol{W}_S & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{W}_S \end{bmatrix}, \tag{4}$$

in which only the values in the diagonal blocks are nonzero. As a result, the computational cost and the number of parameters is reduced to $\mathcal{O}(HWTSC^2)$ and $\mathcal{O}(S^2C^2)$, respectively. This operation is ideally similar to the window-based self-attention [21, 38] but remould for time mixing.

(2) Long-range GTM. The second design extends the first one by having an interval of $\frac{T}{S}$ time step between two consecutive tokens in each group. Such group captures long-term dependency in a video while disregarding the local patterns across adjacent frames, which is complementary to short-range GTM. The long-range GTM can be simply implemented by reordering the tokens before and after short-range GTM as shown in Figure 3(b).

(3) Shift-window GTM. The third design is a complementary operation to short-range GTM, namely shift-window GTM. The downside of solely utilizing short-range GTM is the lack of connection across groups. Inspired by the recent successes of shifted window self-attention [38], we shift the partition of groups in short-range GTM by an offset of $\frac{S}{2}$. Thus, alternately applying short-range GTM and shift-window GTM across different blocks in a network provides an efficient way of interaction across groups.

(4) Shift-token GTM. Different from the others, the last design of shift-token GTM forms groups through shifting tokens. Specifically, with the group size S, each token is grouped with another $S-1$ ones, each of which reaches the reference position via shifting $1, 2, ..., S-1$ time steps in a circular manner. Figure 3(d) showcases an example where a token is concatenated with another token circularly shifted by one time step for linear mapping. Please note that the number of parameters of shift-token GTM is $\mathcal{O}(SC^2)$, which is less than other GTM operations.

Algorithm 1 Codes for Grouped Time Mixing (PyTorch-like)

```
# x: input tensor of shape (H, W, T, C)
# ty: mixing type, S: group size

if ty == 'shift_token':
  self.linear = nn.Linear(S*C, C)
else:
  self.linear = nn.Linear(S*C, S*C)

def grouped_time_mixing(x):
  if ty == 'short_range':
    x = self.linear(x.reshape(H, W, -1, S*C))
    x = x.reshape(H, W, T, C)
  elif ty == 'long_range':
    x = x.reshape(H, W, S, -1, C).transpose(2, 3)
    x = self.linear(x.reshape(H, W, -1, S*C))
    x = x.reshape(H, W, -1, S, C).transpose(2, 3)
    x = x.reshape(H, W, T, C)
  elif ty == 'shift_window'
    x = shift(x, S//2)
    x = self.linear(x.reshape(H, W, -1, S*C))
    x = shift(x.reshape(H, W, T, C), -S//2)
  elif ty == 'shift_token':
    x = [shift(x, i) for i in range(S)]
    x = self.linear(torch.cat(x, dim=3))
  return x
```

Reducing parameters by weight sharing. As per our discussion above, the number of parameters of the first three GTM operations is $\mathcal{O}(S^2C^2)$ which grows fast with the increase of group size. Here, we propose to further reduce the number of parameters by sharing the projection weights between the tokens with the same time interval. Formally, the matrix $\boldsymbol{W}_S$ in Eq. (4) with weight sharing can be rewritten as

$$\boldsymbol{W}_S = \begin{bmatrix} \boldsymbol{w}_0 & \boldsymbol{w}_1 & \cdots & \boldsymbol{w}_{S-1} \\ \boldsymbol{w}_{-1} & \boldsymbol{w}_0 & \cdots & \boldsymbol{w}_{S-2} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{w}_{-S+1} & \boldsymbol{w}_{-S+2} & \cdots & \boldsymbol{w}_0 \end{bmatrix}, \tag{5}$$

where $\boldsymbol{w}_{\Delta t} \in \mathbb{R}^{C \times C}$ is the projection matrix between two tokens with Δt time interval. The number of parameters is thus reduced to $\mathcal{O}((S \times 2 - 1)C^2)$.

Implementation. The proposed GTM operations can be readily implemented with a few lines of codes in Python. We provide an example of the codes in Algorithm 1 based on PyTorch [42] platform. We construct the groups of tokens by calling the default *reshape*, *transpose*, *cat* and a pre-defined *shift* functions. We execute the linear mapping by the default *Linear* layer.

3.4. The MLP-3D Networks

In order to verify the merit of the four GTM operations, we first develop several MLP-3D network variants based on the 10-layer CycleMLP (CycleMLP-B1) [6] by replacing all the basic blocks with MLP-3D blocks which involve one certain type or two complementary types of GTM operations. Specifically, the MLP-3D network variants with a single type of GTM operations, i.e., **MLP-3D-SR**, **MLP-3D-LR** and **MLP-3D-ST**, solely utilize short-range, long-range and shift-token GTM operation, respectively. Please note that shift-window GTM is theoretically equivalent to short-range GTM when using a single type of GTM. For the variants with two mixed types of GTM operations, we exploit short-range GTM and long-range/shift-window GTM in turn for different blocks, called **MLP-3D-SR-LR/MLP-3D-SR-SW**, respectively. The comparisons of performance and computational cost between the basic CycleMLP-B1 and the five MLP-3D variants are discussed. Then, based on the observations from these comparisons, an oracle version of MLP-3D network is proposed for the optimal arrangement of GTM operations by a greedy search.

Comparisons between MLP-3D network variants. The comparisons are conducted on Something-Something V2 (SS-V2) [13] dataset that is related to human-object interaction scenario and requires a precise modeling of temporal evolution. The dimension of the input video clip is set as $64 \times 128 \times 128$ which contains randomly cropped 128×128 patches from the uniformly sampled 64 frames. For the videos with less than 64 frames, all the frames are repeated until obtaining enough frames. For each architecture, the weights are initialized with ImageNet-1K pre-trained CycleMLP-B1 model, and an extra dropout layer with 0.5 dropout rate is added before the final fully-connected layer. In the training stage, following [10,39], we exploit label smoothing, random augment [8], random erasing [74] and drop path [20] to reduce the over-fitting effect. We set each mini-batch as 512 clips, which are implemented with multiple GPUs in parallel. The network parameters are optimized by AdamW optimizer with basic learning rate of 0.0005 and weight decay of 0.05. The learning rate has one-epoch warmup and then is annealed down to zero after 32 epochs following a cosine decay.

Figure 4 compares the performances and computations of MLP-3D network variants on SS-V2. Overall, all the MLP-3D network variants exhibit higher performance by a large margin than utilizing 2D CycleMLP-B1 on each frame independently. Specifically, among the variants with a single type of GTM operation, MLP-3D-ST with shift-token GTM achieves the best top-1 accuracy across different group size S. For variants with two mixed types of GTM operations, both MLP-3D-SR-LR and MLP-3D-SR-SW attain higher accuracy against solely using short-range GTM (MLP-3D-SR) or long-range GTM (MLP-3D-LR). The result basically indicates the advantage of combining different types of GTM. Moreover, for each MLP-3D network variant, the accuracy curve is like the "Λ" shape when S varies from 1 to 16. This implies that larger group size will not always lead to higher performance, and proper sparsity constraint in GTM might benefit the network learning.

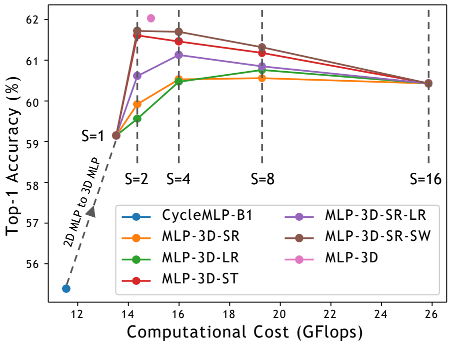


Figure 4. Comparisons of different MLP-3D network variants in terms of computational cost and top-1 accuracy on SS-V2 dataset.

MLP-3D network architecture. Based on the empirical findings, the MLP-3D architecture can be promoted with 1) different GTM operations in different blocks; 2) carefully chosen group sizes for GTM operations; 3) proper trade-off between accuracy and computational complexity. In order to optimize these designs, we propose an efficient greedy search algorithm to determine the MLP-3D network architecture, i.e., the type of GTM operation in each block and the corresponding group size. Particularly, inspired by the high efficiency of weight sharing NAS [4, 40, 44], we split the architecture search into two steps: 1) pre-training the shared weights with randomly assigned types and group sizes; 2) gradually searching the architecture with the best evaluation accuracy with regard to the pre-trained weights. For the first step, we randomly assign the types and group sizes of GTM at each iteration, and a set of time-interval-based matrices $\{\boldsymbol{w}_{\Delta t}|_{-S_{max}+1\leqslant\Delta t\leqslant S_{max}-1}\}$ used in Eq. (5) is shared, where S_{max} is the maximum possible group size. For the second step, the pre-trained weights are used to evaluate each architecture without additional training. In other words, given an architecture, the performance can be approximately estimated by only inferring on validation set with the shared weights. Nevertheless, it is still time-consuming to evaluate all the candidate architectures. To further reduce the time cost of architecture search, we propose to gradually determine the GTM operation of each block one-by-one. An example of the greedy search process is given in Figure 5. At the beginning of architecture search, all the operations are randomly assigned at each forward. Then, the operation of each block is decided in turn by choosing the best-performing operation. We repeat the search process three times for more consistent results. In addition, when comparing the performances of different architectures, we further consider the computational com-

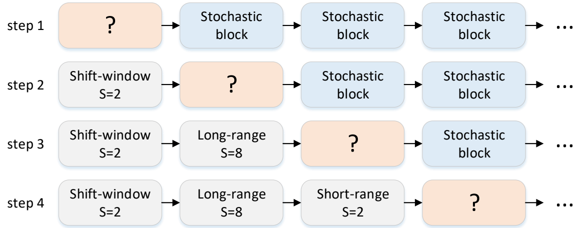


Figure 5. An example of the greedy search process to determine the MLP-3D network architecture.

plexity to approach a good balance. Specifically, given an architecture $\boldsymbol{\theta}$, the revised performance with the consideration of computations is given by

$$\mathcal{V}(\boldsymbol{\theta}) - \alpha\mathcal{C}(\boldsymbol{\theta}), \tag{6}$$

where $\mathcal{V}(\cdot)$ and $\mathcal{C}(\cdot)$ denotes the validation accuracy and computational complexity, respectively. We set the trade-off hyper-parameters α as $5\mathrm{e}^{-3}$ by default. Note that the very specific MLP-3D network in Figure 4 is greedily sought by this algorithm.

4. Experiments

We empirically evaluate our MLP-3D networks on three challenging action recognition benchmarks: Something-Something V1&V2 [13] and Kinetics-400 [5].

4.1. Datasets

Something-Something is a large-scale video dataset that focuses on human-object interaction scenario. The average video length is 4.0 seconds and all videos are captured from object-centric view with fairly clean backgrounds. The dataset contains 174 fine-grained categories of human-object interactions. The differentiation between the similar interactions is very challenging, which requires the understanding of cause-and-effect relationship in videos, e.g., “Pushing something so that it falls off the table” and “Pushing something so that it almost falls off but doesn’t.” The first version (SS-V1) of the dataset contains 108K videos, which are divided into 86K, 11K, 11K for training, validation and test sets, respectively. The extended version (SS-V2) further increases the video number to 220K, which are partitioned into 170K, 25K and 25K for training, validation and test sets, respectively.

Kinetics-400 (K-400) is a standard large-scale benchmark for video recognition, covering 400 action classes. It consists of 246K training videos, 20K validation videos and 40K test videos. Each video in the dataset is 10-second short clip trimmed from the raw YouTube video. K-400 lays particular emphasis on the visual details of objects and background instead of temporal evolution, and is usually treated as a complement to Something-Something dataset.

Table 1. MLP-3D networks with various complexity.

Network	C_1, C_2, C_3, C_4	L_1, L_2, L_3, L_4
MLP-3D-XS	64, 128, 320, 512	2, 2, 4, 2
MLP-3D-S		2, 3, 10, 3
MLP-3D-M		3, 4, 18, 3
MLP-3D-L	96, 192, 384, 768	3, 4, 24, 3

Note that the labels for test sets are not publicly available, and thus the performances of SS-V1, SS-V2 and Kinetics-400 are all reported on the validation set.

4.2. Implementation Details

MLP-3D Networks. We build a family of MLP-3D networks with various model complexities, as detailed in Table 1. C_i and L_i denote the number of output channels and the repeated number of MLP-3D block in the i-th stage, respectively. These settings are considered as free parameters to make the network structure tailored to the scale of video recognition problem. Here, we exploit the free parameters of CycleMLP-B1, CycleMLP-B2, CycleMLP-B3 and CycleMLP-B5 in [6], and build a series of MLP-3D networks, namely MLP-3D-XS, MLP-3D-S, MLP-3D-M and MLP-3D-L, respectively.

Training stage. The training strategies of searching and evaluating MLP-3D network variants have been described in Section 3.4. After determining the architecture, we retrain the MLP-3D network with a similar strategy except for larger input resolution and batch augmentation [16], which leads to longer training time and higher performances. In addition, considering that different GTM operations can share the same weights, we propose a novel regularization that randomly changes the type and group size of GTM to improve the generalization ability of MLP-3D networks.

Weights initialization. As introduced in Section 3.4, the weights of MLP-3D networks are initialized with the ImageNet-1K pre-trained CycleMLP models. In order to maintain the semantic information of pre-trained models, center initialization in [1] is exploited. The idea is to copy the weights of pre-trained 2D patch embedding to the center of 3D tubelet embedding matrix. Similarly, the projection matrix $\boldsymbol{w}_0$ between the input and output tokens at the same time point is initialized by channel mixing operation in CycleMLP, and the other matrices $\boldsymbol{w}_{\triangle t}$ are set as zero. Such initializations make MLP-3D network perform like a 2D network when the training proceeds.

Inference stage. During inference, we evenly sample one/four clip(s) from each test video of Something-Something/Kinetics-400, respectively. We extract the prediction of each clip by using the three-crop strategy as in [12], which crops three square patches. The video-level prediction is obtained by averaging scores from all the clips.

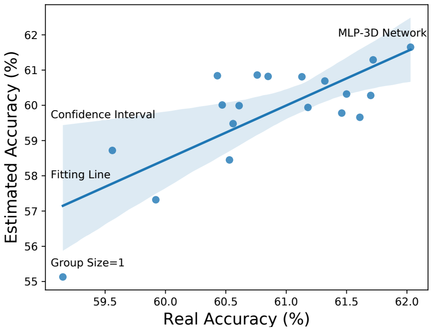
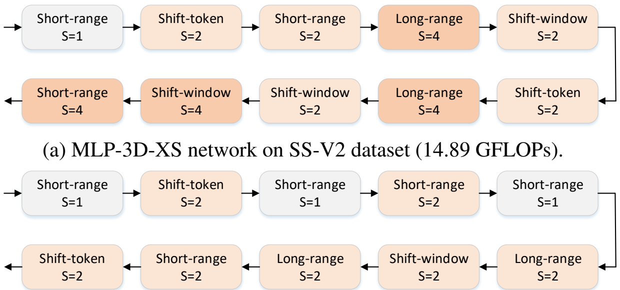


Figure 6. The visualization of real accuracy and estimated accuracy of MLP-3D-XS network variants. The fitting line of two accuracies and its confidence interval are also depicted.



(a) MLP-3D-XS network on SS-V2 dataset (14.89 GFLOPs).

(b) MLP-3D-XS network on Kinetics-400 dataset (14.11 GFLOPS).

Figure 7. The MLP-3D-XS network greedily sought on (a) SS-V2 dataset, and (b) Kinetics-400 dataset, respectively. The blocks with different group sizes are in the different colors.

4.3. Experimental Analysis of MLP-3D Networks

We firstly analyze the greedy search of MLP-3D networks. Figure 6 reveals the correlation between real accuracy and estimated accuracy using shared weights of MLP-3D-XS network variants in Section 3.4. The real accuracy is achieved by training each architecture individually. The estimated accuracy is obtained without training by directly utilizing the pre-trained shared weights. A positive correlation is identified between these two accuracies, as shown in the figure. The result basically validates the use of estimated accuracy as an efficient approximation of real accuracy in the architecture search.

Figure 7 depicts the MLP-3D-XS networks greedily sought on SS-V2 and Kinetics-400 datasets, respectively. The optimal type of GTM and group size are given for each block. An interesting observation is that, with the same constraint of computations in Eq. (6), the network complexity searched on Kinetics-400 dataset is less than that on SS-V2 dataset. This reasonably meets our expectation since the videos in SS-V2 are known to be more complex in terms of temporal dynamics than those in Kinetics-400 dataset.

Next, we study how each design in MLP-3D networks influences the overall performance. Here, we re-train the greedily sought MLP-3D networks, i.e., from MLP-3D-XS to MLP-3D-L, with different input size (128^2 or 224^2). In order to explore the effectiveness of two-path networks following [10, 12], we further extend the MLP-3D networks to a two-path networks by adding an additional path. To maximize the complementarity between paths, we erase the background of the input frames in the additional path by subtracting the averaged frame over time.

Table 2. Performance contribution of each design in MLP-3D networks. Top-1 accuracies are reported on SS-V2 and Kinetics-400 validation set, respectively.

Network	Input size		Two-path	SS-V2	K-400
	128^2	224^2			
MLP-3D-XS	✓			64.6	75.0
	✓		✓	65.6	76.2
		✓		65.4	76.5
		✓	✓	66.0	77.2
MLP-3D-S	✓			65.5	77.2
	✓		✓	66.7	78.0
		✓		66.7	79.2
		✓	✓	67.2	80.0
MLP-3D-M	✓			65.7	78.0
	✓		✓	66.9	78.8
		✓		67.2	79.9
		✓	✓	68.0	81.0
MLP-3D-L	✓			66.0	78.3
	✓		✓	66.7	78.9
		✓		67.6	80.4
		✓	✓	68.5	81.3

Table 2 details the accuracy improvements on SS-V2 and Kinetics-400 datasets by different designs in MLP-3D networks. When exploiting MLP-3D-XS as the backbone, the larger input size (224^2) successfully boosts up the top-1 accuracy from 64.6% to 65.4% on SS-V2 and from 75.0% to 76.5% on Kinetics-400. This demonstrates the effectiveness of training on larger resolution for video recognition. The extension to two-path networks which explores the complementarity across paths further leads to the performance improvement of 0.6% and 0.7% on SS-V2 and Kinetics-400, respectively. Moreover, across different MLP-3D networks, the deeper networks exhibit significantly better performance than the shallower ones. Specifically, the overall performance is improved from 66.0% to 68.5% on SS-V2 and from 77.2% to 81.3% on Kinetics-400 by replacing MLP-3D-XS with MLP-3D-L. The results verify that a deeper network has the larger learning capacity.

4.4. Comparisons with State-of-the-Art Models

We compare with several state-of-the-art techniques on SS-V2 dataset. Table 3 summarizes the performance comparisons. The baselines are pre-trained on ImageNet-1K (IN-1K), ImageNet-21K (IN-21K), Kinetics-400 (K-400) or Kinetics-600 (K-600) datasets. The "views" represents the number of clips sampled from the full video during inference. Overall, the small-sized MLP-3D-S network achieves the highest top-1 accuracy of 67.2% among the methods pre-trained on ImageNet-1K. More importantly, MLP-3D-S only spends 108G FLOPs, which is 18% less

Table 3. Comparisons with the state-of-the-art methods on SS-V2.

Method	Pre-train	GFLOPs	Views	Params	Top-1	Top-5
TSM-RGB [34]	IN-1K	62	2 × 3	42.9	63.4	88.5
ACTION-Net [65]		69	1 × 1	28.0	64.0	89.3
STM [24]		66	10×3	24.0	64.2	89.8
SmallBig [31]		157	2 × 3	–	64.5	89.1
MSNet [26]		67	1 × 1	24.6	64.7	89.4
TEA [32]		70	10×3	–	65.1	89.9
DG-P3D [47]		123	10×3	–	65.5	90.3
TDN [62]		132	1 × 1	–	66.9	90.9
TimeSformer-HR [3]	IN-21K	1703	1 × 3	121.4	62.5	–
ViViT-L/16 × 2 [1]		903	–	352.1	65.4	89.8
SlowFast R101 [12]	K-400	106	1 × 3	53.3	63.1	87.6
MViT-B, 64 × 3 [10]		455	1 × 3	36.6	67.7	90.9
Video Swin-B [39]		321	1 × 3	88.8	69.6	92.7
MViT-B-24, 32 × 3 [10]	K-600	236	1 × 3	53.2	68.7	91.5
MLP-3D-XS	IN-1K	60	1 × 3	55.1	66.0	90.4
MLP-3D-S		108	1 × 3	74.1	67.2	91.3
MLP-3D-M		183	1 × 3	88.3	68.0	91.7
MLP-3D-L		336	1 × 3	149.4	68.5	92.0

Table 4. Comparisons with the state-of-the-art methods on SS-V1.

Method	Pre-train	GFLOPs	Views	Params	Top-1	Top-5
TSM-RGB [34]	IN-1K	62	2 × 3	42.9	47.2	77.1
STM [24]		66	10 × 3	24.0	50.7	80.4
SmallBig [31]		157	2 × 3	–	51.4	80.7
MSNet [26]		67	1 × 1	24.6	52.1	82.3
TEA [32]		70	10 × 3	–	52.3	81.9
DG-P3D [47]		123	10 × 3	–	52.8	81.8
TDN [62]		132	1 × 1	–	55.3	83.3
MLP-3D-XS	IN-1K	60	1 × 3	55.1	54.4	82.5
MLP-3D-S		108	1 × 3	74.1	55.2	83.2
MLP-3D-M		183	1 × 3	88.3	56.2	83.5
MLP-3D-L		336	1 × 3	149.4	56.5	83.5

than that of TDN. The top-1 accuracy is further improved to 68.5% by the deeper MLP-3D-L network, which leads to the performance boost of 1.6% over the best competitor TDN. Please note that most transformer-based models employ the pre-training on larger datasets. Nevertheless, MLP-3D-L network still outperforms TimeSformer-HR and ViViT-L by 6.0% and 3.1%, respectively. MViT-B and Video Swin-B, which are pre-trained with more video data, expectably obtain higher accuracy. Table 4 shows the comparisons on SS-V1 dataset. In this comparison, we merely transfer the architectures searched on SS-V2 but train the weights on SS-V1 to validate the transferability. The similar performance trends are observed on SS-V1. Specifically, MLP-3D-L attains 56.5% top-1 accuracy, which leads the performance by 1.2% against TDN method. The results validate the use of the searched MLP-3D networks to the dataset with similar target categories.

Then, we turn to evaluate MLP-3D networks on the large-scale Kinetics-400 dataset. The performance comparisons are reported in Table 5. Specifically, with ImageNet-1K pre-training, MLP-3D-L network achieves 81.4% top-1 accuracy, making the improvements over the recent approaches Video Swin-B, DG-P3D, TDN and LGD-3D by 0.8%, 0.9%, 2.0% and 2.0%, respectively. Furthermore, MLP-3D-L with less FLOPs is impressively superior to several video transformers pre-trained on ImageNet-21K, e.g., ViT-VTN, TokShift, TimeSformer-L and ViViT-L, which spend about ten times more FLOPs.

Table 5. Comparisons with the state-of-the-art methods on K-400.

Method	Pre-train	GFLOPs	Views	Params	Top-1	Top-5
R(2+1)D [59]	None	75	10×1	61.8	72.0	90.0
ip-CSN-152 [58]		109	10×3	32.8	77.8	92.8
CorrNet-101 [61]		224	10×3	–	79.8	93.9
SlowFast R101+NL [12]		234	10×3	59.9	79.8	93.9
X3D-XXL [11]		144	10×3	20.3	80.4	94.6
MViT-B, 32 × 3 [10]		170	1 × 5	36.6	80.2	94.4
MViT-B, 64 × 3 [10]		455	3 × 3	36.6	81.2	95.1
I3D [5]	IN-1K	108	–	25.0	72.1	90.3
NL I3D-101 [64]		359	10×3	61.8	77.7	93.3
SmallBig [31]		475	4 × 3	–	78.7	93.7
LGD-3D [48]		195	–	–	79.4	94.4
TDN [62]		198	10×3	–	79.4	94.4
DG-P3D [47]		218	10×3	–	80.5	94.6
Video Swin-T [39]		88	4 × 3	28.2	78.8	93.6
Video Swin-S [39]		166	4 × 3	49.8	80.6	94.5
Video Swin-B [39]		282	4 × 3	88.1	80.6	94.6
ViT-VTN [41]	IN-21K	4218	1 × 1	11.0	78.6	93.7
TokShift [71]		2096	10×3	303.4	80.4	94.4
TimeSformer-L [3]		2380	1 × 3	121.4	80.7	94.7
ViViT-L/16 × 2 [1]		1446	4 × 3	310.8	80.6	94.7
ViViT-L/16 × 2 320 [1]		3992	4 × 3	310.8	81.3	94.7
Video Swin-B [39]		282	4 × 3	88.1	82.7	95.5
Video Swin-L(384↑) [39]		2107	10×5	200.0	84.9	96.7
MLP-3D-XS	IN-1K	57	4 × 3	50.1	77.2	93.1
MLP-3D-S		102	4 × 3	68.5	80.2	93.8
MLP-3D-M		170	4 × 3	80.5	81.0	94.9
MLP-3D-L		308	4 × 3	135.6	81.4	95.2

5. Conclusion and Discussion

We have proposed a new family of MLP-like 3D architectures named MLP-3D networks for video recognition. Particularly, we investigate the token interaction across time in MLP-like architecture, by designing MLP-3D blocks with token-mixing MLP decomposed by height, width and time dimensions. For time dimension, we have devised variants of novel grouped time mixing (GTM) operations for group-based interaction between tokens. The type of GTM and its group size of each block are determined by an efficient greedy architecture search. Experiments conducted on three datasets, i.e., Something-Something V1 & V2, and Kinetics-400, validate that MLP-3D networks achieve superior performances than other video recognition techniques under the same pre-training scheme. The competitive performances also show the high potential of MLP-like architectures for video analysis. More remarkably, the network is easier to train and consumes less FLOPs.

Broader Impact. Our MLP-3D shows a great potential of MLP-like architectures for video analysis, which are easily developed with less computations. This could increase the risk of video understanding model or its outputs being used incorrectly, such as for unauthorized surveillance.

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

References

[1] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. *arXiv preprint arXiv:2103.15691*, 2021. 2, 3, 6, 8

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 3

[3] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *ICML*, 2021. 2, 3, 8

[4] Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. Efficient architecture search by network transformation. In *AAAI*, 2018. 5

[5] João Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017. 1, 2, 6, 8

[6] Shoufa Chen, Enze Xie, Chongjian Ge, Ding Liang, and Ping Luo. Cyclemlp: A mlp-like architecture for dense prediction. *arXiv preprint arXiv:2107.10224*, 2021. 1, 2, 3, 5, 6

[7] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *ICCV*, 2019. 2

[8] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *NeurIPS*, 2020. 5

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 1, 2

[10] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. *arXiv preprint arXiv:2104.11227*, 2021. 2, 5, 7, 8

[11] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *CVPR*, 2020. 1, 2, 8

[12] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *ICCV*, 2019. 1, 2, 6, 7, 8

[13] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fründ, and *et al*. The "something something" video database for learning and evaluating visual common sense. In *ICCV*, 2017. 5, 6

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 2

[15] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 3

[16] Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. Augment your batch: Improving generalization through instance repetition. In *CVPR*, 2020. 6

[17] Qibin Hou, Zihang Jiang, Li Yuan, Ming-Ming Cheng, Shuicheng Yan, and Jiashi Feng. Vision permutator: A permutable mlp-like architecture for visual recognition. *arXiv preprint arXiv:2106.12368*, 2021. 1, 2, 3

[18] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 2

[19] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018. 1, 2

[20] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *ECCV*, 2016. 5

[21] Zilong Huang, Youcheng Ben, Guozhong Luo, Pei Cheng, Gang Yu, and Bin Fu. Shuffle transformer: Rethinking spatial shuffle for vision transformer. *arXiv preprint arXiv:2106.03650*, 2021. 1, 2, 4

[22] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 2

[23] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE Trans. on PAMI*, 35(1):221–231, 2013. 2

[24] Boyuan Jiang, MengMeng Wang, Weihao Gan, Wei Wu, and Junjie Yan. Stm: Spatiotemporal and motion encoding for action recognition. In *ICCV*, 2019. 8

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 2

[26] Heeseung Kwon, Manjin Kim, Suha Kwak, and Minsu Cho. Motionsqueeze: Neural motion feature learning for video understanding. In *ECCV*, 2020. 8

[27] Changlin Li, Tao Tang, Guangrun Wang, Jiefeng Peng, Bing Wang, Xiaodan Liang, and Xiaojun Chang. Bossnas: Exploring hybrid cnn-transformers with block-wisely self-supervised neural architecture search. *arXiv preprint arXiv:2103.12424*, 2021. 2

[28] Dong Li, Zhaofan Qiu, Yingwei Pan, Ting Yao, Houqiang Li, and Tao Mei. Representing videos as discriminative subgraphs for action recognition. In *CVPR*, 2021. 1

[29] Guohao Li, Guocheng Qian, Itzel C Delgadillo, Matthias Muller, Ali Thabet, and Bernard Ghanem. Sgas: Sequential greedy architecture search. In *CVPR*, 2020. 2

[30] Rui Li, Yiheng Zhang, Zhaofan Qiu, Ting Yao, Dong Liu, and Tao Mei. Motion-focused contrastive learning of video representations. In *ICCV*, 2021. 1

[31] Xianhang Li, Yali Wang, Zhipeng Zhou, and Yu Qiao. Smallbignet: Integrating core and contextual views for video classification. In *CVPR*, 2020. 8

[32] Yan Li, Bin Ji, Xintian Shi, Jianguo Zhang, Bin Kang, and Limin Wang. Tea: Temporal excitation and aggregation for action recognition. In *CVPR*, 2020. 8

[33] Yehao Li, Ting Yao, Yingwei Pan, and Tao Mei. Contextual transformer networks for visual recognition. *IEEE Trans. on PAMI*, 2022. 1, 2

[34] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *ICCV*, 2019. 8

[35] Chenxi Liu, Barret Zoph, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *ECCV*, 2018. 2

[36] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. In *ICLR*, 2018. 2

[37] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *ICLR*, 2019. 2

[38] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 1, 2, 4

[39] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. *arXiv preprint arXiv:2106.13230*, 2021. 2, 5, 8

[40] Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. In *NIPS*, 2018. 2, 5

[41] Daniel Neimark, Omri Bar, Maya Zohar, and Dotan Asselmann. Video transformer network. *arXiv preprint arXiv:2102.00719*, 2021. 2, 8

[42] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 4

[43] Mandela Patrick, Dylan Campbell, Yuki M Asano, Ishan Misra Florian Metze, Christoph Feichtenhofer, Andrea Vedaldi, Jo Henriques, et al. Keeping your eye on the ball: Trajectory attention in video transformers. In *NeurIPS*, 2021. 2

[44] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *ICML*, 2018. 2, 5

[45] Zhaofan Qiu, Ting Yao, and Tao Mei. Deep quantization: Encoding convolutional activations with deep generative model. In *CVPR*, 2017. 1

[46] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *ICCV*, 2017. 1, 2, 3

[47] Zhaofan Qiu, Ting Yao, Chong-Wah Ngo, and Tao Mei. Optimization planning for 3d convnets. In *ICML*, 2021. 8

[48] Zhaofan Qiu, Ting Yao, Chong-Wah Ngo, Xinmei Tian, and Tao Mei. Learning spatio-temporal representation with local and global diffusion. In *CVPR*, 2019. 1, 8

[49] Zhaofan Qiu, Ting Yao, Chong-Wah Ngo, Xiao-Ping Zhang, Dong Wu, and Tao Mei. Boosting video representation learning with multi-faceted integration. In *CVPR*, 2021. 1

[50] Zhaofan Qiu, Ting Yao, Yan Shu, Chong-Wah Ngo, and Tao Mei. Condensing a sequence to one informative frame for video recognition. In *ICCV*, 2021. 1

[51] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1, 2

[52] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. In *CVPR*, 2021. 2

[53] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 2

[54] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *arXiv preprint arXiv:2105.01601*, 2021. 1, 2, 3

[55] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. 1

[56] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. *arXiv preprint arXiv:2103.17239*, 2021. 2

[57] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015. 1, 2

[58] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *ICCV*, 2019. 2, 8

[59] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, 2018. 2, 3, 8

[60] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017. 1, 2, 3

[61] Heng Wang, Du Tran, Lorenzo Torresani, and Matt Feiszli. Video modeling with correlation networks. In *CVPR*, 2020. 8

[62] Limin Wang, Zhan Tong, Bin Ji, and Gangshan Wu. Tdn: Temporal difference networks for efficient action recognition. In *CVPR*, 2021. 8

[63] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *arXiv preprint arXiv:2102.12122*, 2021. 1, 2

[64] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018. 1, 8

[65] Zhengwei Wang, Qi She, and Aljosa Smolic. Action-net: Multipath excitation for action recognition. In *CVPR*, 2021. 8

[66] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. *arXiv preprint arXiv:2103.15808*, 2021. 2

[67] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 2

[68] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning:

Speed-accuracy trade-offs in video classification. In *ECCV*, 2018. 2, 3

[69] Ting Yao, Yiheng Zhang, Zhaofan Qiu, Yingwei Pan, and Tao Mei. Seco: Exploring sequence supervision for unsupervised representation learning. In *AAAI*, 2021. 1

[70] Tan Yu, Xu Li, Yunfeng Cai, Mingming Sun, and Ping Li. S^2-mlpv2: Improved spatial-shift mlp architecture for vision. *arXiv preprint arXiv:2108.01072*, 2021. 1, 2, 3

[71] Hao Zhang, Yanbin Hao, and Chong-Wah Ngo. Token shift transformer for video classification. In *ACM MM*, 2021. 2, 8

[72] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018. 2

[73] Yanyi Zhang, Xinyu Li, Chunhui Liu, Bing Shuai, Yi Zhu, Biagio Brattoli, Hao Chen, Ivan Marsic, and Joseph Tighe. Vidtr: Video transformer without convolutions. In *ICCV*, 2021. 2

[74] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *AAAI*, 2020. 5

[75] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *ICML*, 2017. 2