arXiv:1711.07971v3 [cs.CV] 13 Apr 2018

Non-local Neural Networks

Xiaolong Wang[1,2*] Ross Girshick[2] Abhinav Gupta[1] Kaiming He[2]
[1]Carnegie Mellon University [2]Facebook AI Research

Abstract

Both convolutional and recurrent operations are building blocks that process one local neighborhood at a time. In this paper, we present non-local operations as a generic family of building blocks for capturing long-range dependencies. Inspired by the classical non-local means method [4] in computer vision, our non-local operation computes the response at a position as a weighted sum of the features at all positions. This building block can be plugged into many computer vision architectures. On the task of video classification, even without any bells and whistles, our non-local models can compete or outperform current competition winners on both Kinetics and Charades datasets. In static image recognition, our non-local models improve object detection/segmentation and pose estimation on the COCO suite of tasks. Code is available at https://github.com/facebookresearch/video-nonlocal-net.

1. Introduction

Capturing *long-range* dependencies is of central importance in deep neural networks. For sequential data (*e.g.*, in speech, language), *recurrent* operations [38, 23] are the dominant solution to long-range dependency modeling. For image data, long-distance dependencies are modeled by the large receptive fields formed by deep stacks of *convolutional* operations [14, 30].

Convolutional and recurrent operations both process a *local* neighborhood, either in space or time; thus long-range dependencies can only be captured when these operations are applied repeatedly, propagating signals progressively through the data. Repeating local operations has several limitations. First, it is computationally inefficient. Second, it causes optimization difficulties that need to be carefully addressed [23, 21]. Finally, these challenges make multi-hop dependency modeling, *e.g.*, when messages need to be delivered back and forth between distant positions, difficult.

In this paper, we present *non-local* operations as an efficient, simple, and generic component for capturing long-range dependencies with deep neural networks. Our proposed non-local operation is a generalization of the classical non-local mean operation [4] in computer vision. Intuitively, a non-local operation computes the response at a position as a weighted sum of the features at *all positions* in the input feature maps (Figure 1). The set of positions can be in space, time, or spacetime, implying that our operations are applicable for image, sequence, and video problems.

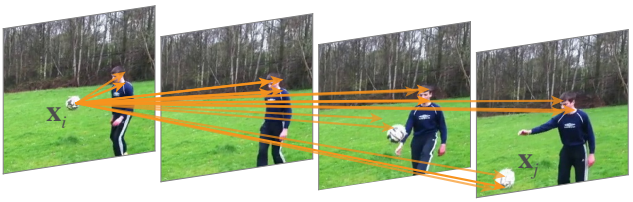

Figure 1. A spacetime ***non-local*** operation in our network trained for video classification in Kinetics. A position $\mathbf{x}_i$'s response is computed by the weighted average of the features of *all* positions $\mathbf{x}_j$ (only the highest weighted ones are shown here). In this example computed by our model, note how it relates the ball in the first frame to the ball in the last two frames. More examples are in Figure 3.

There are several advantages of using non-local operations: (a) In contrast to the progressive behavior of recurrent and convolutional operations, non-local operations capture long-range dependencies directly by computing interactions between any two positions, regardless of their positional distance; (b) As we show in experiments, non-local operations are efficient and achieve their best results even with only a few layers (*e.g.*, 5); (c) Finally, our non-local operations maintain the variable input sizes and can be easily combined with other operations (*e.g.*, convolutions as we will use).

We showcase the effectiveness of non-local operations in the application of video classification. In videos, long-range interactions occur between distant pixels in space as well as time. A single non-local block, which is our basic unit, can directly capture these spacetime dependencies in a feedforward fashion. With a few non-local blocks, our architecures called *non-local neural networks* are more accurate for video classification than 2D and 3D convolutional networks [48] (including the inflated variant [7]). In addition, non-local neural networks are more computationally economical than their 3D convolutional counterparts. Comprehensive ablation studies are presented on the Kinetics [27] and Charades [44] datasets. *Using RGB only and without any bells and whistles* (*e.g.*, optical flow, multi-scale testing), our method achieves results on par with or better than the latest competitions winners on both datasets.

*Work done during an internship at Facebook AI Research.

To demonstrate the generality of non-local operations, we further present object detection/segmentation and pose estimation experiments on the COCO dataset [33]. On top of the strong Mask R-CNN baseline [19], our non-local blocks can increase accuracy on all three tasks at a small extra computational cost. Together with the evidence on videos, these image experiments show that non-local operations are generally useful and can become a basic building block in designing deep neural networks.

2. Related Work

Non-local image processing. Non-local means [4] is a classical filtering algorithm that computes a weighted mean of all pixels in an image. It allows distant pixels to contribute to the filtered response at a location based on patch appearance similarity. This non-local filtering idea was later developed into BM3D (block-matching 3D) [10], which performs filtering on a group of similar, but non-local, patches. BM3D is a solid image denoising baseline even compared with deep neural networks [5]. Block matching was used with neural networks for image denoising [6, 31]. Non-local matching is also the essence of successful texture synthesis [12], super-resolution [16], and inpainting [1] algorithms.

Graphical models. Long-range dependencies can be modeled by graphical models such as conditional random fields (CRF) [29, 28]. In the context of deep neural networks, a CRF can be exploited to post-process semantic segmentation predictions of a network [9]. The iterative mean-field inference of CRF can be turned into a recurrent network and trained [56, 42, 8, 18, 34]. In contrast, our method is a simpler feedforward block for computing non-local filtering. Unlike these methods that were developed for segmentation, our general-purpose component is applied for classification and detection. These methods and ours are also related to a more abstract model called graph neural networks [41].

Feedforward modeling for sequences. Recently there emerged a trend of using feedforward (*i.e.*, non-recurrent) networks for modeling sequences in speech and language [36, 54, 15]. In these methods, long-term dependencies are captured by the large receptive fields contributed by very deep 1-D convolutions. These feedforward models are amenable to parallelized implementations and can be more efficient than widely used recurrent models.

Self-attention. Our work is related to the recent *self-attention* [49] method for machine translation. A self-attention module computes the response at a position in a sequence (*e.g.*, a sentence) by attending to all positions and taking their weighted average in an embedding space. As we will discuss in the next, self-attention can be viewed as a form of the non-local mean [4], and in this sense our work bridges self-attention for machine translation to the more general class of non-local filtering operations that are applicable to image and video problems in computer vision.

Interaction networks. *Interaction Networks* (IN) [2, 52] were proposed recently for modeling physical systems. They operate on graphs of objects involved in pairwise interactions. Hoshen [24] presented the more efficient Vertex Attention IN (VAIN) in the context of multi-agent predictive modeling. Another variant, named Relation Networks [40], computes a function on the feature embeddings at all pairs of positions in its input. Our method also processes all pairs, as we will explain ($f(\mathbf{x}_i, \mathbf{x}_j)$ in Eq.(1)). While our non-local networks are connected to these approaches, our experiments indicate that the *non-locality* of the model, which is orthogonal to the ideas of attention/interaction/relation (*e.g.*, a network can attend to a local region), is the key to their empirical success. Non-local modeling, a long-time crucial element of image processing (*e.g.*, [12, 4]), has been largely overlooked in recent neural networks for computer vision.

Video classification architectures. A natural solution to video classification is to combine the success of CNNs for images and RNNs for sequences [55, 11]. In contrast, feedforward models are achieved by 3D convolutions (C3D) [26, 48] in spacetime, and the 3D filters can be formed by "inflating" [13, 7] pre-trained 2D filters. In addition to end-to-end modeling on raw video inputs, it has been found that optical flow [45] and trajectories [50, 51] can be helpful. Both flow and trajectories are off-the-shelf modules that may find long-range, non-local dependency. A systematic comparison of video architectures can be found in [7].

3. Non-local Neural Networks

We first give a general definition of non-local operations and then we provide several specific instantiations of it.

3.1. Formulation

Following the non-local mean operation [4], we define a generic non-local operation in deep neural networks as:

$$\mathbf{y}_i = \frac{1}{\mathcal{C}(\mathbf{x})} \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j) g(\mathbf{x}_j). \tag{1}$$

Here i is the index of an output position (in space, time, or spacetime) whose response is to be computed and j is the index that enumerates all possible positions. $\mathbf{x}$ is the input signal (image, sequence, video; often their features) and $\mathbf{y}$ is the output signal of the same size as $\mathbf{x}$. A pairwise function f computes a scalar (representing relationship such as affinity) between i and all j. The unary function g computes a representation of the input signal at the position j. The response is normalized by a factor $\mathcal{C}(\mathbf{x})$.

The non-local behavior in Eq.(1) is due to the fact that all positions ($\forall j$) are considered in the operation. As a comparison, a convolutional operation sums up the weighted input in a *local* neighborhood (*e.g.*, $i-1 \le j \le i+1$ in a 1D case with kernel size 3), and a recurrent operation at time

i is often based only on the current and the latest time steps (*e.g.*, $j = i$ or $i - 1$).

The non-local operation is also different from a fully-connected (*fc*) layer. Eq.(1) computes responses based on relationships between different locations, whereas *fc* uses learned weights. In other words, the relationship between $\mathbf{x}_j$ and $\mathbf{x}_i$ is not a function of the input data in *fc*, unlike in non-local layers. Furthermore, our formulation in Eq.(1) supports inputs of *variable* sizes, and maintains the corresponding size in the output. On the contrary, an *fc* layer requires a fixed-size input/output and loses positional correspondence (*e.g.*, that from $\mathbf{x}_i$ to $\mathbf{y}_i$ at the position i).

A non-local operation is a flexible building block and can be easily used together with convolutional/recurrent layers. It can be added into the earlier part of deep neural networks, unlike *fc* layers that are often used in the end. This allows us to build a richer hierarchy that combines both non-local and local information.

3.2. Instantiations

Next we describe several versions of f and g. Interestingly, we will show by experiments (Table 2a) that our non-local models are not sensitive to these choices, indicating that the generic non-local behavior is the main reason for the observed improvements.

For simplicity, we only consider g in the form of a linear embedding: $g(\mathbf{x}_j) = W_g\mathbf{x}_j$, where W_g is a weight matrix to be learned. This is implemented as, *e.g.*, 1×1 convolution in space or $1\times1\times1$ convolution in spacetime.

Next we discuss choices for the pairwise function f.

Gaussian. Following the non-local mean [4] and bilateral filters [47], a natural choice of f is the Gaussian function. In this paper we consider:

$$f(\mathbf{x}_i, \mathbf{x}_j) = e^{\mathbf{x}_i^T \mathbf{x}_j}. \tag{2}$$

Here $\mathbf{x}_i^T\mathbf{x}_j$ is dot-product similarity. Euclidean distance as used in [4, 47] is also applicable, but dot product is more implementation-friendly in modern deep learning platforms. The normalization factor is set as $\mathcal{C}(\mathbf{x}) = \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j)$.

Embedded Gaussian. A simple extension of the Gaussian function is to compute similarity in an embedding space. In this paper we consider:

$$f(\mathbf{x}_i, \mathbf{x}_j) = e^{\theta(\mathbf{x}_i)^T \phi(\mathbf{x}_j)}. \tag{3}$$

Here $\theta(\mathbf{x}_i) = W_\theta\mathbf{x}_i$ and $\phi(\mathbf{x}_j) = W_\phi\mathbf{x}_j$ are two embeddings. As above, we set $\mathcal{C}(\mathbf{x}) = \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j)$.

We note that *the self-attention module [49] recently presented for machine translation is a special case of non-local operations in the embedded Gaussian version.* This can be seen from the fact that for a given i, $\frac{1}{\mathcal{C}(\mathbf{x})} f(\mathbf{x}_i, \mathbf{x}_j)$ becomes the *softmax* computation along the dimension j. So we have $\mathbf{y} = \mathit{softmax}(\mathbf{x}^T W_\theta^T W_\phi \mathbf{x}) g(\mathbf{x})$, which is the self-attention form in [49]. As such, our work provides insight by relating this recent self-attention model to the classic computer vision method of non-local means [4], and extends the sequential self-attention network in [49] to a generic space/spacetime non-local network for image/video recognition in computer vision.

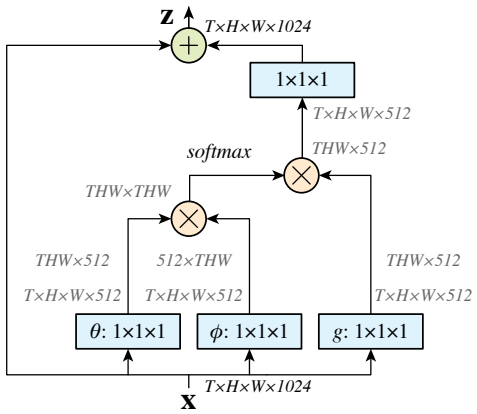

Figure 2. A spacetime **non-local block**. The feature maps are shown as the shape of their tensors, *e.g.*, $T\times H\times W\times 1024$ for 1024 channels (proper reshaping is performed when noted). "$\otimes$" denotes matrix multiplication, and "$\oplus$" denotes element-wise sum. The softmax operation is performed on each row. The blue boxes denote $1\times1\times1$ convolutions. Here we show the embedded Gaussian version, with a bottleneck of 512 channels. The vanilla Gaussian version can be done by removing θ and ϕ, and the dot-product version can be done by replacing softmax with scaling by $1/N$.

Despite the relation to [49], we show that the attentional behavior (due to softmax) is *not* essential in the applications we study. To show this, we describe two alternative versions of non-local operations next.

Dot product. f can be defined as a dot-product similarity:

$$f(\mathbf{x}_i, \mathbf{x}_j) = \theta(\mathbf{x}_i)^T \phi(\mathbf{x}_j). \tag{4}$$

Here we adopt the embedded version. In this case, we set the normalization factor as $\mathcal{C}(\mathbf{x}) = N$, where N is the number of positions in $\mathbf{x}$, rather than the sum of f, because it simplifies gradient computation. A normalization like this is necessary because the input can have variable size.

The main difference between the dot product and embedded Gaussian versions is the presence of softmax, which plays the role of an activation function.

Concatenation. Concatenation is used by the pairwise function in Relation Networks [40] for visual reasoning. We also evaluate a concatenation form of f:

$$f(\mathbf{x}_i, \mathbf{x}_j) = \mathrm{ReLU}(\mathbf{w}_f^T [\theta(\mathbf{x}_i),\ \phi(\mathbf{x}_j)]). \tag{5}$$

Here $[\cdot, \cdot]$ denotes concatenation and $\mathbf{w}_f$ is a weight vector that projects the concatenated vector to a scalar. As above, we set $\mathcal{C}(\mathbf{x}) = N$. In this case, we adopt ReLU [35] in f.

The above several variants demonstrate the flexibility of our generic non-local operation. We believe alternative versions are possible and may improve results.

3.3. Non-local Block

We wrap the non-local operation in Eq.(1) into a non-local block that can be incorporated into many existing architectures. We define a non-local block as:

$$\mathbf{z}_i = W_z \mathbf{y}_i + \mathbf{x}_i, \tag{6}$$

where $\mathbf{y}_i$ is given in Eq.(1) and "$+\mathbf{x}_i$" denotes a residual connection [21]. The residual connection allows us to insert a new non-local block into any pre-trained model, without breaking its initial behavior (*e.g.*, if W_z is initialized as zero). An example non-local block is illustrated in Figure 2. The pairwise computation in Eq.(2), (3), or (4) can be simply done by matrix multiplication as shown in Figure 2; the concatenation version in (5) is straightforward.

The pairwise computation of a non-local block is lightweight when it is used in high-level, sub-sampled feature maps. For example, typical values in Figure 2 are $T = 4$, $H = W = 14$ or 7. The pairwise computation as done by matrix multiplication is comparable to a typical convolutional layer in standard networks. We further adopt the following implementations that make it more efficient.

Implementation of Non-local Blocks. We set the number of channels represented by W_g, W_θ, and W_ϕ to be half of the number of channels in $\mathbf{x}$. This follows the bottleneck design of [21] and reduces the computation of a block by about a half. The weight matrix W_z in Eq.(6) computes a position-wise embedding on $\mathbf{y}_i$, matching the number of channels to that of $\mathbf{x}$. See Figure 2.

A subsampling trick can be used to further reduce computation. We modify Eq.(1) as: $\mathbf{y}_i = \frac{1}{\mathcal{C}(\hat{\mathbf{x}})} \sum_{\forall j} f(\mathbf{x}_i, \hat{\mathbf{x}}_j) g(\hat{\mathbf{x}}_j)$, where $\hat{\mathbf{x}}$ is a subsampled version of $\mathbf{x}$ (*e.g.*, by pooling). We perform this in the spatial domain, which can reduce the amount of pairwise computation by 1/4. This trick does not alter the non-local behavior, but only makes the computation sparser. This can be done by adding a max pooling layer after ϕ and g in Figure 2.

We use these efficient modifications for all non-local blocks studied in this paper.

4. Video Classification Models

To understand the behavior of non-local networks, we conduct comprehensive ablation experiments on video classification tasks. First we describe our baseline network architectures for this task, and then extend them into 3D ConvNets [48, 7] and our proposed non-local nets.

2D ConvNet baseline (C2D). To isolate the temporal effects of our non-local nets *vs*. 3D ConvNets, we construct a simple 2D baseline architecture in which the temporal dimension is trivially addressed (*i.e.*, only by pooling).

layer		output size
conv_1	7×7, 64, stride 2, 2, 2	16×112×112
pool_1	3×3×3 max, stride 2, 2, 2	8×56×56
res_2	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times3$	8×56×56
pool_2	3×1×1 max, stride 2, 1, 1	4×56×56
res_3	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times4$	4×28×28
res_4	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times6$	4×14×14
res_5	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times3$	4×7×7
global average pool, fc		1×1×1

Table 1. Our *baseline* ResNet-50 C2D model for video. The dimensions of 3D output maps and filter kernels are in T×H×W (2D kernels in H×W), with the number of channels following. The input is 32×224×224. Residual blocks are shown in brackets.

Table 1 shows our C2D baseline under a ResNet-50 backbone. The input video clip has 32 frames each with 224×224 pixels. All convolutions in Table 1 are in essence 2D kernels that process the input frame-by-frame (implemented as $1\times k\times k$ kernels). This model can be directly initialized from the ResNet weights pre-trained on ImageNet. A ResNet-101 counterpart is built in the same way.

The only operation involving the temporal domain are the pooling layers. In other words, this baseline simply aggregates temporal information.

Inflated 3D ConvNet (I3D). As done in [13, 7], one can turn the C2D model in Table 1 into a 3D convolutional counterpart by "inflating" the kernels. For example, a 2D $k\times k$ kernel can be inflated as a 3D $t\times k\times k$ kernel that spans t frames. This kernel can be initialized from 2D models (pre-trained on ImageNet): each of the t planes in the $t\times k\times k$ kernel is initialized by the pre-trained $k\times k$ weights, rescaled by $1/t$. If a video consists of a single static frame repeated in time, this initialization produces the same results as the 2D pre-trained model run on a static frame.

We study two cases of inflations: we either inflate the 3×3 kernel in a residual block to 3×3×3 (similar to [7]), or the first 1×1 kernel in a residual block to 3×1×1 (similar to [13]). We denote these as $\text{I3D}_{3\times3\times3}$ and $\text{I3D}_{3\times1\times1}$. As 3D convolutions are computationally intensive, we only inflate one kernel for every 2 residual blocks; inflating more layers shows diminishing return. We inflate conv_1 to 5×7×7.

The authors of [7] have shown that I3D models are more accurate than their CNN+LSTM counterparts.

Non-local network. We insert non-local blocks into C2D or I3D to turn them into non-local nets. We investigate adding 1, 5, or 10 non-local blocks; the implementation details are described in the next section in context.

Figure 3. Examples of the behavior of a non-local block in res_3 computed by a 5-block non-local model trained on Kinetics. These examples are from held-out validation videos. The starting point of arrows represents one $\mathbf{x}_i$, and the ending points represent $\mathbf{x}_j$. The 20 highest weighted arrows for each $\mathbf{x}_i$ are visualized. The 4 frames are from a 32-frame input, shown with a stride of 8 frames. These visualizations show how the model finds related clues to support its prediction.

4.1. Implementation Details

Training. Our models are pre-trained on ImageNet [39]. Unless specified, we fine-tune our models using 32-frame input clips. These clips are formed by randomly cropping out 64 consecutive frames from the original full-length video and then dropping every other frame. The spatial size is 224×224 pixels, randomly cropped from a scaled video whose shorter side is randomly sampled in $[256, 320]$ pixels, following [46]. We train on an 8-GPU machine and each GPU has 8 clips in a mini-batch (so in total with a mini-batch size of 64 clips). We train our models for 400k iterations in total, starting with a learning rate of 0.01 and reducing it by a factor of 10 at every 150k iterations (see also Figure 4). We use a momentum of 0.9 and a weight decay of 0.0001. We adopt dropout [22] after the global pooling layer, with a dropout ratio of 0.5. We fine-tune our models with BatchNorm (BN) [25] enabled when it is applied. This is in contrast to common practice [21] of fine-tuning ResNets, where BN was frozen. We have found that enabling BN in our application reduces overfitting.

We adopt the method in [20] to initialize the weight layers introduced in the non-local blocks. We add a BN layer right after the last 1×1×1 layer that represents W_z; we do not add BN to other layers in a non-local block. The scale parameter of this BN layer is initialized as zero, following [17]. This ensures that the initial state of the entire non-local block is an identity mapping, so it can be inserted into any pre-trained networks while maintaining its initial behavior.

Inference. Following [46] we perform spatially fully-convolutional inference on videos whose shorter side is rescaled to 256. For the temporal domain, in our practice we sample 10 clips evenly from a full-length video and compute the softmax scores on them individually. The final prediction is the averaged softmax scores of all clips.

5. Experiments on Video Classification

We perform comprehensive studies on the challenging Kinetics dataset [27]. We also report results on the Charades dataset [44] to show the generality of our models.

5.1. Experiments on Kinetics

Kinetics [27] contains ~246k training videos and 20k validation videos. It is a classification task involving 400 human action categories. We train all models on the training set and test on the validation set.

model, R50	top-1	top-5
C2D baseline	71.8	89.7
Gaussian	72.5	90.2
Gaussian, embed	72.7	**90.5**
dot-product	**72.9**	90.3
concatenation	72.8	**90.5**

(a) **Instantiations**: 1 non-local block of different types is added into the C2D baseline. All entries are with ResNet-50.

model, R50	top-1	top-5
baseline	71.8	89.7
res_2	72.7	90.3
res_3	**72.9**	90.4
res_4	72.7	**90.5**
res_5	72.3	90.1

(b) **Stages**: 1 non-local block is added into different stages. All entries are with ResNet-50.

model		top-1	top-5
R50	baseline	71.8	89.7
	1-block	72.7	90.5
	5-block	73.8	91.0
	10-block	**74.3**	**91.2**
R101	baseline	73.1	91.0
	1-block	74.3	91.3
	5-block	**75.1**	**91.7**
	10-block	**75.1**	91.6

(c) **Deeper non-local models**: we compare 1, 5, and 10 non-local blocks added to the C2D baseline. We show ResNet-50 (top) and ResNet-101 (bottom) results.

model		top-1	top-5
R50	baseline	71.8	89.7
	space-only	72.9	90.8
	time-only	73.1	90.5
	spacetime	**73.8**	**91.0**
R101	baseline	73.1	91.0
	space-only	74.4	91.3
	time-only	74.4	90.5
	spacetime	**75.1**	**91.7**

(d) **Space *vs*. time *vs*. spacetime**: we compare non-local operations applied along space, time, and spacetime dimensions respectively. 5 non-local blocks are used.

model, R101	params	FLOPs	top-1	top-5
C2D baseline	1×	1×	73.1	91.0
$\text{I3D}_{3\times3\times3}$	1.5×	1.8×	74.1	91.2
$\text{I3D}_{3\times1\times1}$	**1.2**×	1.5×	74.4	91.1
NL C2D, 5-block	**1.2**×	**1.2**×	**75.1**	**91.7**

(e) **Non-local *vs*. 3D Conv**: A 5-block non-local C2D *vs*. inflated 3D ConvNet (I3D) [7]. All entries are with ResNet-101. The numbers of parameters and FLOPs are relative to the C2D baseline (43.2M and 34.2B).

model		top-1	top-5
R50	C2D baseline	71.8	89.7
	I3D	73.3	90.7
	NL I3D	**74.9**	**91.6**
R101	C2D baseline	73.1	91.0
	I3D	74.4	91.1
	NL I3D	**76.0**	**92.1**

(f) **Non-local 3D ConvNet**: 5 non-local blocks are added on top of our best I3D models. These results show that non-local operations are complementary to 3D convolutions.

model		top-1	top-5
R50	C2D baseline	73.8	91.2
	I3D	74.9	91.7
	NL I3D	**76.5**	**92.6**
R101	C2D baseline	75.3	91.8
	I3D	76.4	92.7
	NL I3D	**77.7**	**93.3**

(g) **Longer clips**: we fine-tune and test the models in Table 2f on the 128-frame clips. The gains of our non-local operations are consistent.

Table 2. **Ablations** on Kinetics action classification. We show top-1 and top-5 classification accuracy (%).

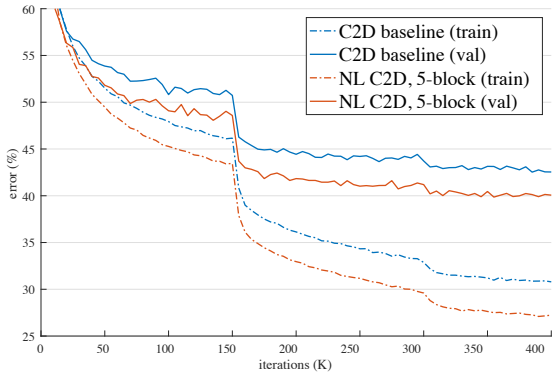

Figure 4. Curves of the training procedure on Kinetics for the ResNet-50 C2D baseline (blue) *vs*. non-local C2D with 5 blocks (red). We show the top-1 training error (dash) and validation error (solid). The validation error is computed in the same way as the training error (so it is 1-clip testing with the same random jittering at training time); the final results are in Table 2c (R50, 5-block).

Figure 4 shows the curves of the training procedure of a ResNet-50 C2D baseline *vs*. a non-local C2D with 5 blocks (more details in the following). Our non-local C2D model is consistently better than the C2D baseline *throughout the training procedure*, in both training and validation error.

Figure 1 and Figure 3 visualize several examples of the behavior of a non-local block computed by our models. Our network can learn to find meaningful relational clues regardless of the distance in space and time.

Table 2 shows the ablation results, analyzed as follows:

Instantiations. Table 2a compares different types of a single non-local block added to the C2D baseline (right before the last residual block of res_4). Even adding one non-local block can lead to ~1% improvement over the baseline.

Interestingly, the embedded Gaussian, dot-product, and concatenation versions perform similarly, up to some random variations (72.7 to 72.9). As discussed in Sec. 3.2, the non-local operations with Gaussian kernels become similar to the self-attention module [49]. However, our experiments show that the attentional (softmax) behavior of this module is *not* the key to the improvement in our applications; instead, it is more likely that the non-local behavior is important, and it is insensitive to the instantiations.

In the rest of this paper, we use the embedded Gaussian version by default. This version is easier to visualize as its softmax scores are in the range of $[0, 1]$.

Which stage to add non-local blocks? Table 2b compares a single non-local block added to different stages of ResNet. The block is added to right before the last residual block of a stage. The improvement of a non-local block on res_2, res_3, or res_4 is similar, and on res_5 is slightly smaller. One possible explanation is that res_5 has a small spatial size (7×7) and it is insufficient to provide precise spatial information. More evidence of a non-local block exploiting spatial information will be investigated in Table 2d.

model	backbone	modality	top-1 val	top-5 val	top-1 test	top-5 test	avg test†
I3D in [7]	Inception	RGB	72.1	90.3	71.1	89.3	80.2
2-Stream I3D in [7]	Inception	RGB + flow	75.7	92.0	74.2	91.3	82.8
RGB baseline in [3]	Inception-ResNet-v2	RGB	73.0	90.9	-	-	-
3-stream late fusion [3]	Inception-ResNet-v2	RGB + flow + audio	74.9	91.6	-	-	-
3-stream LSTM [3]	Inception-ResNet-v2	RGB + flow + audio	77.1	93.2	-	-	-
3-stream SATT [3]	Inception-ResNet-v2	RGB + flow + audio	77.7	93.2	-	-	-
NL I3D [ours]	ResNet-50	RGB	76.5	92.6	-	-	-
	ResNet-101	RGB	**77.7**	**93.3**	-	-	**83.8**

Table 3. Comparisons with state-of-the-art results in **Kinetics**, reported on the val and test sets. We include the Kinetics 2017 competition winner's results [3], but their best results exploited audio signals (marked in gray) so were not vision-only solutions. †: "avg" is the average of top-1 and top-5 accuracy; individual top-1 or top-5 numbers are not available from the test server at the time of submitting this manuscript.

Going deeper with non-local blocks. Table 2c shows the results of more non-local blocks. We add 1 block (to res_4), 5 blocks (3 to res_4 and 2 to res_3, to every other residual block), and 10 blocks (to every residual block in res_3 and res_4) in ResNet-50; in ResNet-101 we add them to the corresponding residual blocks. Table 2c shows that more non-local blocks in general lead to better results. We argue that multiple non-local blocks can perform long-range multi-hop communication. Messages can be delivered back and forth between distant positions in spacetime, which is hard to do via local models.

It is noteworthy that the improvement of non-local blocks is *not* just because they add depth to the baseline model. To see this, we note that in Table 2c the non-local 5-block ResNet-50 model has 73.8 accuracy, higher than the deeper ResNet-101 baseline's 73.1. However, the 5-block ResNet-50 has only ~70% parameters and ~80% FLOPs of the ResNet-101 baseline, and is also *shallower*. This comparison shows that the improvement due to non-local blocks is complementary to going deeper in standard ways.

We have also tried to add standard residual blocks, instead of non-local blocks, to the baseline models. The accuracy is not increased. This again shows that the improvement of non-local blocks is not just because they add depth.

Non-local in spacetime. Our method can naturally handle spacetime signals. This is a nice property: related objects in a video can present at distant space and long-term time interval, and their dependency can be captured by our model.

In Table 2d we study the effect of non-local blocks applied along space, time, or spacetime. For example, in the space-only version, the non-local dependency only happens within the same frame: *i.e.*, in Eq.(1) it only sums over the index j in the same frame of the index i. The time-only version can be set up similarly. Table 2d shows that both the space-only and time-only versions improve over the C2D baseline, but are inferior to the spacetime version.

Non-local net *vs.* 3D ConvNet. Table 2e compares our non-local C2D version with the inflated 3D ConvNets. Non-local operations and 3D convolutions can be seen as two ways of extending C2D to the temporal dimensions.

Table 2e also compares the number of parameters and FLOPs, relative to the baseline. Our non-local C2D model is more accurate than the I3D counterpart (*e.g.*, 75.1 *vs.* 74.4), while having a smaller number of FLOPs (1.2× *vs.* 1.5×). This comparison shows that our method can be more effective than 3D convolutions when used alone.

Non-local 3D ConvNet. Despite the above comparison, non-local operations and 3D convolutions can model different aspects of the problem: 3D convolutions can capture local dependency. Table 2f shows the results of inserting 5 non-local blocks into the $\text{I3D}_{3\times1\times1}$ models. These non-local I3D (NL I3D) models improve over their I3D counterparts (+1.6 point accuracy), showing that non-local operations and 3D convolutions are complementary.

Longer sequences. Finally we investigate the generality of our models on longer input videos. We use input clips consisting of 128 consecutive frames without subsampling. The sequences throughout all layers in the networks are thus 4× longer compared to the 32-frame counterparts. To fit this model into memory, we reduce the mini-batch size to 2 clips per GPU. As a result of using small mini-batches, we freeze all BN layers in this case. We initialize this model from the corresponding models trained with 32-frame inputs. We fine-tune on 128-frame inputs using the same number of iterations as the 32-frame case (though the mini-batch size is now smaller), starting with a learning rate of 0.0025. Other implementation details are the same as before.

Table 2g shows the results of 128-frame clips. Comparing with the 32-frame counterparts in Table 2f, all models have better results on longer inputs. We also find that our NL I3D can maintain its gain over the I3D counterparts, showing that our models work well on longer sequences.

Comparisons with state-of-the-art results. Table 3 shows the results from the I3D authors [7] and from the Kinetics 2017 competition winner [3]. We note that these are comparisons of systems which can differ in many aspects. Nevertheless, our method surpasses all the existing RGB or RGB + flow based methods by a good margin. *Without using optical flow and without any bells and whistles*, our method is on par with the heavily engineered results of the 2017 competition winner.

model	modality	*train/val*	*trainval/test*
2-Stream [43]	RGB + flow	18.6	-
2-Stream +LSTM [43]	RGB + flow	17.8	-
Asyn-TF [43]	RGB + flow	22.4	-
I3D [7]	RGB	32.9	34.4
I3D [ours]	RGB	35.5	37.2
NL I3D [ours]	RGB	**37.5**	**39.5**

Table 4. Classification mAP (%) in the **Charades** dataset [44], on the *train/val* split and the *trainval/test* split. Our results are based on ResNet-101. Our NL I3D uses 5 non-local blocks.

5.2. Experiments on Charades

Charades [44] is a video dataset with ~8k training, ~1.8k validation, and ~2k testing videos. It is a multi-label classification task with 157 action categories. We use a per-category sigmoid output to handle the multi-label property.

We initialize our models pre-trained on Kinetics (128-frame). The mini-batch size is set to 1 clip per GPU. We train our models for 200k iterations, starting from a learning rate of 0.00125 and reducing it by 10 every 75k iterations. We use a jittering strategy similar to that in Kinetics to determine the location of the 224×224 cropping window, but we rescale the video such that this cropping window outputs 288×288 pixels, on which we fine-tune our network. We test on a single scale of 320 pixels.

Table 4 shows the comparisons with the previous results on Charades. The result of [7] is the 2017 competition winner in Charades, which was also fine-tuned from models pre-trained in Kinetics. Our I3D baseline is higher than previous results. As a controlled comparison, our non-local net improves over our I3D baseline by 2.3% on the test set.

6. Extension: Experiments on COCO

We also investigate our models on static image recognition. We experiment on the Mask R-CNN baseline [19] for COCO [33] object detection/segmentation and human pose estimation (keypoint detection). The models are trained on COCO `train2017` (*i.e.*, `trainval35k` in 2014) and tested on `val2017` (*i.e.*, `minival` in 2014).

Object detection and instance segmentation. We modify the Mask R-CNN backbone by adding one non-local block (right before the last residual block of res_4). All models are fine-tuned from ImageNet pre-training. We evaluate on a standard baseline of ResNet-50/101 and a high baseline of ResNeXt-152 (X152) [53]. Unlike the original paper [19] that adopted stage-wise training regarding RPN, we use an improved implementation with end-to-end joint training similar to [37], which leads to higher baselines than [19].

Table 5 shows the box and mask AP on COCO. We see that a single non-local block improves all R50/101 and X152 baselines, on all metrics involving detection and segmentation. AP^{box} is increased by ~1 point in all cases (*e.g.*, +1.3 point in R101). Our non-local block is *complementary* to increasing the model capacity, even when the model is upgraded from R50/101 to X152. This comparison suggests that *non-local dependency has not been sufficiently captured by existing models despite increased depth/capacity*.

method		AP^{box}	$\text{AP}^{\text{box}}_{50}$	$\text{AP}^{\text{box}}_{75}$	AP^{mask}	$\text{AP}^{\text{mask}}_{50}$	$\text{AP}^{\text{mask}}_{75}$
R50	baseline	38.0	59.6	41.0	34.6	56.4	36.5
	+1 NL	**39.0**	**61.1**	**41.9**	**35.5**	**58.0**	**37.4**
R101	baseline	39.5	61.4	42.9	36.0	58.1	38.3
	+1 NL	**40.8**	**63.1**	**44.5**	**37.1**	**59.9**	**39.2**
X152	baseline	44.1	66.4	48.4	39.7	63.2	42.2
	+1 NL	**45.0**	**67.8**	**48.9**	**40.3**	**64.4**	**42.8**

Table 5. Adding 1 non-local block to Mask R-CNN for COCO **object detection** and **instance segmentation**. The backbone is ResNet-50/101 or ResNeXt-152 [53], both with FPN [32].

model	AP^{kp}	$\text{AP}^{\text{kp}}_{50}$	$\text{AP}^{\text{kp}}_{75}$
R101 baseline	65.1	86.8	70.4
NL, +4 in head	66.0	87.1	71.7
NL, +4 in head, +1 in backbone	**66.5**	**87.3**	**72.8**

Table 6. Adding non-local blocks to Mask R-CNN for COCO **keypoint detection**. The backbone is ResNet-101 with FPN [32].

In addition, the above gain is at a very small cost. The single non-local block only adds <5% computation to the baseline model. We also have tried to use more non-local blocks to the backbone, but found diminishing return.

Keypoint detection. Next we evaluate non-local blocks in Mask R-CNN for keypoint detection. In [19], Mask R-CNN used a stack of 8 convolutional layers for predicting the keypoints as 1-hot masks. These layers are local operations and may overlook the dependency among keypoints across long distance. Motivated by this, we insert 4 non-local blocks into the keypoint head (after every 2 convolutional layers).

Table 6 shows the results on COCO. On a strong baseline of R101, adding 4 non-local blocks to the keypoint head leads to a ~1 point increase of keypoint AP. If we add one extra non-local block to the backbone as done for object detection, we observe an in total 1.4 points increase of keypoint AP over the baseline. In particular, we see that the stricter criterion of AP_{75} is boosted by 2.4 points, suggesting a stronger localization performance.

7. Conclusion

We presented a new class of neural networks which capture long-range dependencies via non-local operations. Our non-local blocks can be combined with any existing architectures. We show the significance of non-local modeling for the tasks of video classification, object detection and segmentation, and pose estimation. On all tasks, a simple addition of non-local blocks provides solid improvement over baselines. We hope non-local layers will become an important component of future network architectures.

Acknowledgement: This work was partially supported by ONR MURI N000141612007, Sloan, Okawa Fellowship to AG and NVIDIA Fellowship to XW. We would also like to thank Haoqi Fan, Du Tran, Heng Wang, Georgia Gkioxari and Piotr Dollar for many helpful discussions.

References

[1] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. In *Proceedings of SIGGRAPH, ACM Transactions on Graphics*, 2009. 2

[2] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Neural Information Processing Systems (NIPS)*, 2016. 2

[3] Y. Bian, C. Gan, X. Liu, F. Li, X. Long, Y. Li, H. Qi, J. Zhou, S. Wen, and Y. Lin. Revisiting the effectiveness of off-the-shelf temporal modeling approaches for large-scale video classification. *arXiv:1708.03805*, 2017. 7

[4] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *Computer Vision and Pattern Recognition (CVPR)*, 2005. 1, 2, 3

[5] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with BM3D? In *Computer Vision and Pattern Recognition (CVPR)*, 2012. 2

[6] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising with multi-layer perceptrons, part 2: training trade-offs and analysis of their mechanisms. *arXiv:1211.1552*, 2012. 2

[7] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2, 4, 6, 7, 8

[8] S. Chandra, N. Usunier, and I. Kokkinos. Dense and low-rank Gaussian CRFs using deep embeddings. In *International Conference on Computer Vision (ICCV)*, 2017. 2

[9] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. *arXiv:1412.7062*, 2014. 2

[10] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *Transactions on Image Processing (TIP)*, 2007. 2

[11] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. 2

[12] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *International Conference on Computer Vision (ICCV)*, 1999. 2

[13] C. Feichtenhofer, A. Pinz, and R. Wildes. Spatiotemporal residual networks for video action recognition. In *Neural Information Processing Systems (NIPS)*, 2016. 2, 4

[14] K. Fukushima and S. Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*. Springer, 1982. 1

[15] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. In *International Conference on Machine Learning (ICML)*, 2017. 2

[16] D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. In *Computer Vision and Pattern Recognition (CVPR)*, 2009. 2

[17] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv:1706.02677*, 2017. 5

[18] A. Harley, K. Derpanis, and I. Kokkinos. Segmentation-aware convolutional networks using local attention masks. In *International Conference on Computer Vision (ICCV)*, 2017. 2

[19] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *International Conference on Computer Vision (ICCV)*, 2017. 2, 8

[20] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *International Conference on Computer Vision (ICCV)*, 2015. 5

[21] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 4, 5

[22] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*, 2012. 5

[23] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 1997. 1

[24] Y. Hoshen. Multi-agent predictive modeling with attentional commnets. In *Neural Information Processing Systems (NIPS)*, 2017. 2

[25] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015. 5

[26] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. In *International Conference on Machine Learning (ICML)*, 2010. 2

[27] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv:1705.06950*, 2017. 1, 5

[28] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Neural Information Processing Systems (NIPS)*, 2011. 2

[29] J. Lafferty, A. McCallum, and F. C. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning (ICML)*, 2001. 2

[30] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1989. 1

[31] S. Lefkimmiatis. Non-local color image denoising with convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[32] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 8

[33] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*. 2014. 2, 8

[34] S. Liu, S. De Mello, J. Gu, G. Zhong, M.-H. Yang, and J. Kautz. Learning affinity via spatial propagation networks. In *Neural Information Processing Systems (NIPS)*, 2017. 2

[35] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning (ICML)*, 2010. 3

[36] A. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv:1609.03499*, 2016. 2

[37] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2017. 8

[38] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 1986. 1

[39] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 2015. 5

[40] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. In *Neural Information Processing Systems (NIPS)*, 2017. 2, 3

[41] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 2009. 2

[42] A. G. Schwing and R. Urtasun. Fully connected deep structured networks. *arXiv preprint arXiv:1503.02351*, 2015. 2

[43] G. A. Sigurdsson, S. Divvala, A. Farhadi, and A. Gupta. Asynchronous temporal fields for action recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 8

[44] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision (ECCV)*, 2016. 1, 5, 8

[45] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Neural Information Processing Systems (NIPS)*, 2014. 2

[46] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015. 5

[47] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *International Conference on Computer Vision (ICCV)*, 1998. 3

[48] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *International Conference on Computer Vision (ICCV)*, 2015. 1, 2, 4

[49] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Neural Information Processing Systems (NIPS)*, 2017. 2, 3, 6

[50] H. Wang and C. Schmid. Action recognition with improved trajectories. In *International Conference on Computer Vision (ICCV)*, 2013. 2

[51] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. 2

[52] N. Watters, A. Tacchetti, T. Weber, R. Pascanu, P. Battaglia, and D. Zoran. Visual interaction networks. In *Neural Information Processing Systems (NIPS)*, 2017. 2

[53] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 8

[54] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig. The Microsoft 2016 Conversational Speech Recognition System. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2017. 2

[55] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. 2

[56] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. In *International Conference on Computer Vision (ICCV)*, 2015. 2