Dynamic Equilibrium Module for Action Recognition

QILI ZENG, M. OZAN TEZCAN, (Member, IEEE), AND JANUSZ KONRAD, (Fellow, IEEE)
Boston University, Boston, MA 02215

Corresponding author: Qili Zeng (e-mail: qzeng@bu.edu).

ABSTRACT
Temporal variations, such as sudden motion, acceleration and occlusions, occur frequently in real-world videos and force video-modeling networks to account for them. However, often they are not beneficial for recognizing actions at coarse granularity and thus may impede spatio-temporal learning. Prior solutions to this problem usually introduce multiple network branches to process input frames at different sampling rates or employ special components to explore inter-frame relations, which are computationally expensive. In this paper we propose a simple and flexible Dynamic Equilibrium Module (DEM) for video modeling through adaptive Eulerian motion manipulation. The proposed module can be directly inserted into 3D and (2+1)D backbone networks to effectively reduce the impact of temporal variations on video modeling and learn spatio-temporal representations with higher robustness. We demonstrate performance gains due to the use of DEM in R3D and R(2+1)D models on Kinetics-400, UCF-101, and HMDB-51 datasets.

INDEX TERMS
Action Recognition, Video Analysis, Deep Learning

I. INTRODUCTION

ACTION recognition is a fundamental task in video understanding and its key component is learning motion information from video. Motion in real-world videos can dramatically change in just a few frames and, therefore, is difficult to predict. Figure 1 provides an example of a tennis-game video. A player, most of the time, moves around the court with a relatively constant speed, but could suddenly accelerate in order to reach a far-out ball. Various forms of acceleration, deceleration, occlusions, etc., can be thought of as temporal variations. These variations lead to temporal discontinuities in visual perception. Since the human visual system is endowed with bottom-up and top-down attention [1], [2] and with prior knowledge of object structure, visual coherence with respect to a moving object can be maintained over time, despite these temporal variations. While the human visual system learns to account for these variations, neural networks trained from scratch struggle with spatio-temporal learning because of temporal variations.

At a low level, inter-frame temporal variations (e.g., high acceleration) lead to large pixel displacements between adjacent frames. It has been observed in early CNN-based optical flow estimators [3], [4], that a large pixel displacement between adjacent frames cannot be accurately handled by local convolutional operators without special structures [5], [6].

FIGURE 1. Example of inter-frame motion variations in video. An ideal scenario for spatio-temporal learning would contain actions evolving smoothly and coherently (with temporal variations loosely depicted as the thin green-dashed curve). However, a real-world tennis-game video involves highly-variable player movements (loosely depicted as the bold green curve). In this example, four types of player movement are shown as time-lapse frames. The two bottom frames depict two usual movements in tennis: a baseline forehand return (left) and an after-return walk to the baseline center (right). The two top frames show two extreme movements with high player acceleration: a jumping backhand return (left) and a sudden sprint to reach a far-out ball (right). These extreme accelerations are difficult to learn and often lead to noisy intermediate spatio-temporal features that are not helpful for recognizing the action at a coarse level (i.e., tennis game).
This, however, can also be a problem for action recognition models with mainstream backbone networks [7]–[9] that simply consist of stacked local convolutions. When recognizing actions from several frames (a video segment), the network must include different groups of filters across layers in order to adapt to time-varying motion. At a higher level, inter-segment temporal variations (e.g., temporary occlusion) add extra difficulties to higher layers when fusing intermediate representations along the temporal axis, which affects the quality of clip-level video features.

Although temporal variations may contribute to recognizing certain motion details, they lead to heavy but unnecessary cost for a network that is trying to predict a coarse action label rather than a fine detail in an action. For general action recognition, detailed motion information (e.g., that a tennis player suddenly jumps, as shown in Figure 1) should not influence the final classification of a video (e.g., a tennis game). Therefore, if some temporal variations in a video could be suppressed or, in other words, if the dynamics in a video could be equilibrated, then learning from spatio-temporal data could be significantly simplified.

To this end, we introduce the Dynamic Equilibrium Module (DEM), an insertable, flexible module that works with both 3D convolutions [10] and (2+1)D convolutions [9], [11]. DEM generates Eulerian motion compensation based on temporal variations detected in input video frames or in intermediate feature maps, and passes this information back to the backbone network for motion manipulation. DEM modifies feature maps before they are convolved with subsequent temporal or spatio-temporal filters in order to reduce the impact of temporal variations on video recognition.

The contributions of this paper can be summarized as follows. 1) We propose a quantitative description of temporal variations based on Eulerian interpretation of motion. 2) We propose a simple and novel Dynamic Equilibrium Module for video modeling and demonstrate its effectiveness in handling temporal variations via experiments on mainstream datasets. 3) We provide a detailed analysis of the influence of temporal variations on action recognition through extensive ablation study.

II. RELATED WORK

Action Recognition. After the initial success on still-image recognition [12], [13], many deep networks have been proposed for video recognition. Most of the mainstream networks can be classified into either two-stream networks [14]–[17] or 3D networks [7], [8], [10], [18], [19].

Although two-stream models, by introducing an independent network branch to learn from optical flow, provide good performance, computing optical flow is expensive and, more importantly, could be considered as feature engineering in the context of deep networks (optical flow is pre-computed from video frames). Recent work made progress in constructing learnable motion features [20]–[23] and exploring alternative structures for spatio-temporal learning with better performance or higher efficiency [9], [11], [24]–[26].

Some solutions investigated possible approaches to learn from dynamics without explicitly using or estimating motion features. Non-local Networks [27] established pixel-to-pixel relations across all feature maps, implicitly learning motion through generalized self-attention. TSM [26] performs efficient temporal modeling by moving the feature map along the temporal dimension. Correlation Networks [28] established frame-to-frame matches over convolutional feature maps through learnable correlation operators.

Motion Representation. Motion in a video sequence implies a relationship between video frames and reflects important properties of moving objects.

The Lagrangian perspective on motion considers it as the movement of particles in a medium. The most successful hand-crafted Lagrangian approaches are dense optical flow [29] and improved dense trajectories [30]. Since accurate optical flow computation using variational approaches requires hundreds of iterations [31], CNNs were explored for optical flow estimation as well [3]–[6], [32].

The Eulerian perspective on motion, on the other hand, considers motion as a variation of pixel values at fixed positions over time. Previous explorations have successfully employed Eulerian motion in video motion magnification [33]–[35] and video frame interpolation [36]. In the area of action recognition, RGB differences, the simplest Eulerian motion representation, have been explored in TSN [37]. Similar to Wadhwa et al. [34], phase-based motion, where movement’s state is represented by the phase of pixels in complex domain, was also applied to action recognition [38]. TDN [39] extracts learning-based Eulerian motion as an independent stream for video classification.

Temporal Modeling in Video. Recently, learning from temporal information has been extensively researched [40]–[43]. The method proposed in this paper is in line with video modeling using multivariate or temporal multi-scale sampling, which emphasizes learning representations for actions occurring at various speeds. mGRU [44] followed the idea of Clockwork RNN [45] and encoded video frames with different intervals. Random Temporal Skipping [46] attempted to cover all motion speed variations by randomizing the sampling rate during training in an exhaustive manner. Similarly, DTPN [47] sampled frames at different frame rates to construct a natural pyramidal representation for arbitrary-length input videos. SlowFast [48] included two network streams for both high and low frame-rate inputs, modeling motion at fine and coarse temporal resolutions separately. TPN [49] aggregated the information of temporal variations at multiple feature levels in the backbone network and fused them to make the final prediction. TEA [50] calculated the first-order difference of temporally-adjacent features to discover and enhance motion-sensitive channels. The proposed method differs from the above solutions in the way it handles temporal variations. While most of prior research focuses on fusing or leveraging temporally multi-scale information in order to obtain robust video features, our method attempts to directly suppress its influence on spatio-temporal learning in
the backbone network through second-order Eulerian motion manipulation.

III. DYNAMIC EQUILIBRIUM MODULE

As discussed in the previous sections, large temporal variations in videos may reduce the performance of action recognition networks. In this section, we formulate DEM aiming to reduce the impact of temporal variations on the network’s final prediction.

A. EULERIAN DESCRIPTION OF TEMPORAL VARIATIONS

Temporal variations occurring in a video sequence capture the change in dynamics of objects. We first define general Eulerian motion representation in the context of neural networks.

A Eulerian motion description typically involves computing the difference of certain properties of an image sequence either in space-time or in spatio-temporal frequency domain. For instance, using temporal convolution $TConv$ with filters of size $1 \times 1 \times 1$ (temporal $\times$ horizontal $\times$ vertical dimensions), the dynamics present in a video sequence could be described in the most general form as follows:

$$\Delta I_n = TConv(I_{n-1}, I_{n-2}, \ldots, I_{n-t})$$

(1)

where $I_n$ denotes a video frame at time $n$. As modern convolutional neural networks learn representation in a hierarchical manner, we assume such operation is not only applicable to the input frames (low-level motion), but also to intermediate feature maps (high-level motion). In practice, motion description is usually inferred from a pair of input frames, in which case $T$ in equation (1) would equal 2.

By observing at least 3 adjacent video frames, humans can easily determine whether a particular frame contains large-amplitude motion, occlusions, acceleration, deceleration, etc. One feasible quantitative description of such variations can be obtained by analyzing either three consecutive video frames (in the input layer) or three consecutive spatio-temporal representations (in subsequent layers), denoted $x_{n-1}, x_n, x_{n+1}$, as follows:

$$D_n = g(f(x_{n-1}, x_n), f(x_n, x_{n+1})) - f(x_{n-1}, x_{n+1})$$

(2)

where $f$ and $g$ refer to $TConv$ operations with different filters. The role of $D_n$ can be explained as follows. In case of an action that evolves uniformly in time (for example, linear, constant-velocity movement such as a cyclist coasting on a flat road), motion description based on the observation of $(x_{n-1}, x_{n+1})$ should be numerically close to the composition of motion descriptions based on observations of $(x_{n-1}, x_n)$ and $(x_n, x_{n+1})$ and, consequently, $D_n$ should be close to 0. If the magnitude of $D_n$ is large, then either $x_n$ or $(x_{n-1}, x_{n+1})$ likely disobeys action uniformity in time (e.g., the cyclist makes a sudden turn). See Figure 2 for detailed illustration on a simple example. While a large magnitude of $D_n$ can be useful in recognizing a particular detail in an action (e.g., cyclist’s turn), it is not beneficial for determining a coarse-grained action (i.e., cycling in this example). In order to learn the fine details of an action, a more complex network (more parameters) or extra supervision would be needed. However, since our interest is in action recognition with coarse granularity, the goal is to “discover” such fine details and help the backbone network compensate for them.
**B. MODULE ARCHITECTURE**

An observation of time-varying appearance by human visual system leads to motion perception. What is presented in consecutive video frames determines how motion is interpreted, e.g., by speeding up a video of "touching", people may understand it as "hitting". Motion interpretation in a neural network works similarly — pattern matching between frames could fail if excessive temporal variability is present. Following the idea of motion magnification [51], we believe that spatio-temporal representation learning can be influenced by adaptive manipulation of the appearance in an image sequence.

The Dynamic Equilibrium Module (DEM) is proposed to facilitate spatio-temporal learning via motion manipulation. DEM implements equation (2) to estimate temporal variations around a certain frame or a spatial feature map. Function $f$ in DEM is realized by using dilated convolution, where computing $f(x_n, x_{n+1})$ is implemented with a normal $TConv$ and computing $f(x_{n-1}, x_{n+1})$ is implemented by the same $TConv$ with the dilation rate of 2. Function $g$ is implemented by another $TConv$ operator with dilation rate of 1. The representation of temporal variations needs then to be fused with the original spatial feature map to generate a compensation signal. We considered several possible fusion schemes to generate this signal and selected pixel-wise multiplication due to its higher efficiency and lower memory usage. The signal is then passed back to the backbone layer for motion manipulation, implemented simply by pixel addition. By doing so, DEM attempts to compensate those locations in the feature sequences where temporal variations are detected, which further alleviates their influence. Figure 3(a) shows a diagram of DEM.

DEM is able to work with both 3D and (2+1)D convolutions. In standard 3D convolutional layers, that are used for video modeling, learning spatial and temporal information is intermixed and performed simultaneously, which means the temporal learning can only be influenced externally, as illustrated in Figure 3(b). In this case, motion manipulation would actually take effect in the next convolutional layer. On the contrary, a DEM can be inserted into a (2+1)D convolution and influences the temporal learning internally due to the fact that (2+1)D convolution executes spatial and temporal learning in two consecutive stages. Figure 3(c) illustrates the interaction between a (2+1)D layer and a DEM.

**IV. EXPERIMENTS**

**A. BACKBONE NETWORK**

DEM is designed for spatio-temporal convolutional networks, which typically consist of 3D and (2+1)D filters. Although many such networks have been proposed [9], [11], [18], [24], [52], we select residual networks built upon a homogeneous architecture of 3D filters (R3D [18]) and (2+1)D filters (R(2+1)D [9]) as the backbone for more clear analysis of the impact of DEM on spatio-temporal learning.

Except for experiments in Section V, we insert DEM into all layers of R3D/R(2+1)D. Since training very deep networks is computationally expensive and time-consuming, without loss of generality, we implement all the experiments with 18-layer R3D/R(2+1)D. We believe the presented results also provide insights and predictions for DEM’s impact on deeper networks or other applicable models.

**B. PARAMETER ADJUSTMENT**

The direct insertion of DEM into a convolutional layer leads to an increased number of model parameters. For a fair comparison, in the experiments with 3D convolution we divide the number of filters in all convolutional layers in the R3D+DEM model by 1.25. We found empirically that such a strategy leads to nearly the same number of parameters in R3D and R3D+DEM. The architecture of an R3D-18 network is described in Table 1.

In experiments with (2+1)D convolution, similarly to Tran et al. [9], we adjusted the number of midplane channels, i.e., the number of spatial filters, in all the (2+1)D convolutional layers with a DEM to ensure that the total number of parameters in the network is equivalent to that of R3D networks. More specifically, the number of parameters in one 3D convolutional layer can be calculated as $t \times d^2 \times N_{in} \times N_{out}$, where $t$ refers to the temporal length, $d$ is the spatial width and height, and $N_{in}, N_{out}$ are the numbers of channels in the input and output tensors, respectively. In 3D convolution, the typical value of $t$ and $d$ is 3. A (2+1)D convolutional layer would then have $d^2 \times N_{in} \times N_{mid} + t \times N_{mid} \times N_{out}$ parameters, where $N_{mid}$ denotes the number of midplane channels. After DEM insertion, the number of parameters in a (2+1)D + DEM unit would be $d^2 \times N_{in} \times N_{mid} + t \times N_{mid} \times N_{out} + (4 + d^2) \times N_{mid} \times N_{mid}$. We solve for $N_{mid}$ in order to have the same number of network parameters in R3D, R(2+1)D and R(2+1)D+DEM.

**C. DATASETS**

Kinetics-400 [55] is a large-scale human-centric video dataset collected from Youtube, containing 400 human action classes, 246,534 training videos and 20,000 validation videos. Due to data deletion, our version of Kinetics-400 consists of 234,584 videos in the training set and 19,760 videos in the validation set (≈5% data lost).

UCF-101 [56] is another trimmed video dataset for human action recognition, consisting of 13,320 videos with 101

| layer | output size | R3D-18 |
|-------|-------------|--------|
| conv1 | $L \times 56 \times 56$ | $3 \times 7^2 \times 64$, stride $1 \times 2^2$ |
| conv2_x | $L \times 56 \times 56$ | $3 \times 3^2 \times 64, 3 \times 3^2 \times 64 \times 2$ |
| conv3_x | $L \times 28 \times 28$ | $3 \times 3^2 \times 128, 3 \times 3^2 \times 128 \times 2$ |
| conv4_x | $L \times 14 \times 14$ | $3 \times 3^2 \times 256, 3 \times 3^2 \times 256 \times 2$ |
| conv5_x | $L \times 7 \times 7$ | $3 \times 3^2 \times 512, 3 \times 3^2 \times 512 \times 2$ |
|       | $1 \times 1 \times 1$ | global average pooling and fc |
TABLE 2. Performance comparison of DEM with state-of-the-art methods on Kinetics-400 validation set. Performance is measured by Top-1 and Top-5 video-based accuracy (%). Avg. reports the arithmetic mean of Top-1 and Top-5 accuracy.

| Model          | Backbone          | Pre-train | $T \times \tau$ | Top-1 | Top-5 | Avg. | GFLOPs x views | #Param |
|----------------|-------------------|-----------|-----------------|-------|-------|------|----------------|--------|
| R3D [9]        | ResNet-18         | None      | $16 \times 1$   | 64.2  | -     | 3.1  | $10 \times 33.4$ | 33.4M  |
| R(2+1)D [9]    |                   |           |                 | 68.0  | -     | 83.4 | 33.4M          |        |
| ARTNet-18 [53] | ImageNet          |           |                 | 69.2  | 88.3  | 78.8 | $23.7 \times 250$ | 35.2M  |
| TrajectoryNet [54] |                 |           |                 | -     | 77.8  |      | N/A            | N/A    |
| R3D+DEM        | ResNet-18         | None      | $16 \times 1$   | 62.3  | 83.3  | 72.8 | $10 \times 33.4$ | 33.4M  |
| R(2+1)D+DEM    |                   |           |                 | 63.8  | 84.4  | 74.1 | 33.1M          |        |
| R3D (Ours)     | ResNet-18         | None      | $16 \times 1$   | 64.4  | 85.1  | 74.8 | $10 \times 33.4$ | 33.4M  |
| R(2+1)D (Ours) | ARTNet-18         |           |                 | 67.3  | 86.9  | 77.1 | 33.3M          |        |

TABLE 3. Impact of subsampling rate ($\tau$) on video-based accuracy (%) of R3D/R(2+1)D+DEM on Kinetics-400.

| $T \times \tau$ | R3D+DEM | R(2+1)D+DEM |
|----------------|---------|-------------|
| $16 \times 1$  | 63.8    | 84.4        |
| $16 \times 2$  | 64.2    | 85.0        |
| $16 \times 4$  | 64.3    | 85.3        |

Analyzed classes.

HMDB-51 [57] comprises 6,766 videos collected from real movies and YouTube, and annotated into 51 classes.

D. TRAINING AND EVALUATION

We mainly follow the training configurations provided in [9] and only use RGB frames as input. For Kinetics, we extract 5 clips per video in each epoch (temporal jittering) and resize the extracted frames to $128 \times 171$ multiplied by a random factor in $[1, 1.25]$ range (spatial jittering). The sampled clips are then horizontally flipped with a random factor of 0.5 and randomly cropped to $112 \times 112$. The number of frames per clip, denoted as $T$, is set to 16.

The experiments described below also involve using temporally subsampled frames with a subsampling rate $\tau$ selected from $\{1, 2, 4, 8\}$. Therefore, the effective time span of an action captured in one clip covers the temporal range $[k, \ldots, k + T \cdot \tau]$ in the original video. Since we use 8 GPUs in our experiments, we adopt gradient accumulation to simulate the batch size used in the original recipe [9] and use a smaller momentum of 0.05 in all the batch normalization (BN) layers. The first 10 epochs are used for warm-up, followed by normal training stage with SGD optimizer and the initial learning rate of $0.01$. We divide the learning rate by 10 at epoch 20, 26, and 30. A typical training cycle consists of 35 epochs. This training configuration is adjusted for other datasets according to the volume of each dataset. Models trained on Kinetics are trained from scratch, while experiments on UCF-101 and HMDB-51 are implemented with pre-trained models on Kinetics.

Unless specified otherwise, we report video-based performance as top-1/top-5 accuracy, calculated by averaging the predictions of uniformly sampled clips. For Kinetics, we follow the common practice to extract 10 clips per video, while for the other datasets we use 2 clips per video. Clip-based accuracy, reported in some experiments, is calculated by evaluating the accuracy based on clip-level predictions.

E. EXPERIMENTAL RESULTS

Results on Kinetics-400. In Table 2, we compare the performance of R3D and R(2+1)D with DEM and those without DEM and other state-of-the-art models on Kinetics-400. Note that most high-performance models in the literature employ...
FIGURE 5. Histogram of top-1 accuracy difference (%) due to the use of DEM in 3D backbone across Kinetics-400 action classes. The difference is calculated as the per-class accuracy of a model with DEM minus that of the same model without DEM. All 400 classes are included in this evaluation.

FIGURE 6. Top-1 accuracy difference (%) for individual Kinetics-400 action classes. The difference is calculated as the per-class accuracy of a model with DEM minus that of the same model without DEM. Only the top and bottom 10 classes are shown.

much deeper backbones [26], [48], [49] and thereby it is unfair to compare our models to them. Table 2 only includes models whose backbone network is the same as ours. The first four rows show results of the state-of-the-art algorithms as reported in their respective papers. Since the number of training and validation videos of Kinetics-400 decreased by 5% after the publication of these papers, it would not be fair to compare our work against these results. Therefore, we trained R3D and R(2+1)D in our environment using the most recent version of Kinetics-400 and following the training details provided in [9]. In Table 2, they are referred to as “R3D (Ours)” and “R(2+1)D (Ours)”. We adopt these reproduced versions as the baseline for comparison. Clearly, both R3D+DEM and R(2+1)D+DEM outperform their corresponding baseline models that use roughly the same number of parameters. R(2+1)D+DEM also achieves close performance to ARTNet-18 and TrajectoryNet although ARTNet-18 has more parameters and TrajectoryNet is pretrained on ImageNet. Furthermore, they both sample many more clips during evaluation and thus are more computationally expensive than our model.

Table 3 further shows that, as the subsampling rate $\tau$ becomes larger, the prediction accuracy of R3D+DEM and R(2+1)D+DEM increases. Since the number of sampled
frames is always 16, this performance gain is realized without extra computational cost and demonstrates the contribution of DEM to learning from sparsely-sampled frames that encompass a larger temporal receptive field of the video and higher temporal variations. Figure 4 illustrates the impact of $\tau$ on the accuracy in more detail based on a R(2+1)D+DEM model. It can be observed that video-based accuracy of R(2+1)D+DEM saturates at $\tau = 4$ and decreases at $\tau = 8$. We believe this is due to the limited length of videos (~300 frames) in Kinetics-400. When uniformly sampling 16 frames with $\tau = 8$, the sampled clips overlap each other and thus their representations are very similar, making it hard to correct mistakes in the prediction. When using a larger $\tau$, video-based accuracy also saturates at a smaller number of testing clips because one single clip already covers a large part of the video, as shown in Figure 4(b). However, as shown in Figure 4(a), the clip-based accuracy of R(2+1)D+DEM for $\tau = 8$ is nearly the same as that for $\tau = 2$, which indicates that R(2+1)D+DEM is able to generate accurate clip-level predictions within a wide range of temporal variations. The same conclusion should also stand for R3D+DEM. A more detailed discussion of the influence of temporal variations on action recognition is included in Section V.

Figure 5(a) shows the distribution of accuracy difference between R3D with and without DEM across all Kinetics-400 classes. Note that 208 out of 400 classes are predicted more accurately when using DEM. The top 10 and bottom 10 classes in terms of accuracy improvement due to the insertion of DEM are shown in Figure 6(a), from which it can be observed that the classes with the most prominent performance gain are more motion-related. For instance, correctly predicting class “washing hands” requires effective learning for motion clues that distinguish such an action from action classes taking place in similar environment, such as “taking a shower” and “washing head”. The classes with worse prediction accuracy after inserting DEM, however, are more appearance-driven in recognition or do not provide distinct motion clue, such as class “blowing nose” and “cutting nails”. It is also observed that the absolute value of the largest performance decrease (in class “blowing nose”) is much smaller than that of the largest performance increase (in class “reading book”) when using DEM. Figure 5(b) and Figure 6(b) show the same comparisons but with R(2+1)D network as the backbone, from which the same conclusion can be drawn. Note that 274 out of 400 classes are predicted more accurately when using DEM in this scenario.

**Results on UCF-101 and HMDB-51.** In order to demonstrate the generalization performance of the proposed module, we fine-tune R3D+DEM and R(2+1)D+DEM on UCF-101 and HMDB-51 datasets as well. The models are pre-trained on Kinetics-400. As recommended in [18], we only fine-tune conv5_x and the fully-connected layers as a trade-off between performance and time. Please note that Tran et al. [9] did not report the performance of R3D-18 and R(2+1)D-18 on these two datasets, so we use the weights pre-trained on Kinetics-400 provided by the authors and fine-tune them ourselves. The results are shown in Table 4. Both datasets contain 3 splits of the data and the reported accuracy is the average accuracy across these splits.

It can be observed that R3D with DEM outperforms the original version without DEM by 2.1% on UCF-101 and by 1.6% on HMDB-51. For R(2+1)D with DEM, the corresponding improvements are 2.4% and 5.7%. Although the performance of the original R(2+1)D is inferior to that of ARTNet [53] and STC [58], the model with DEM achieves better performance than STC and the best performance among the presented 18-layer models. This demonstrates that an addition of DEM contributes to spatio-temporal learning. It can be also concluded from these results that R3D+DEM and R(2+1)D+DEM generalize well to other datasets. Since the lower layers are frozen during fine-tuning, these results also indicate that the ability to handle temporal variations learned by DEM is transferable across different scenarios.

**V. ABLATION STUDY**

In order to further analyze the performance and benefits of DEM, we conduct a series of ablation experiments. All the experiments are performed on Kinetics-200 [24] with an R(2+1)D-18 backbone. The dataset contains 200 action classes with 80,000 and 5,000 videos for training and validation, respectively. Due to data deletion, our version consists of 77,152 videos in the training set and 4,988 videos in the validation set. We train all the models from scratch and adopt AdamW [59] optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.99$, initial learning rate of 0.001, and weight decay of 0.0001.

**Impact of Temporal Variations.** DEM is developed based on the assumption that temporal variations would lead to extra difficulties for backbone networks to learn from spatio-temporal data. To demonstrate such an influence of temporal variations, we train R(2+1)D with and without DEM under multiple combinations of $T \times \tau$. The results reported in Table 5 can be explained along three axes.

1) **Fixed time span:** When we fix the time span $T \cdot \tau$ across video clips, the corresponding elapsed time in the source videos that the clips come from is the same, e.g., 2 seconds (assuming all videos have the same frame rate). This corresponds to a similar temporal fragment of an action, e.g., a tennis racquet swing. In the experiment with time span of 16, we trained two models, one with subsampling rate of $\tau = 1$ and the other with $\tau = 2$. Compared with $\tau = 1$, the clips obtained for $\tau = 2$ contain a higher

| Time Span | $T \times \tau$ | w/o DEM | w/ DEM |
|-----------|----------------|---------|--------|
| 8         | 8 $\times$ 1   | 55.61   | 57.80  |
| 16        | 16 $\times$ 2  | 55.66   | 59.48  |
| 32        | 32 $\times$ 2  | 58.76   | 62.55  |

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/
level of inter-frame temporal variations since we drop every other frame during sampling. However, they still contain the same temporal range of a real-world action because the time span remains unchanged. We see in Table 5 that the performance of both R(2+1)D with and without DEM drops when \( \tau \) gets higher, thus demonstrating the impact of frame-level temporal variations on action recognition. However, R(2+1)D with DEM provides lower performance degradation (0.7%) than that of R(2+1)D without DEM (1.08%) when \( \tau \) increases from 1 to 2, which indicates DEM improves the ability of the backbone network to adapt to environment with stronger low-level temporal variations.

2) **Fixed subsampling rate:** When we fix \( \tau \) but increase \( T \), clips capture a larger elapsed time in the source video, which can be thought of as a larger temporal receptive field. This is expected to improve action recognition. However, a larger receptive field also leads to higher segment-level temporal variations since more components of an action are present in longer clips. This could have a negative impact on action recognition. Comparing the performance of R(2+1)D with and without DEM under \( T \times \tau = 8 \times 1 \) and \( T \times \tau = 16 \times 1 \), we find that both models achieve higher accuracy for larger \( T \), but the performance gain of the model with DEM (2.58%) is higher than that of the model without DEM (1.13%). This is also true for experiments with \( \tau = 2 \) (8 \times 2 versus 16 \times 2). These results demonstrate that DEM helps the backbone network to alleviate the influence of high-level temporal variations.

3) **Fixed video-clip length:** When we fix clip length \( T \) and increase \( \tau \), the computational cost per clip is fixed, but the inference process will be influenced by a complex combination of multiple factors explained in 1) and 2) above. This sampling scheme may facilitate video recognition due to a larger temporal receptive field per clip, but may also hinder accurate recognition because it introduces additional temporal variations, at both frame level and segment level. From Table 5, it can be observed that when \( T = 8 \) and \( \tau \) is increased from 1 to 2, the original R(2+1)D performs nearly the same, indicating the model suffers from the negative factors that neutralize the benefits of a largereceptive field. On the contrary, R(2+1)D with DEM achieves 1.68% performance gain. A similar observation can be made for the case of \( T = 16 \). These two groups of experiments demonstrate the benefits of DEM for video modeling with complex temporal variations and for more accurate action recognition without using more frames.

**Robustness of DEM to Temporal Variations.** The models trained and tested with the same temporal subsampling rate \( \tau \) are expected to achieve the best performance since training and testing are conducted on similar dynamics. In order to evaluate the DEM’s capability to generate a robust representation regardless of temporal variations, we use different \( \tau \) in training and testing. None of the models tested with a different \( \tau \) is fine-tuned in the new setting.

From the results shown in Table 6, it is clear that models with DEM outperform their counterparts without DEM when trained and tested with different value of \( \tau \), which induces different temporal variations. We also computed the difference between the accuracy of testing with one training \( \tau \) and the accuracy of testing with another \( \tau \), which reflects the model’s invariance to temporal variations. It can be seen that when increasing \( \tau \) from the original value 1 during training to 2, the performance of R(2+1)D without DEM decreases while that of the model with DEM is not affected. When \( \tau \) is increased to 4, although the performance of both models drops, the performance of the model with DEM decreases less. It is clear that when training and testing subsampling rates are mismatched, R(2+1)D with DEM has smaller changes in performance than the original R(2+1)D, which indicates a contribution of DEM to more robust spatio-temporal learning.

**At what depth to add DEM?** As discussed in Section III, we believe DEM, developed for Eulerian motion manipulation, should be able to handle both low-level and high-level temporal variations. To study its ability to deal with temporal variations at different levels, we add the module separately to each layer in R(2+1)D-18. Since higher layers have a larger temporal receptive field in the input, DEM inserted into higher layers should learn to handle inter-segment variations while those inserted into lower layers should mostly deal with low-level variations across a few frames only. Table 7 shows the corresponding experimental results.

![Table 6](image)

**Table 6. Impact of subsampling rate mismatch between training and testing evaluated on Kinetics-200 for R(2+1)D-18 with (w/) and without (w/o) DEM.** Using the accuracy of the model trained and tested with the same \( \tau \) as the reference, \( \Delta \text{Top-1}(\%) \) is calculated by testing the model at a different \( \tau \) and subtracting the accuracy from the reference value.

| Train \( T \times \tau \) | Test \( T \times \tau \) | Top-1 w/o | Top-1 w/ | \( \Delta \text{Top-1} \) w/o | \( \Delta \text{Top-1} \) w/ |
|--------------------------|------------------------|----------|--------|------------------|------------------|
| 8 \times 1               | 8 \times 1              | 55.61    | 57.80  | -                | -                |
| 8 \times 2               | 8 \times 2              | 55.11    | 57.84  | -0.50            | +0.04            |
| 8 \times 4               | 8 \times 4              | 52.66    | 55.35  | -2.95            | -2.45            |
| 16 \times 1             | 16 \times 1             | 56.74    | 60.18  | -3.35            | +0.08            |
| 16 \times 2             | 16 \times 2             | 56.43    | 60.26  | -0.31            | +0.08            |
| 16 \times 4             | 16 \times 4             | 53.37    | 57.32  | -3.35            | -2.86            |

![Table 7](image)

**Table 7. Impact of the insertion depth of DEM on Kinetics-200.** A checked box indicates the layer into which DEM was inserted.

| Layer | conv1 | conv2_x | conv3_x | conv4_x | conv5_x | Top-1 |
|-------|-------|---------|---------|---------|---------|-------|
|       | √     | √       | √       | √       | √       | 58.76 |
|       |       |         |         |         |         | 58.90 |
|       |       |         |         |         |         | 59.86 |
|       |       |         |         |         |         | 59.39 |
|       |       |         |         |         |         | 60.26 |
|       |       |         |         |         |         | 61.45 |
|       |       |         |         |         |         | 62.55 |
the performance improvement is more prominent at deeper layers, where filters have segment-level receptive field, DEM is more capable of handling high-level temporal variations. Table 7 also shows that R(2+1)D with DEM inserted into every layer provides the best prediction accuracy. Since conv1 is a simple (2+1)D layer, it is also reasonable that the improvement by adding a DEM only here is less impactful than by adding the same DEM to other layers (residual blocks with four (2+1)D layers in each).

**Visualizing the Impact of DEM.** Figure 7 illustrates the influence of DEM on video modeling. The shown feature maps have been collected at the output of layer conv2_1 in R(2+1)D with and without DEM. We chose one of the initial layers of the network for its higher resolution allowing easier interpretation. Since DEM manipulates motion information before temporal convolution in a (2+1)D layer, the differences between Figures 7(b) and 7(c), and those between Figures 7(e) and 7(f) reflect the impact of DEM.

The top image in column 7(a) shows a skier rotating with high acceleration. The corresponding feature map in column 7(c), generated by R(2+1)D, contains a “ghosting” effect, due to fast-rotating skis and ski poles, that is not useful for action recognition. However, in column 7(b) DEM successfully suppresses this effect and creates a sharply-outlined body and ski. A similar effect, caused by a fast-moving long stick, can be observed in the bottom image in column 7(a). Furthermore, examples in columns 7(d)-7(f) show that DEM can produce sharper boundaries of fast-moving people (top image in column 7(d)) or of body parts under partial occlusion (bottom image in column 7(d)), where a player’s arm is occluded by the basketball in several frames. Clearly, the “ghosting” effects, in case of thin objects, and blurred boundaries, in case of wider objects, result from temporal video variations and affect standard models, while DEM is capable of suppressing their influence on video representation learning.

**VI. CONCLUSIONS**

In this paper, we introduced a Dynamic Equilibrium Module (DEM), an insertable module for spatio-temporal convolutions. The module facilitates video modeling via Eulerian motion manipulation and alleviates the negative impact of temporal variations on action recognition. Extensive experiments on Kinetics-400, UCF-101, and HMDB-51 with R3D and R(2+1)D models demonstrate significant performance gains due to the use of DEM. Additional studies on Kinetics-200 with R(2+1)D model further illustrate the influence of temporal variations and the contributions of DEM towards more robust spatio-temporal learning.

**REFERENCES**

[1] Z.-L. Lu and G. Sperling, “The functional architecture of human visual motion perception,” *Vision Research*, vol. 35, no. 19, pp. 2697–2722, Oct. 1995.

[2] T. J. Buschman and E. K. Miller, “Top-Down Versus Bottom-Up Control of Attention in the Prefrontal and Posterior Parietal Cortices,” *Science*, vol. 315, no. 5820, pp. 1860–1862, Mar. 2007.

[3] A. Dosovitskiy, Philipp Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, “FlowNet: Learning Optical Flow with Convolutional Networks,” in *Int. Conf. Comput. Vis.* Santiago: IEEE, Dec. 2015, pp. 2758–2766.

[4] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, “FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks,” in *IEEE Conf. Comput. Vis. Pattern Recog.* Honolulu, HI: IEEE, Jul. 2017, pp. 1647–1655.

[5] A. Ranjan and M. J. Black, “Optical Flow Estimation Using a Spatial Pyramid Network,” in *IEEE Conf. Comput. Vis. Pattern Recog.* Honolulu, HI: IEEE, Jul. 2017, pp. 2720–2729.

[6] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, “PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume,” in *IEEE Conf. Comput. Vis. Pattern Recog.* Salt Lake City, UT: IEEE, Jun. 2018, pp. 8934–8943.

[7] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, “Learning Spatiotemporal Features with 3D Convolutional Networks,” in *Int. Conf. Comput. Vis.* Santiago, Chile: IEEE, Dec. 2015, pp. 4489–4497.

[8] J. Carreira and A. Zisserman, “Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset,” in *IEEE Conf. Comput. Vis. Pattern Recog.* Honolulu, HI: IEEE, Jul. 2017, pp. 4724–4733.

[9] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, “A Closer Look at Spatiotemporal Convolutions for Action Recognition,” in *IEEE Conf. Comput. Vis. Pattern Recog.* Salt Lake City, UT: IEEE, Jun. 2018, pp. 6450–6459.
