

# Non-local Neural Networks

*Proceedings of the IEEE conference on computer vision and pattern recognition. (CVPR) 2018.*

**Presenter:** Giorgio Roffo

**Date:** 03/12/2021

## Non-local Neural Networks

Xiaolong Wang<sup>1,2\*</sup>

Ross Girshick<sup>2</sup>

Abhinav Gupta<sup>1</sup>

Kaiming He<sup>2</sup>

<sup>1</sup>Carnegie Mellon University

<sup>2</sup>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*



Figure 1. A spacetime **non-local** operation in our network trained for video classification in Kinetics. A position  $x_i$ 's response is computed by the weighted average of the features of *all* positions  $x_j$  (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.

---

# Non-local Neural Networks

Xiaolong Wang<sup>1,2\*</sup>Ross Girshick<sup>2</sup>Abhinav Gupta<sup>1</sup>Kaiming He<sup>2</sup><sup>1</sup>Carnegie Mellon University<sup>2</sup>Facebook AI Research

## Motivation

- Both convolutional and recurrent operations are building blocks that process one local neighborhood at a time.
- Non-local operations are used for capturing long-range dependencies.
- This work relates to:
  - The Non-local Means method in computer vision [1] (i.e., non-local image processing)
  - Self-Attention [2] (the Transformer)

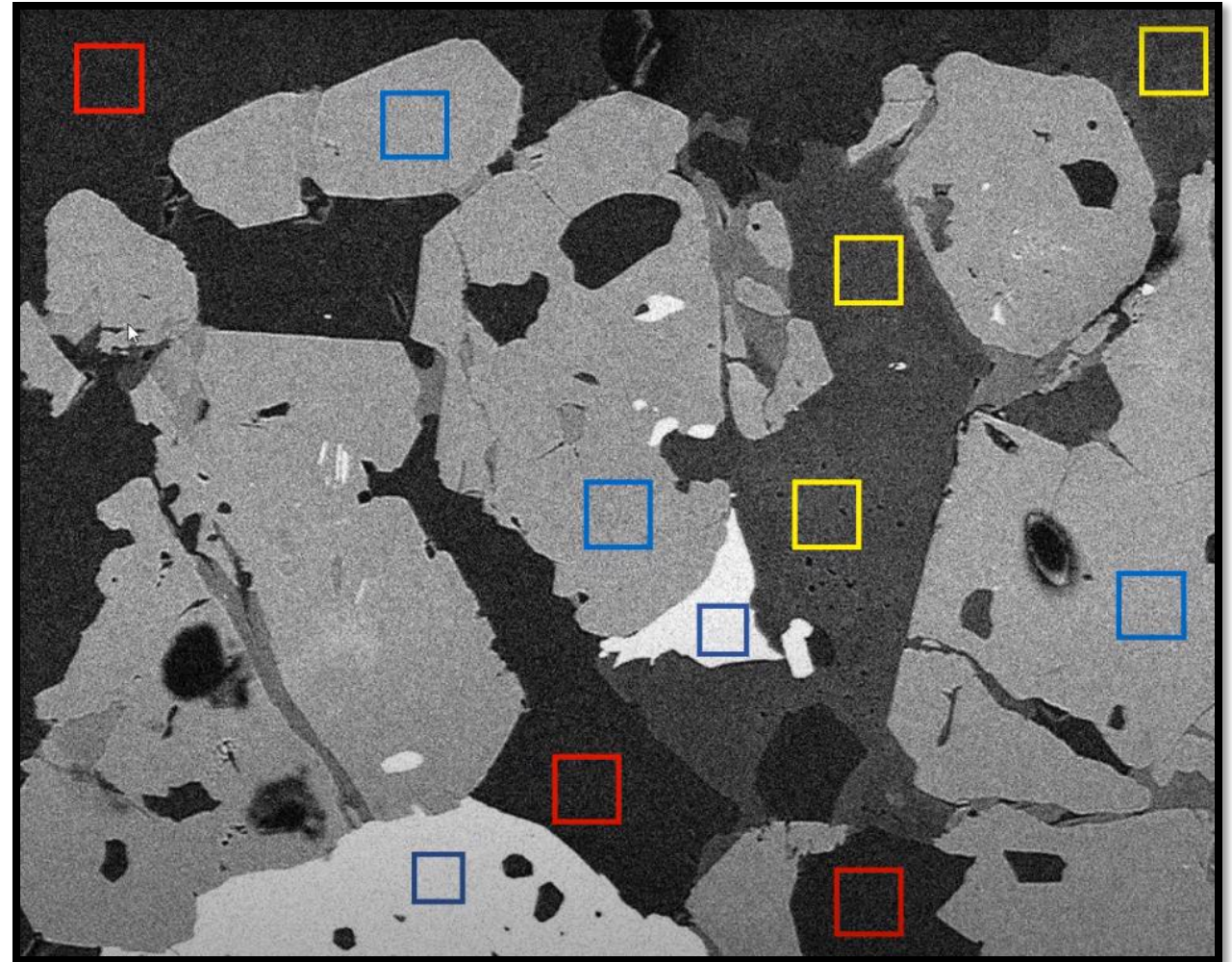
1. A. Buades, B. Coll, and J.-M. Morel. **A non-local algorithm for image denoising**. In *Computer Vision and Pattern Recognition (CVPR)*, 2005.
  2. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. **Attention is all you need**. In *Neural Information Processing Systems (NIPS)*, 2017.
-

# Non-local mean operation (image processing)

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

1. Estimated value  $\mathbf{y}$  is the weighted average of all pixels in the image.
2. The weights  $\mathbf{f}$  depend on the similarity between the pixels  $\mathbf{i}$  and  $\mathbf{j}$ .

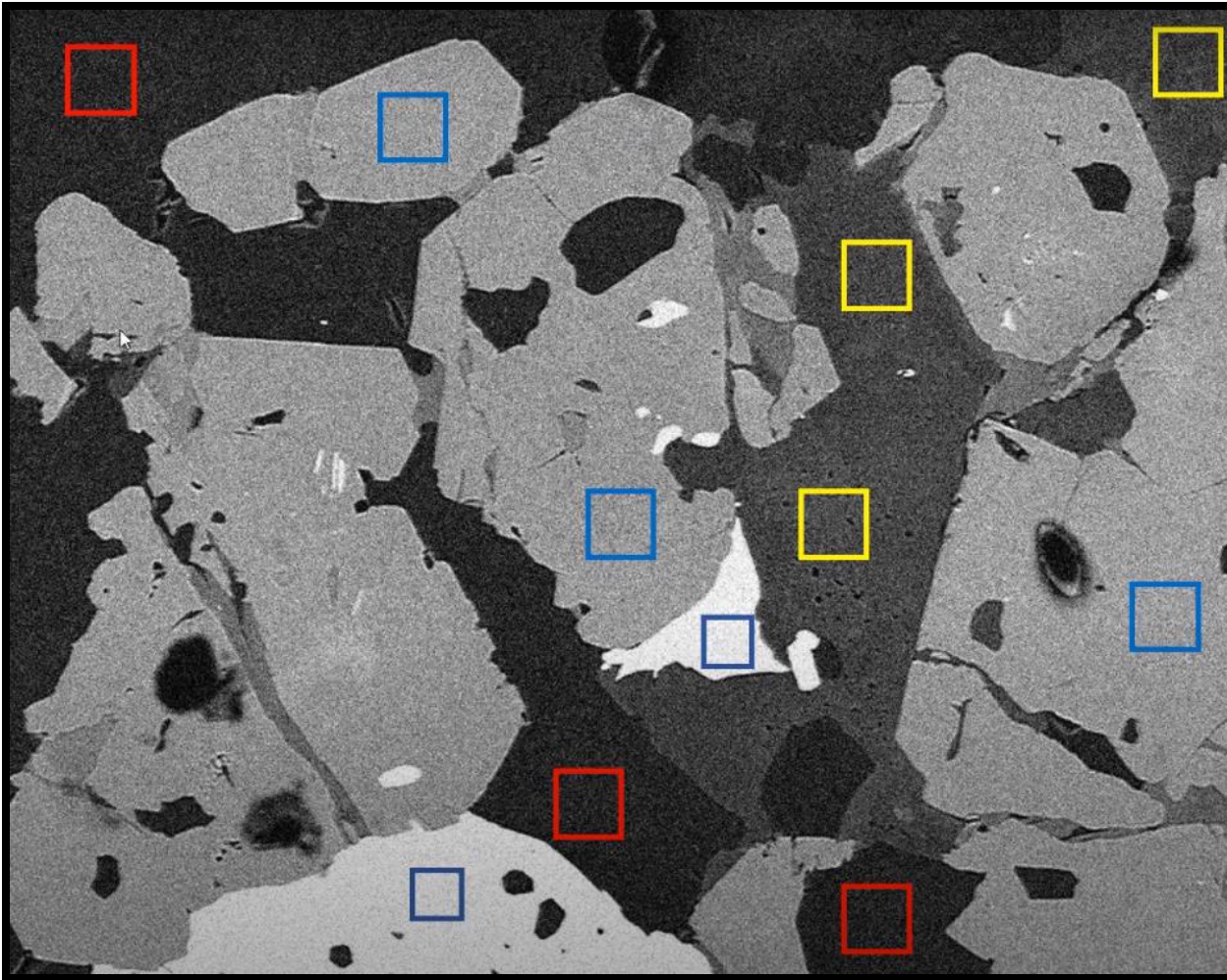
*It looks at a patch of image and then identifies other similar patches in the whole image and takes a weighted average of them.*



*Similar patches are marked with the same colored square boxes*



# Non-local mean operation - denoising



*Similar patches are marked with the same colored square boxes*



*Denoised*

# Non-local as self-attention

- In image processing, similar regions are updated to the same value.
  - If  $R1=[x1,x2,x3]$  and  $R2=[x4,x5,x6]$  are two different regions  $R1 \neq R2$ , and Each  $x_i$  in  $R_i$  have similar characteristics.
  - The process:
$$y_i = \frac{1}{C(\mathbf{x})} \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j) g(\mathbf{x}_j).$$
  - Assigns to  $R1=[s1,s1,s1]$  and  $R2=[s2,s2,s2]$
- In CNNs,  $\mathbf{g}(\mathbf{x})$  is a feature learned for a specific task (not a pixel).
- Self-attention allows **important** features to have the same values.

# Non-local as self-attention

- In image processing, similar regions are updated to the same value.
  - If  $R1=[x1,x2,x3]$  and  $R2=[x4,x5,x6]$  are two different regions  $R1 \neq R2$ , and Each  $x_i$  in  $R_i$  have similar characteristics.
  - The process:
$$y_i = \frac{1}{C(\mathbf{x})} \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j) g(\mathbf{x}_j).$$
    - Assigns to  $R1=[s1,s1,s1]$  and  $R2=[s2,s2,s2]$
- In CNNs,  $\mathbf{g}(\mathbf{x})$  is a feature learned for a specific task (not a pixel).
- Self-attention allows **important** features to have the same values.

*ball and actor play an important role in the classification of this activity.*



# Pairwise Functions



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.


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

Pairwise Function	Definition
Gaussian	$f(x_i, x_j) = e^{x_i^T x_j}$
Embedded Gaussian	$f(x_i, x_j) = e^{\theta(x_i)^T \phi(x_j)}$
Dot product	$f(x_i, x_j) = \theta(x_i)^T \phi(x_j)$
Concatenation	$f(x_i, x_j) = \text{ReLU}(w_f^T [\theta(x_i), \phi(x_j)])$

# Pairwise Functions

Pairwise Function	Definition
Gaussian	$f(x_i, x_j) = e^{x_i^T x_j}$
Embedded Gaussian	$f(x_i, x_j) = e^{\theta(x_i)^T \phi(x_j)}$
Dot product	$f(x_i, x_j) = \theta(x_i)^T \phi(x_j)$
Concatenation	$f(x_i, x_j) = \text{ReLU}(w_f^T [\theta(x_i), \phi(x_j)])$

## Self-Attention


$$A(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

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



# Non-local blocks

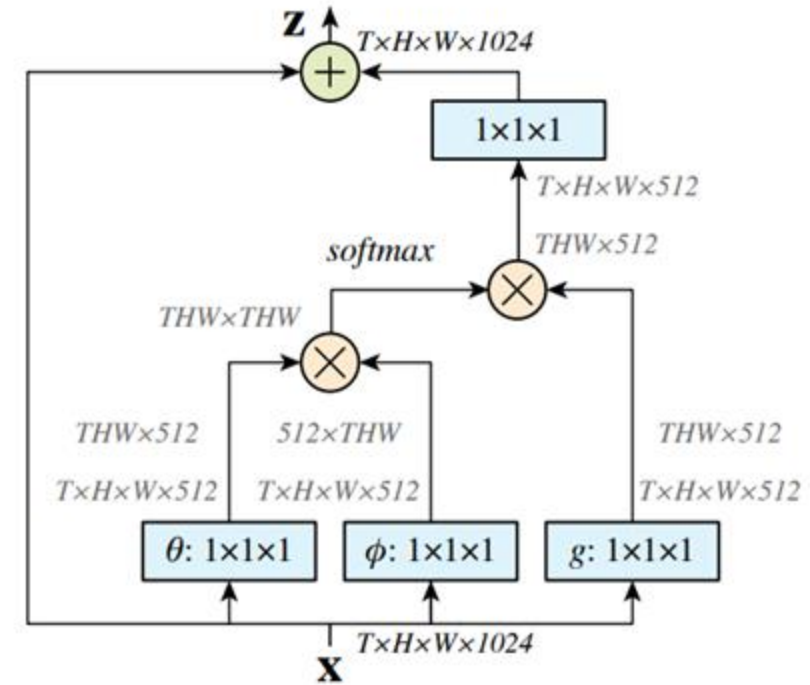
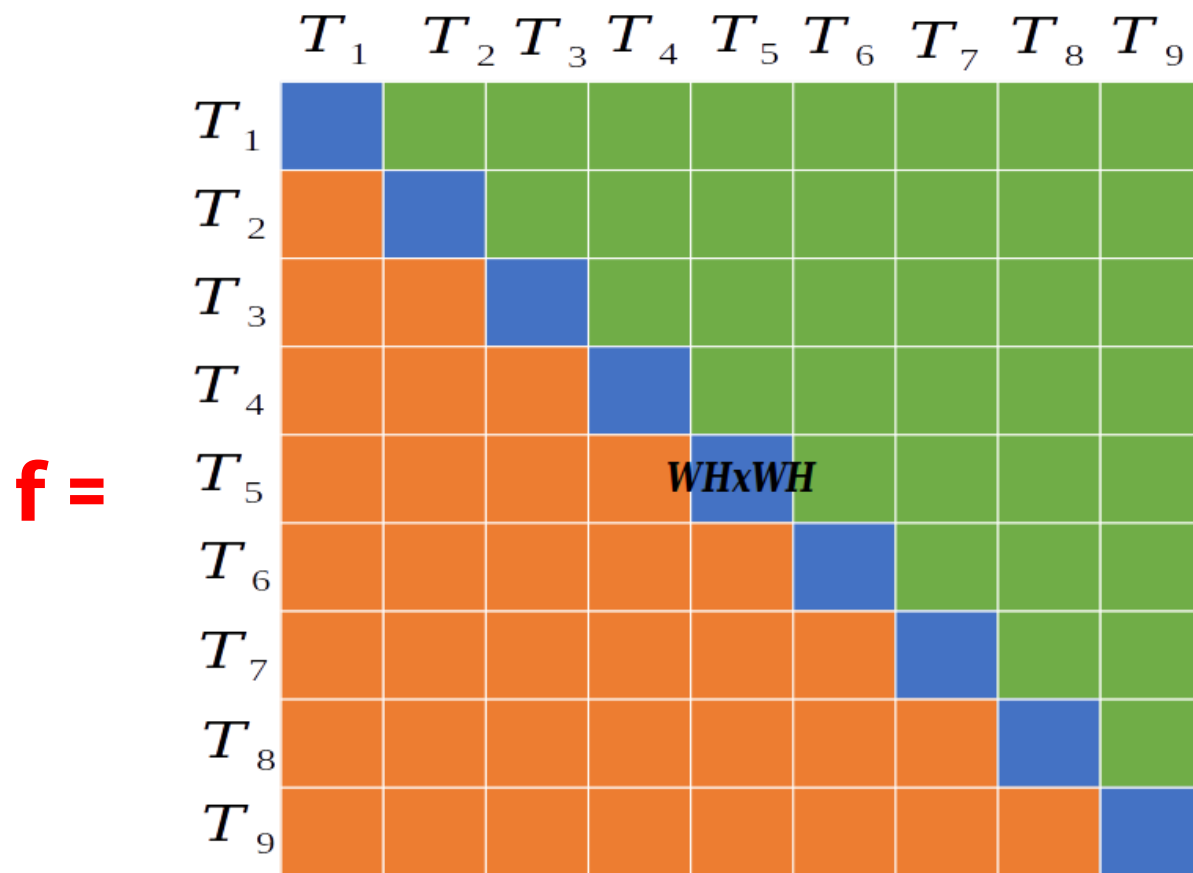


Figure 2. A spacetime **non-local block**. The feature maps are shown as the shape of their tensors, *e.g.*,  $T \times H \times W \times 1024$  for 1024 channels (proper reshaping is performed when noted). “ $\otimes$ ” denotes matrix multiplication, and “ $\oplus$ ” denotes element-wise sum. The softmax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

# Attention Matrix TWHxTWH



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

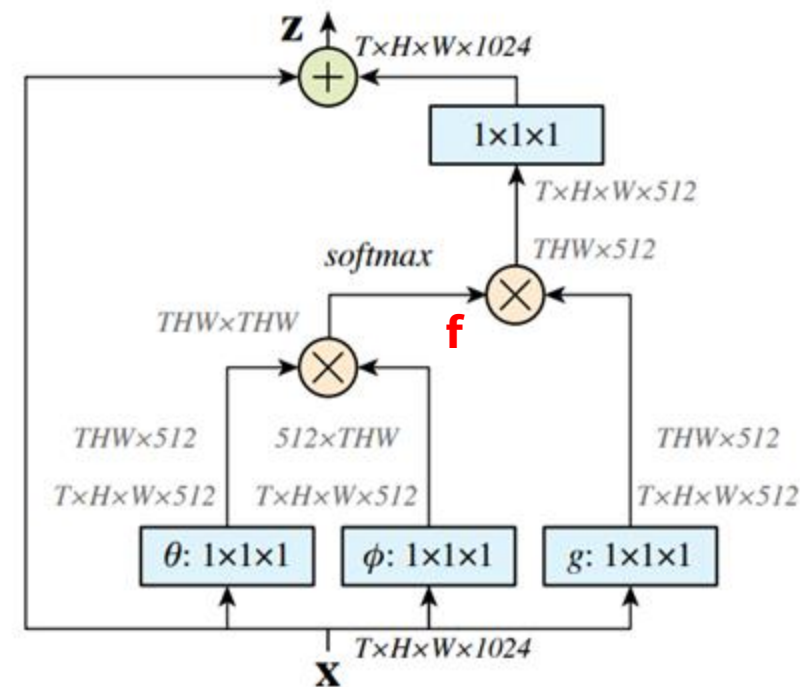


Figure 2. A spacetime **non-local block**. The feature maps are shown as the shape of their tensors, *e.g.*,  $T \times H \times W \times 1024$  for 1024 channels (proper reshaping is performed when noted). “ $\otimes$ ” denotes matrix multiplication, and “ $\oplus$ ” denotes element-wise sum. The softmax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

# Backbones

- The baseline is a ResNet-50 C2D where the temporal dimension is trivially addressed (i.e., only by pooling).
- This baseline simply aggregates temporal information.
- The input video clip has 32 frames each with  $224 \times 224$  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.

	layer	output size
conv <sub>1</sub>	$7 \times 7, 64, \text{stride } 2, 2, 2$	$16 \times 112 \times 112$
pool <sub>1</sub>	$3 \times 3 \times 3 \text{ max, stride } 2, 2, 2$	$8 \times 56 \times 56$
res <sub>2</sub>	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$8 \times 56 \times 56$
pool <sub>2</sub>	$3 \times 1 \times 1 \text{ max, stride } 2, 1, 1$	$4 \times 56 \times 56$
res <sub>3</sub>	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$4 \times 28 \times 28$
res <sub>4</sub>	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$4 \times 14 \times 14$
res <sub>5</sub>	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$4 \times 7 \times 7$
global average pool, fc		$1 \times 1 \times 1$

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

# Backbones – NL i3D

- Inflated 3D ConvNet (i3D) [1].
- As in [1], the authors turn the C2D model in Table 1 into a 3D convolutional counterpart by “inflating” the kernels.
  - 2D  $\mathbf{k} \times \mathbf{k}$  kernel can be inflated as a 3D  $\mathbf{t} \times \mathbf{k} \times \mathbf{k}$  kernel that spans  $\mathbf{t}$  frames.
  - This kernel can be initialized from 2D models (pretrained on ImageNet). Each of the  $\mathbf{t}$  planes in the  $\mathbf{t} \times \mathbf{k} \times \mathbf{k}$  kernel is initialized by the pre-trained  $\mathbf{k} \times \mathbf{k}$  weights, rescaled by  $1/\mathbf{t}$ .
  - As 3D convolutions are computationally intensive, only one kernel is inflated for every 2 residual blocks; inflating more layers shows diminishing return.
  - The authors of [1] have shown that i3D models are more accurate than their CNN+LSTM counterparts.

	layer	output size
conv <sub>1</sub>	7×7, 64, stride 2, 2, 2	16×112×112
pool <sub>1</sub>	3×3×3 max, stride 2, 2, 2	8×56×56
res <sub>2</sub>	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	8×56×56
pool <sub>2</sub>	3×1×1 max, stride 2, 1, 1	4×56×56
res <sub>3</sub>	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	4×28×28
res <sub>4</sub>	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	4×14×14
res <sub>5</sub>	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	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.



---

# Non-local networks

- Non-local blocks are inserted into C2D or I3D to turn them into non-local nets.
  - **Which stage to add non-local blocks?**
    - The block is added to right before the last residual block of a stage.
    - The improvement of a non-local block on res2, res3, or res4 is similar, and on res5 is slightly smaller (Res5 has a small spatial size ( $7 \times 7$ )).
  - **Going deeper with non-local blocks**
    - 1 block (to res4),
    - **5 blocks** (3 to res4 and 2 to res3, to every other residual block), and
    - 10 blocks (to every residual block in res3 and res4)
-

## Non-local in spacetime

related objects in a video can present at distant space and long-term time interval, and their dependency can be captured by our model

### Longer sequences.

The authors investigate the generality of the model on longer input videos (128 frames, 2 clips per GPU).



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  $x_i$ , and the ending points represent  $x_j$ . The 20 highest weighted arrows for each  $x_i$  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.

# Ablation study

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

model, R50	top-1	top-5
baseline	71.8	89.7
res <sub>2</sub>	72.7	90.3
res <sub>3</sub>	<b>72.9</b>	90.4
res <sub>4</sub>	72.7	<b>90.5</b>
res <sub>5</sub>	72.3	90.1

	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	<b>74.3</b>	<b>91.2</b>
R101	baseline	73.1	91.0
	1-block	74.3	91.3
	5-block	<b>75.1</b>	<b>91.7</b>
	10-block	<b>75.1</b>	91.6

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

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

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

(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.

(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
I3D <sub>3×3×3</sub>	1.5×	1.8×	74.1	91.2
I3D <sub>3×1×1</sub>	<b>1.2×</b>	1.5×	74.4	91.1
NL C2D, 5-block	<b>1.2×</b>	<b>1.2×</b>	<b>75.1</b>	<b>91.7</b>

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

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

(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).

(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.

(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.

# Comparisons with state-of-the-art results in Kinetics

model	backbone	modality	top-1 val	top-5 val	top-1 test	top-5 test	avg test <sup>†</sup>
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	<b>77.7</b>	<b>93.3</b>	-	-	<b>83.8</b>

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. <sup>†</sup>: “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.

**Kinetics.** A collection of 650,000 video clips that cover 400/600/700 human action classes, depending on the dataset version. The videos include human-object interactions such as playing instruments, as well as human-human interactions such as shaking hands and hugging.



# Experiments

- **Action rec.** Charades is a video dataset for a multi-label classification task with 157 action categories.
- **Object detection and instance segmentation.** 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.
- **Keypoint detection.** Mask R-CNN used a stack of 8 convolutional layers for predicting the keypoints as 1-hot masks. 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  $\sim 1$  point increase of keypoint AP

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	<b>37.5</b>	<b>39.5</b>

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.

method		AP <sup>box</sup>	AP <sup>box</sup> <sub>50</sub>	AP <sup>box</sup> <sub>75</sub>	AP <sup>mask</sup>	AP <sup>mask</sup> <sub>50</sub>	AP <sup>mask</sup> <sub>75</sub>
R50	baseline	38.0	59.6	41.0	34.6	56.4	36.5
	+1 NL	<b>39.0</b>	<b>61.1</b>	<b>41.9</b>	<b>35.5</b>	<b>58.0</b>	<b>37.4</b>
R101	baseline	39.5	61.4	42.9	36.0	58.1	38.3
	+1 NL	<b>40.8</b>	<b>63.1</b>	<b>44.5</b>	<b>37.1</b>	<b>59.9</b>	<b>39.2</b>
X152	baseline	44.1	66.4	48.4	39.7	63.2	42.2
	+1 NL	<b>45.0</b>	<b>67.8</b>	<b>48.9</b>	<b>40.3</b>	<b>64.4</b>	<b>42.8</b>

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 <sup>kp</sup>	AP <sup>kp</sup> <sub>50</sub>	AP <sup>kp</sup> <sub>75</sub>
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	<b>66.5</b>	<b>87.3</b>	<b>72.8</b>

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

---

# Conclusions

- Non-local operations allow the model to capture long-range dependencies
  - Space, Time, or Spacetime (set of frames are analyzed at the same time and the relationships between objects learned even if they are distant in time).
- ResNet-50: the non-local blocks perform better if used at res3 and res4 stages.
- Adding NL blocks improves C2d and i3D architectures.
- The best performances are obtained with the NL i3D model with 5 blocks.
  - **Pros: NL i3D** was the SOTA on activity recognition, object detection, instance segmentation and KeyPoint detection tasks (in 2018).
  - **Cons: the i3D models are computationally intensive (therefore NL i3D as well)**

***Is it possible to achieve the performance of 3D CNNs while maintaining the complexity of 2D CNNs?***

---

---

# Summary

- **NL Nets**

- Non-local operations allows the model to capture long-range dependencies.
- Adding NL blocks improves C2d and i3D architectures.
- The best performances are obtained with the NL i3D model with 5 blocks.

**Pros:** NL i3D SOTA on activity recognition, object detection, instance segmentation and KeyPoint detection tasks.

**Cons:** i3D models are computationally intensive (\*).

- **TSM**

- **TSM converts any 2D CNN model into a pseudo-3D model (\*).**
  - TSM achieves the performance of 3D CNN but maintain C2D's complexity.
  - TSM **shifts part of the channels along the temporal dimension (1/4)**
  - TSM supports both **offline** and **online** video recognition.
-

Thank you

