Causal Spatiotemporal Representations

Anonymous ICCV submission

Paper ID 810

Abstract

Recently, three dimensional (3D) convolutional neural networks (CNNs) have emerged as dominant methods to capture spatiotemporal representations, by adding to pre-existing 2D CNNs a third, temporal dimension. Such 3D CNNs, however, are non-causal (i.e., they exploit information from both the past and the future to produce feature representations, thus preventing their use in online settings), constrain the temporal reasoning horizon to the size of the temporal convolution kernel, and are not temporal resolution-preserving for video sequence-to-sequence modelling, as, e.g., in action detection. To address these serious limitations, we present a new architecture for the causal/online spatiotemporal representation of videos.

Namely, we propose a novel Recurrent Convolutional Network (RCN), which relies on recurrence to capture the temporal context across frames at every level of network depth. Our network decomposes 3D convolutions into (1) a 2D spatial convolution component, and (2) an additional hidden state $1 \times 1$ convolution applied across time. The hidden state at any time $t$ is assumed to depend on the hidden state at $t-1$ and on the current output of the spatial convolution component. As a result, the proposed network: (i) provides flexible temporal reasoning, (ii) produces causal outputs, (iii) preserves temporal resolution. Our experiments on the large-scale large “Kinetics” and “MultiThumos” datasets show that the proposed method achieves superior performance even when compared with non-causal 3D CNNs, while being causal and using fewer parameters.

1. Introduction

Convolutional neural networks (CNN) are starting to deliver gains in action recognition from videos similar to those previously observed in image recognition [24, 41] thanks to new 3D CNNs [5, 55, 49, 13, 54]. For instance, Hare et al. [13] have shown that this is the case for the 3D version of 2D residual networks (ResNets) [14]. Other recent works [55, 49] show that 3D convolutions can be decomposed into 2D (spatial) and 1D (temporal) convolutions (yielding the S3D architecture), and that these ‘decomposed’ architectures not only have fewer parameters to train [55], but also perform better than standard 3D (spatiotemporal) ones.

All such 3D CNNs, however, have significant issues. First and foremost, they are not causal [4], for they process future frames to predict the label of the current one. Causal inference is essential in many video understanding settings, e.g., online action prediction [35, 43, 45], future action label prediction [23], and future representation prediction [51]. Secondly, the size of temporal convolution needs to be specified by hand at every level of network depth, and is usually set to be equal to the spatial convolution size [5, 49] in all state-of-the-art 3D networks [5, 4, 48]. Whatever the choice, the temporal reasoning horizon or ‘receptive field’ is effectively constrained by the size of the temporal convolution kernel(s). Varol et al. [50] suggest that using long-term temporal convolution could enable long-term reasoning. However, fixing the size of the temporal convolution kernel at each level is a non-trivial task, which requires expert knowledge and extensive cross-validation.
Lastly, 3D CNNs do not preserve temporal resolution, as the latter drops with network depth. Preserving temporal resolution (i.e., ensuring the a prediction is made for each input frame), in opposition, is essential when predicting e.g. each individual video frame label for temporal action segmentation/detection [58, 39, 34, 3] or in video segmentation [57].

Our proposal: combining implicit and explicit temporal modelling. An alternative to the implicit modelling of a video’s temporal characteristics via 3D CNNs is the use of models which encode this dynamics explicitly. Hidden state models, such as Markov ones [2], recurrent neural networks (RNN) [18, 11], and long short-term memory (LSTM) networks [15] can all be used to model temporal dynamics in videos [10, 32], allowing flexible temporal reasoning without any need to specify a temporal window size.

In an approach which aims to combine the representation power of explicit dynamical models with the discriminative power of 3D networks, in this work we propose a recurrent alternative to 3D convolution illustrated in Figure 1(c). In this new architecture, spatial reasoning, expressed in the form of a mapping from the input to a hidden state, is performed by 2D convolution (with kernel size $\times$ $d \times d$), whereas temporal reasoning (represented by hidden state-to-hidden state transformations) is performed by a 1D convolution (with kernel size $1 \times 1 \times 1$) taking place at every point in time and at each level (depth-wise) of the network. In a setting which combines the effect of both operators, the hidden state at time $t$ (denoted by $h_t$) is a function of both the output of the spatial convolution and of the output of the temporal (hidden) convolution with $h_{t-1}$ as an input. As a result, the temporal reasoning horizon is effectively unconstrained, as the hidden state $h_t$ is a function of the input in the interval $[0, t]$.

Causality. Contrarily to 2D CNNs [40], which are both causal and preserve temporal resolution but do not perform temporal reasoning using the context provided by neighbouring frames, Carreira et al. [4] propose to solve this problem by means of a sequential and parallel causal 3D network which uses present and past frames for predicting the action class of the present frame. However, performance is observed dropping in both cases as compared to the counterpart 3D CNN version [5] of the same network [4].

Our proposed method, in opposition, solves both problems via a recurrent convolutional network (RCN) which explicitly performs temporal reasoning at each level of the network thanks to recurrence, while maintaining temporal resolution and being causal, without decline in performance.

Transfer learning and initialisation. The ability of a network to be conferred knowledge acquired by solving other tasks has been proved to be crucial to performance. Famously, when Tran et al. [48] and Ji et al. [16] first proposed 3D CNNs for video action recognition their observed performance turned out to be merely comparable to that of 2D CNNs [40], e.g., on the Sports1M dataset [20]. For these reasons, Carreira et al. [5] later proposed to use transfer learning to boost 3D CNN performance. There, 2D CNNs are inflated into 3D ones by replacing 2D convolutions with 3D convolution: as a result, 2D network weights as pretrained on ImageNet [9] can be used to initialise their 3D CNNs. This makes the use of 3D CNNs more widely accessible, for training a full 3D CNN is a computationally expensive task: 64 GPUs were used to train the latest state-of-the-art 3D nets [5, 49, 4], a kind of firepower not accessible to everyone. That makes ImageNet-based initialisation even more crucial for speeding up the training process of parameter-heavy 3D networks.

Unlike Tran et al. [49], where the number of filters changes, our Recurrent Convolutional Network (RCN) exhibits similar performance improvement gains when it comes to ImageNet initialisation as those of inflated 3D CNNs (I3D) [5]. Interestingly, Le et al. [25] show that simple RNNs can exhibit long-term memory properties if appropriately initialised, even more so than LSTMs. We thus follow [25] and initialise our hidden state-to-hidden state convolution (kernel size $N \times N \times 1 \times 1 \times 1$) by the identity matrix, where $N$ is number of hidden state kernels.

Contributions: we present a new approach to video feature representation based on an original convolutional network with recurrent hidden states at each depth level, which:

- allows flexible temporal reasoning, as it exploits by design information coming from all the input sequence observed up to time $t$;
- generates output representations in a causal way, allowing online video processing and enabling the use of 3D networks in scenarios in which causality is key;
- is designed to directly benefit from model initialisation via ImageNet pre-trained weights, as opposed to state of the art approaches, and in line with clear emerging trends in the field.

In our experiments we show that our proposed RCN outperforms baseline 1D and (2+1)D models, while dispalying all the above desirable properties.

2. Related Work

Since the two-stream 2D CNNs proposed by Simonyan et al. [40] produced performances comparable to that of traditional features such as IDT [52], HOG [7], HOG3D [22], HOF [8], 2D features has been extensively used in action recognition and detection. Efforts have also been made to better capture temporal information. For instance, Donahue used LSTMs [10] on top of 2D CNN features. Wang et al. [53] proposed to train 2D CNNs with segment-level inputs. Other approaches include, among others, CNN fea-
tures in combination with LSTMs [27] for temporal action detection, 2D features used in an encoder-decoder setup along with temporal convolutions [34], and conditional random field on series of 2D features [38] for temporal action detection and recognition. All these methods showed promising results; in all of them, however, the optical stream and the few layers on the top of the 2D features are the only sources of temporal reasoning.

As mentioned in the Introduction, initial 3D CNNs models [16, 48], which promised to be able to perform spatial and temporal reasoning in parallel but with limited success, were improved by Carreira et al. [5] using ImageNet based initialisation and training on the large scale Kinetics dataset [21]. The resulting models outperformed 2D ones. Nevertheless, 3DCNNs remain heavy and very expensive to train – e.g., 64 GPUs were used in [5].

In alternative, the notion of factorising 3D convolutional networks was explored by Sun et al. [46]. This inspired [55, 33, 49] to decompose 3D convolutions into 2D (spatial) and 1D (temporal) convolutions. Recent work by Xie et al. [55] have promised to reduce complexity (the number of parameters) while making up for the performance lost via a gating mechanism. Tran et al. [49] would keep the number of parameters equal to that of 3D convolutions, but boost performance by increasing the number of kernels in the 2D layer. The size of the temporal convolution kernel needs to be fixed to a relatively small number (e.g., 3 in [5, 55, 49]). Varol et al. [50] have thus proposed the use of long-term convolutions to capture long-range dependencies in the data. Wang et al. [54], instead, have introduced non-local blocks in existing 3D CNN architectures, to capture the non-local context in both the spatial and the temporal (present, past and future) dimensions.

The use of temporal convolutions in all the above methods is, however, inherently non-causal. Moreover, temporal context is limited by the size of the temporal convolution kernel or of the non-local step. Also, temporal resolution is not preserved in temporal convolutions with strides: to address this problem [37, 34] uses temporal deconvolution layers on top of C3D network [48] to produce a one-to-one mapping from input frames to corresponding frame-level label prediction for temporal action detection. Relevantly, Carreira et al. [4] have proposed to address the non-causal nature of 3D CNNs by predicting the future and utilising the flow of information in the network. They train their causal network to mimic a 3D network – however, the resulting performance drop is significant.

Recurrent convolutions have indeed been tried for image generation [19, 29], scene labeling [31], scene text recognition [36] and video representations [56]. In particular, the convolutional LSTM (C-LSTM) model proposed in [56] for precipitation forecasting is closely related to our work. C-LSTM has been applied to videos [26] to capture spatial attention on top of 2D feature maps. However, its performance has turned out to be sub-par as compared to that of 2D CNNs. Nevertheless, recurrent convolutions remain interesting from a causality point of view.

### 3. From 2D to 3D CNNs

There are two main reason why 3D CNNs [5, 49, 55, 54] evolved from 2D CNNs [41, 53] perform better than 3D CNNs built from scratch [48, 16]. Firstly, 2D CNNs are well tried and tested on the problem of image recognition and a video is, after all, a sequence of images – hence, transferability makes sense. Secondly, initialisation from a good starting guess leads to better convergence in videos [5], since the number of parameters in 3D networks is huge.

In this section we recall the two basic types of 3D CNNs that can be built using a 2D CNN architecture. We will use them as baselines in our experiments.

<table>
<thead>
<tr>
<th>Layers</th>
<th>Output Sizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Names n, d, number of kernels</td>
<td>I3D</td>
</tr>
<tr>
<td>conv1 3, 7, 64; stride 1, 2, 2</td>
<td>16 × 56 × 56</td>
</tr>
<tr>
<td>res2 [3, 3, 64 &amp; 3, 3, 64] × 2</td>
<td>16 × 56 × 56</td>
</tr>
<tr>
<td>res3 [3, 3, 128 &amp; 3, 3, 128] × 2</td>
<td>8 × 28 × 28</td>
</tr>
<tr>
<td>res4 [3, 3, 256 &amp; 3, 3, 256] × 2</td>
<td>4 × 14 × 14</td>
</tr>
<tr>
<td>res5 [3, 3, 512 &amp; 3, 3, 512] × 2</td>
<td>2 × 7 × 7</td>
</tr>
<tr>
<td>pool spatial pooling</td>
<td>2 × 1 × 1</td>
</tr>
<tr>
<td>mean temporal pooling</td>
<td>2 × C</td>
</tr>
<tr>
<td>convC classification; 1, 1, C</td>
<td>2 × C</td>
</tr>
</tbody>
</table>

Table 1. I3D ResNet-18 model architecture with its output sizes vs RCN’s output sizes for input of size 16 × 112 × 112. Each convolutional layer of the network is defined by the temporal (n) and the spatial (d) size of the kernels and by the number of kernels. The ConvC layer uses the number of classes as number of kernels.

### 3.1. Inflated 3D Network

A 2D network can be converted/inflated into a 3D one by replacing a 2D (d × d) convolution with a 3D (n × d × d) one as shown in Figure 1(a). Usually, the kernel’s temporal dimension n is set to be equal to the spatial dimension d, as in the inflated 3D (I3D) [5] or the convolutional 3D (C3D) [48] architectures. Here, we inflate the 18-layer ResNet [14] network into an I3D one as shown in Table 1, where each 2D convolution is inflated into a 3D convolution. Similarly to the I3D network in [5], a convolutional layer is used for classification, instead of the fully connected layer used in [49, 54]. A convolutional classification layer allows us to evaluate the model on sequences of variable length at test time. In this way, video-level results can be obtained in a seamless fashion, as opposed to having to compute clip-level outputs in a sliding window fashion to obtain the video-level outcome.
3.2. Separated Convolution Networks

Figure 1(b) shows how a 3D \((n \times d \times d)\) convolution can be decomposed into a \((1 \times d \times d)\) spatial convolution and a \((n \times 1 \times 1)\) temporal one. Usually, the size \(n\) of the temporal kernel is set to be equal to its spatial dimension \(d\), as in both I3D [5] and C3D [48].

Such a separated convolutional network (S3D) was introduced by Xie et al. [55]. The authors showed that such a decomposition not only reduces the number of parameters, but also delivers performances very much comparable to those of traditional 3D CNNs. After taking a closer look at 3D convolution separation, Tran et al. [49] argued that if the number of kernels \(M_i\) used in spatial convolution (Figure 1(b)) are increased in such way that the numbers of parameters of spatial and temporal convolution combined are equal to the number of parameters in 3D convolution, then performance actually improves over 3D networks. However, such a change in the number of kernels does not allow initialisation from ImageNet pre-trained models any longer. They refer to their model as \((2 + 1)D\) model. Although the latter can be considered a special case of pseudo-3D network (P3D) [33] models, because of its homogeneity and simplicity the \((2+1)D\) model performs better than P3D.

We re-implemented \((2+1)D\) without ImageNet initialisation as an additional baseline, as it shows the most promising results without using any additional tricks, such as e.g. gating in S3Dg [55].

4. Recurrent Convolutional 3D Network

We are now ready to describe the architecture of our Recurrent Convolutional (3D) Network (RCN) and its properties in detail. Firstly, we show how Recurrent Convolutional Units (RCUs) (§ 4.1) are used to replace 3D convolutions in the I3D network (§ 3.1), resulting in our RCN model (§ 4.2). Next, we show how RCUs preserve temporal resolution in § 4.3. Then in § 4.4 we show how our network behaves in a causal manner. Lastly, in § 4.5 and § 4.6, we illustrate the initialisation process for RCN and RCU.

4.1. Recurrent Convolutional Unit

A pictorial illustration of our proposed Recurrent Convolutional Unit (RCU) is given in Figure 1(c). The input at any time instant \(t\) passes through 3D spatial convolution (with kernel of size \(1 \times d \times d\), denoted by \(w_{zh}\)). The result is added to the output of a recurrent convolution operation, with kernel denoted by \(w_{hh}\), of size \(1 \times 1 \times 1\).

The result is termed the hidden state \(h_t\) of the unit. Analytically, a recurrent convolutional unit can be described by the following relation:

\[
    h(t) = h_{t-1} \ast w_{hh} + x_t \ast w_{zh},
\]

where \(w_{hh}\) and \(w_{zh}\) are parameters of the RCU, and \(\ast\) denotes the convolution operator.

4.2. Unrolling a Recurrent Convolutional Network

Figure 2 represents a simple recurrent convolutional network (RCN) composed by a single RCU unit, unrolled up to time \(t\). At each time step \(t\), an input \(x_t\) is processed by the RCU and the other layers to produce an output \(y_t\).

The unrolling principle allows us to build an RCN from 2D/3D networks, e.g. by replacing 3D convolutions with RCUs in any I3D network. Indeed, the network architecture of our proposed model builds on the I3D network architecture shown in Table 1, where the same parameters \((d, \text{number of kernels})\) used for 3D convolutions are used for our RCU. Unlike I3D, however, our RCN does not require a temporal convolution size \(n\) (cfr. Table 1) as a parameter.

As in 2D or 3D ResNet models [14, 49, 53], our proposed RCN also has residual connections. The hidden state \(h_t\) at time \(t\) is considered to be the output at that time instant – as such, it acts as input to next hidden state and to the whole next depth-level layer. Table 1 describes the network architecture of ResNet-18 [14] with 18 layers. Similarly, we can build upon other variants of ResNet.

4.3. Temporal Resolution Preservation

The output sizes for both I3D and our proposed RCN are shown in Table 1. Our RCN only uses spatial pooling and a convolutional layer for classification, unlike the spatiotemporal pooling of [5, 54, 49]. From Table 1, compared to I3D, RCN produces 16 classification score vectors with an input sequence length of 16.

This one-to-one mapping from input to output is essential in many tasks, ranging from temporal action segmentation [37, 34], to temporal action detection [42], to action tube detection [43]. In all such tasks, video-level accuracy is not enough, but we need frame-level results in terms, e.g., of
detection bounding boxes and class scores. Temporal convolution behaves in a similar way to spatial convolution: it results in lower resolution feature maps as compared to the input as the depth of the network increases.

Unlike the temporal deconvolution proposed in [37, 34], our RCN inherently addresses this problem (see Table 1). For fair comparison, in our tests we adjusted our baseline 3D model for dense prediction, by setting the temporal stride to 1 (§5.4). The resulting 3D model can produce dense predictions: given $T$ frames as input, it will generate $T$ predictions in 1-1 correspondence with the input frames.

### 4.4. Causality and Longer-Term Dependencies

A size-$n$ temporal convolution operation uses a sequence $x_{t-n/2}, \ldots, x_t, \ldots, x_{t+n/2}$ as input to generate an output $y_t$ at time $t$. In our Recurrent Convolutional Network, instead, $y_t$ is a function of only the inputs $x_0, x_1, \ldots, x_t$ from the present and the past (up to the initial time step), as shown in Figure 2. Its independence from future inputs makes the output $y_t$ at time $t$ causal. Thus RCN as presented here is not only causal, but poses no constraints on the modelling of temporal dependencies (as opposed to an upper bound of $n$ in the case of temporal convolutions). Temporal dependencies are only limited by the input sequence length at training time.

As in traditional RNNs, we have the option to unroll the same network to model arbitrary input sequence lengths at test time, thus further increasing the horizon of temporal dependencies and show a substantial gain in performance.

### 4.5. ImageNet Initialisation for the 2D Layers

The 3D model proposed by Carreira et al. [5] greatly owes its success to a good initialisation from 2D models trained on ImageNet [9]. By inflating these 2D models, we can benefit from their ImageNet pre-trained weights, as in most state-of-the-art 3D models [5, 55, 54]. We follow the same principle and initialise all 2D layers using the weights of available pre-trained 2D ResNet models [14]. It is noteworthy that the other state-of-the-art (2+1)D model by Tran et al. [49] cannot, instead, exploit ImageNet initialisation, because of the change in the number of kernels.

### 4.6. Identity Initialisation for the Hidden Layers

The presence of a hidden state convolution ($w_{hh}$, see Figure 2) layer at every depth level of the unrolled network makes initialisation a tricky issue. The random initialisation of the hidden state convolution component could destabilise the norm of the feature space between two 2D layers. In response to a similar issue, Le et al. [25] presented a simple way to initialise RNNs when used with ReLU [12] activation functions. Most state-of-the-art 2D models [14, 47] make indeed use of ReLU as activation function of choice for fast and optimal convergence [12].

---

**Table 2. Clip-level and video-level action recognition accuracy on the validation set of the Kinetics dataset for different ResNet18 based models, trained using 8-frame-long clips as input.**

Following the example of Le et al. [25] and others [28, 44], we initialise the weights of the hidden state convolution kernel ($w_{hh}$) with the identity matrix. Identity matrix initialisation is shown [25, 28] to capture longer term dependencies. It also helps induce forgetting capabilities in recurrent models, unlike traditional RNNs.

### 5. Experiments

In this section, we evaluate our RCN on the challenging Kinetics [21] and MultiThumos [58] datasets to answer the following questions: i) How does our training setup, which uses 4 GPUs, compare with the 64 GPU training setup of [48]? (Section 5.2) ii) How do recurrent convolutions compare against 3D convolutions in the action recognition problem (§5.3)? iii) How does our RCN help solving the dense prediction task associated with action detection (§5.4)? Finally, iv) we validate our claims on the temporal causality and flexibility of RCN, and check whether those features help with longer-term temporal reasoning (§5.5).

The Kinetics dataset comprises 400 classes and 260K videos; each video contains a single atomic action. Kinetics has become a de facto benchmark for recent action recognition works [4, 55, 49, 5, 54]. The average duration of a video clip in Kinetics is 10 seconds. The MultiThumos [58] dataset is a multilabel extension of THUMOS [17]. It features 65 classes and 400 videos, with a total duration of 30 hours. On average, it provides 1.5 labels per frame, 10.5 action classes per video. Each video can be up to 30 minutes long, in contrast with the Charades [39] dataset. Videos are densely labeled, as opposed to those in THUMOS [17] or ActivityNet [3]. MultiThumos allows us to show the dense prediction capabilities of RCN on long, real-world videos.

#### 5.1. Implementation Details

In all our experiments we used sequences of RGB frames as input for simplicity and computational reasons. All the videos are sampled at a rate of 25 frames per second.

We used the ResNet [14] model as backbone architecture for our experiments, for reasons of simplicity and scalability. We used a batch size of 64 when training ResNet18-
based models, and of 32 for models based on ResNet-34 and -50. The initial learning rate was set to 0.01 for batches of 64, and to 0.005 for batches of 32. We reduced the learning rate by a factor of 10 after both 250\(K\) and 350\(K\) iterations. Moreover, training was stopped after 400\(K\) iterations (number of batches). We used standard data augmentation techniques, such as random crop, horizontal flip with 50% probability, and temporal jittering. More details about training parameters are provided in the supplementary material.

**GPU memory** consumption plays a crucial role in the design of neural network architectures. In our training processes a maximum of 4 GPUs were used. Given our GPU memory and computational constraints, we only considered training networks with 8-frame long input clips, except for ResNet34 which was trained with 16 frames long clips.

**Evaluation:** For fair comparison, we computed clip-level and video-level accuracy in the way described in [49, 54]. Ten regularly sampled clips were evaluated per video, and scores were averaged for video-level classification. On videos of arbitrary length we averaged all the predictions made by the unrolled versions of both our RCN and of I3D.

### 5.2. Training Setup Comparison

The main hyperparameters involved in the training of a 3D network are learning rate, batch size, and number of iterations. These parameters are interdependent, and their optimal setting depends on the computational power at disposal. For instance, Tran et al. [48] would use 64 GPUs, with the training process distributed across multiple machines. In such a case, when vast computational resources are available[48, 5, 4], training takes 10-15 hours [49], allowing for time to identify the optimal parameters. The availability of such computational power, however, is scarce.

In a bid to reproduce the training setup of [48] on 4 GPUs, we re-implemented the I3D and (2+1)D models using ResNet18 and ResNet34 as a backbone. The ResNet18-I3D architecture is described in Table 1. Based on the latter, we built a (2+1)D [49] architecture in which we matched the number of parameters of separated convolutions to that of standard 3D convolutions, as explained in [49].

The results of the I3D and (2+1)D implementations reported in Tran et al. [48] are shown in the top half of Table 2. When comparing them with our implementations of the same networks in the bottom half, it is clear that our training is as performing as that of Tran et al. [49]. This allows a fair comparison of our results.

**Why smaller clips as input:** training a ResNet18-based model on Kinetics with 8 frame clips as input takes up to 2-3 days on 4 GPUs. Training a ResNet50-based model would take up to 4-5 days. In principle, one could train the same model for longer input clip sizes, but the amount of GPU memory and time required to train would grow lin-

<table>
<thead>
<tr>
<th>Model</th>
<th>Clip-length</th>
<th>Initialisation</th>
<th>Acc%</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>ResNet34-(2+1)D [49]</em>†</td>
<td>16</td>
<td>random</td>
<td>67.8</td>
</tr>
<tr>
<td><em>ResNet34-I3D [5]</em></td>
<td>16</td>
<td>ImageNet</td>
<td>68.2</td>
</tr>
<tr>
<td><em>ResNet34-RCN [ours]</em>†</td>
<td>16</td>
<td>ImageNet</td>
<td>70.3</td>
</tr>
<tr>
<td><em>ResNet50-I3D [5]</em></td>
<td>8</td>
<td>ImageNet</td>
<td>68.8</td>
</tr>
<tr>
<td><em>ResNet50-RCN [ours]</em>†</td>
<td>8</td>
<td>ImageNet</td>
<td>71.2</td>
</tr>
<tr>
<td><em>ResNet50-RCN-unrolled [ours]</em>†</td>
<td>8</td>
<td>ImageNet</td>
<td>72.2</td>
</tr>
</tbody>
</table>

* causal model; † trained with our implementation and training setup.

Table 3. Video-level action classification accuracy of different models on the validation set of the Kinetics dataset.

early. As an estimate, it would take more than two weeks to train a ResNet50 model on 64 long frame clips, assuming that all the hyperparameters are known (i.e., batch size, learning rate, step size for learning rate drop, and whether batch normalisation layers should be frozen or not).

For these reasons we stick to smaller input clips to train our models in a fair comparison setting, using the hyperparameter values from § 5.1.

### 5.3. Results on Action Recognition

We compared our RCN with both I3D and (2+1)D models in the action recognition problem on the Kinetics dataset. A fair comparison is shown in Table 2 with ResNet18 as backbone architecture. Table 3 shows the results with ResNet34 and ResNet50 as backbone, trained on 16 frame and 8 frame clips, respectively. It is clear from these figures that RCN significantly outperforms state-of-the-art 3D networks – e.g. our network outperforms the equivalent I3D network by more than 2% across the board.

The ability to model long-term temporal reasoning of RCN is attested by the performance of the unrolled version (last row of Table 3). It shows that, even though the network is trained on input clip of 8 frames, it can reason over longer temporal horizons at test time. The corresponding unrolled I3D version (the last classification layer is also convolutional, see Table 1) showed no substantial improvement in performance – in fact, a slight drop.

**Comparison with I3D variants:** the main variants of the I3D model are separated 3D convolutions with gating (3Dg) [55] and with non-local operators (NL) [54]. We think it appropriate to take a closer look at these variants of I3D as they provide state-of-the-art performance, albeit being non-causal. In [55, 54] the application of non-local or gating operations to I3D yields the best performances to date, mainly thanks to training on longer clips given the large amount of GPU memory at their disposal (S3Dg [55] models are trained using 56 GPUs, [54] uses 8 GPUs with 16GB memory each). The best version of I3D-NL achieves an accuracy of 77.7%, but uses 128 frames and ResNet101 as backbone network; hence we do not deem fair to compare it with our models (which only use 8 frame long clips). It would take almost a month to train such a network using 4 GPUs. What needs to be stressed is that gating and NL op-
5.4. Results on Temporal Action Detection

We also evaluate our model on the temporal action detection problem on the MultiThumos [58] dataset. The latter is a dense label prediction task. As a baseline, we use the temporal resolution preserving version of I3D introduced in Section 4.3. ResNet50 is employed as a backbone for both our RCN and the baseline I3D. To capture the longer duration, we use 16 frame clips as input; sampling period is 4 frames. Both networks are initialised with the respective models pretrained on Kinetics. The initial learning rate is set to 0.001 and dropped after 14K iterations, a batch size of 16 is used, and trained up to 20K iterations. Similar to [30], we use binary cross-entropy as loss function.

We use the evaluation setup of [58] and [30], and computed both frame-wise mean Average Precision at 1 (mAP@1) and every 8th frame (mAP@8).

<table>
<thead>
<tr>
<th>Network</th>
<th>Input</th>
<th>mAP@1 %</th>
<th>mAP@8 %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two-stream+LSTM [58]*</td>
<td>RGB+FLOW</td>
<td>28.1</td>
<td>-</td>
</tr>
<tr>
<td>MultiLSTM [58]</td>
<td>RGB+FLOW</td>
<td>29.7</td>
<td>-</td>
</tr>
<tr>
<td>Inception-I3D by [30]</td>
<td>RGB+FLOW</td>
<td>-</td>
<td>30.1</td>
</tr>
<tr>
<td>Inception-I3D + SE [30]</td>
<td>RGB+FLOW</td>
<td>-</td>
<td>36.2</td>
</tr>
<tr>
<td>ResNet50-I3D [baseline]</td>
<td>rgb</td>
<td>34.8</td>
<td>36.9</td>
</tr>
<tr>
<td>ResNet50-RCN (ours)*</td>
<td>rgb</td>
<td>35.3</td>
<td>37.3</td>
</tr>
<tr>
<td>ResNet50-RCN-unrolled (ours)*</td>
<td>rgb</td>
<td>36.2</td>
<td>38.3</td>
</tr>
</tbody>
</table>

* causal model

Table 4. Action detection/segmentation results on Multithumos dataset, mAP computed from dense prediction at every frame (mAP@1) and every 8th frame (mAP@8).

Table 4 shows the performance of our models along with that of other state-of-the-art methods. Two LSTM-based causal models presented by [58] are shown in rows 1 and 2. Piergiovanni et al. [30] use pre trained I3D [5] to compute features, but do not train I3D end-to-end, hence their performance is lower than in our version of I3D. Our RCN outperforms all other methods, including non-causal I3D+Super-Events (SE) [30] and the I3D baseline. It is safe to say that RCN is well applicable to dense prediction tasks as well.

5.5. Causality and Temporal Reasoning

A comparison with other causal methods is a must, as we claim the causal nature of the network to be the main contributions of our work, making RCN best suited to online applications such as action detection and prediction. In Section 5.4 we have already shown that our model excels in the task of temporal action detection.

Carreira et al. [4] proposed two causal variants of the I3D network. Their sequential version of I3D, however, shows a slight drop [4] in performance as compared to I3D. Their parallel version is much faster than the sequential one but suffers from an even more significant performance decline of 71.8% to 54.5% [4].

In contrast, our causal/online model not only outperforms other causal models (see Table 4) but beats as well strong, inherently non-causal state-of-the-art 3D networks on a large scale dataset such as Kinetics (see Table 3).

In addition, as in [35, 43], we can use the early action prediction task to evaluate the sequential temporal reasoning of RCN. The task consists in guessing the label of an entire action instance (or video, if containing a single action) after observing just a fraction of video frames. Accumulated output scores up to time $t_1$ are used to predict the label of the entire video. The idea is that a system should improve its classification accuracy as it observes a larger fraction of the input video, by exploiting temporal context.

Figure 3(a) shows that our RCN improves drastically as more video frames are observed, when compared to I3D. It indicates that RCN has superior anticipation ability, albeit starting slowly in first 10% of the video.

Furthermore, to provide useful cues about causality and temporal reasoning, we designed an original segment-level classification evaluation setting. Namely, the outputs of the models being tested are divided into ten regularly sampled segments and the difference between the accuracy for each segment and that for the first segment is computed, as shown in Figure 3(b). Within this setting, we compared the I3D baseline with RCN in two different modalities, one considering clip-wise outputs in a sliding window fashion, the other obtained by unrolling both I3D and RCN over test videos of arbitrary length.

Notably, middle segments provide the best relative improvement, which is reasonable as it indicates that the middle part of the video is the most informative. Secondly, the last segment (no. 10) has the lowest relative accuracy of all, except for RCN-unrolled. The relative accuracy of a pure causal system, though, should improve monotonically, i.e., exploit all it has seen. Instead, all compared models end up at the same performance they started with, except
for unrolled RCN for which the final accuracy is almost 5% higher than the initial one. We can conclude that unrolled RCN has longer-term memory than unrolled I3D or both sliding window-based I3D/RCN.

Evolution of Recurrence with Network Depth is another aspect that can provide clues about RCN’s temporal flexibility. To this purpose, we examine the statistics of the weight matrices \( w_{hh} \) associated with the hidden state at every RCU layer in the RCN network.

In Figure 4(a) we can see that the mean of the diagonal elements of the weight matrices increases and their standard deviation decreases with the depth of the network. This means that the \( w_{hh} \) matrix becomes sparser as network depth grows. In our view, this phenomenon is associated with RCN putting more focus on feature learning in the early part of the network, and emphasising temporal reasoning at later depths as the temporal reasoning horizon (‘receptive field’) increases. In other words, RCN learns to select the layers which should contribute towards temporal reasoning automatically.

Arjovsky [1] argue that if the eigenvalues of the hidden state-to-hidden state weight matrix diverge from the value 1, optimisation becomes difficult due to the vanishing gradient problem. Chang et al. [6] explore a similar idea. Taking an ordinary differential equation view of RNNs, they argue that their stability for long-term memory is related to the eigenvalues of the weight matrices. Figure 4(b) shows that in RCN the mean eigenvalue does rise towards 1 as network depth increases, suggesting that later layers are more stable in terms of long-term memory whereas earlier layers are not concerned with long-term reasoning.

5.6 Discussion

In the light of RCN’s superior results on temporal action detection (§ 5.4), early action prediction (see Figure 3(a)), long-term temporal reasoning (in its unrolled incarnation) at segment-level and for action recognition (§ 5.3, see the last row of Table 3), it is fair to say that the proposed Recurrent Convolutional Network is the best performing causal network out there. An even more in-depth analysis of this fact is conducted in the supplementary material.

The number of parameters in our proposed RCN model is 12.8 million (M), as opposed to 33.4M in both the I3D and (2+1)D models, see Table 2. It is remarkable to see that, despite a 2.6 times reduction in the number of parameters, RCN still outperforms both I3D and (2+1)D when trained using ImageNet initialisation. Further, RCN surpasses I3D also under random initialisation, while using 2.6 times fewer model parameters. We also measured the floating point operations (FLOPs) for I3D, R(2+1)D, and RCN, recording 41MMac, 120MMac, and 54MMac, respectively. Thus, RCN requires half the FLOPs as compared to (2+1)D, and is comparable to I3D. As RCN preserves temporal resolution, it requires slightly more FLOPs than I3D.

ImageNet initialisation proves to be useful for both the I3D and our RCN models. While (2+1)D performs (Table 2, row 6) better than RCN (row 5) with random initialisation, our RCN recovers to improve over (2+1)D (row 6) with ImageNet initialisation, whereas (2+1)D cannot make use of free ImageNet initialisation. This seems to be a severe drawback for the (2+1)D model, and a big advantage for I3D and RCN. One may argue that, if the purpose is to build on existing 2D models, then RCN and I3D are a better choice, whereas if new 3D models are preferred then (2+1)D might prove useful. The latter does provide better performance with random initialisation, but at the price of requiring many more parameters than RCN.

Random initialisation for hidden state parameters resulted in unstable training. Thus, in all the experiments with RCN, we used identity matrix initialisation instead. Identity matrix initialisation helps capturing forget capabilities as well, as suggested by Le et al. [25]. An ablation study on the effect of initialisation is provided in supplementary material.

6. Conclusions

In this work, we presented a recurrence-based convolutional network (RCN) able to generate causal spatiotemporal representations while using 2.6 times fewer parameters compared to its traditional 3D counterparts. RCN can model long-term temporal dependencies without the need to specify temporal extents. The proposed RCN is not only causal in nature and temporal resolution-preserving, but was also shown to outperform the main baseline 3D networks in all the fair comparisons we ran. We showed that ImageNet-based initialisation is at the heart of the success of 3D CNNs. Indeed, although RCN is recurrent in nature, it can still utilise the weights of a pre-trained 2D network for initialisation. The causal nature of our recurrent 3D convolutional network opens up manifold research directions (including its combination with non-local or gating methods), with direct and promising potential application in areas such as online action detection and future event prediction.
References


ings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5304–5313, 2018. 7


[45] K. Soomro, H. Idrees, and M. Shah. Predicting the where and what of actors and actions through online action localization. 2016. 1


