Additional Material of Poly-NL: Linear Complexity Non-local Layers With 3rd Order Polynomials

Francesca Babiloni[1], Ioannis Marras[1], Filippos Kokkinos[1,3], Jiankang Deng[2,5], Grigorios Chrysos[4], Stefanos Zafeiriou[2,5]

[1]Huawei, Noah's Ark Lab [2]Imperial College London [3]University College London
[4]École Polytechnique Fédérale de Lausanne [5]Huawei, Multimedia Technology Department

1. Polynomials and Non-local blocks

In this section, we describe the assumptions on the parameter tensor needed to frame Poly-NL and the Non-Local block as polynomial functions.

Given an input matrix $\mathbf{X} \in \mathbb{R}^{N \times C}$, the polynomial function of Eq.(4) of the main paper reads

$$\mathbf{Y} = (((\boldsymbol{\mathcal{W}}^{[3]} \bullet \mathbf{X}) \bullet \mathbf{X}) \bullet \mathbf{X}) \tag{1}$$

It enumerates all the possible 3^{rd} order interactions, as shown in the element-wise formula:

$$y_{(a,b)} = \sum_{c,e,g}^{N} \sum_{d,f,h}^{C} w_{3_{(a,b,c,d,e,f,g,h)}} x_{(c,d)} x_{(e,f)} x_{(g,h)} \tag{2}$$

where $w_{3_{(a,b,c,d,e,f,g,h)}}$ are the elements of $\boldsymbol{\mathcal{W}}^{[3]}$, a tensor of order 8 and dimension $\mathbb{R}^{N \times C \times N \times C \times N \times C \times N \times C}$.

Structuring $\boldsymbol{\mathcal{W}}^{[3]}_{\text{NL}}$. The Non-local block, as described in Eq.(2) of the main paper

$$\mathbf{Y}^{\text{NL}} = \mathbf{X}\mathbf{W}_f \mathbf{X}^{\top} \mathbf{X}\mathbf{W}_g \tag{3}$$

can be viewed as a special case of 3^{rd} order polynomials. Hence, it can alternatively be computed as follows

$$\mathbf{Y}^{\text{NL}} = (((\boldsymbol{\mathcal{W}}^{[3]}_{\text{NL}} \bullet \mathbf{X}) \bullet \mathbf{X}) \bullet \mathbf{X}) \tag{4}$$

where $\boldsymbol{\mathcal{W}}^{[3]}_{\text{NL}}$ is a tensor of order 8 and dimension $\mathbb{R}^{N \times C \times N \times C \times N \times C \times N \times C}$ in charge of extracting the subset of non-local dependencies. The elements of the tensor are listed as $w^{\text{NL}}_{3_{(a,b,c,d,e,f,g,h)}}$ where indexes a, c, e, g are ranging from 1 to N and b, d, f, h are ranging from 1 to C. Similarly, the scalars $w_{f(d,f)}$ and $w_{g(h,b)}$ enumerate the elements of the matrices $\mathbf{W}_f \in \mathbb{R}^{C \times C}$, $\mathbf{W}_g \in \mathbb{R}^{C \times C}$ respectively. The two formulations are equivalent for the following assumptions on the elements $w^{\text{NL}}_{3_{(a,b,c,d,e,f,g,h)}}$: i) elements are null for every c different from a ii) elements are null for every g different from e ii) remaining elements $w^{\text{NL}}_{3_{(a,b,c,d,e,f,g,h)}}$ are equal to $w_{f(d,f)} w_{g(h,b)}$. Using the aforementioned structure, the parameter tensor of the 3^{rd} polynomial function considers only relations associated with non-local dependencies.

Structuring $\boldsymbol{\mathcal{W}}^{[3]}_{\text{Poly-NL}}$. The Poly-NL, as described in Eq.(6) of the main paper can be computed as follows

$$\mathbf{Y}^{\text{Poly-NL}} = (\Phi(\mathbf{X}\mathbf{W}_1 \odot \mathbf{X}\mathbf{W}_2) \odot \mathbf{X})\mathbf{W}_3, \tag{5}$$

where $\mathbf{W}_1 \in \mathbb{R}^{C \times C}$, $\mathbf{W}_2 \in \mathbb{R}^{C \times C}$, and $\mathbf{W}_3 \in \mathbb{R}^{C \times C}$ are matrices of learnable parameters and Φ is an average pooling function. Analogously to what described in the previous paragraph, the above equation can also be computed as follows

$$\mathbf{Y}^{\text{Poly-NL}} = (((\boldsymbol{\mathcal{W}}^{[3]}_{\text{Poly-NL}} \bullet \mathbf{X}) \bullet \mathbf{X}) \bullet \mathbf{X}) \tag{6}$$

where $\boldsymbol{\mathcal{W}}^{[3]}_{\text{Poly-NL}}$ is a tensor of order 8 and dimension $\mathbb{R}^{N \times C \times N \times C \times N \times C \times N \times C}$ of elements $w^{\text{Poly-NL}}_{3_{(a,b,c,d,e,f,g,h)}}$, with indexes a, c, e, g ranging from 1 to N and b, d, f, h ranging from 1 to C. The structure of the tensor $\boldsymbol{\mathcal{W}}^{[3]}_{\text{Poly-NL}}$ considers only a subset of all possible 3^{rd} order dependencies. In particular, its elements $w^{\text{Poly-NL}}_{3_{(a,b,c,d,e,f,g,h)}}$ are null for every c different from a or g different from e and equal to $w'_{1(h,d)} w_{2(f,d)} w_{3(d,b)}$ in the remaining cases. Without loss of generality, we consider the values $w_1{}'_{(h,d)}$ enumerating the parameters of the matrix $\mathbf{W}'_1 = \mathbf{W}_1/N$, where the $1/N$ term of the average pooling function is merged with the parameter matrix. Similarly to the Non-Local block case, Poly-NL can be viewed as 3^{rd} order polynomial function, where the tensor of parameters focuses on extracting non-local interactions from the input tensor.

Method	AP_{box}	AP_{box50}	AP_{box75}	AP_{mask}	AP_{mask50}	AP_{mask75}
MaskR-CNN	40.1	61.3	43.7	36.2	57.9	38.4
w/ Poly-NL						
+ on Res3	40.8	62.8	44.3	36.8	59.3	39.0
+ on Res4	**41.5**	**63.0**	**45.2**	**37.2**	**59.7**	**39.7**
+ on Res5	40.9	62.7	44.6	36.7	59.3	38.9

(a) **Instance Segmentation on ResNet-101**

Method	*Easy*	*Medium*	*Hard*
ResNet-18	90.90	90.39	87.67
+ Non-local	91.23	90.97	88.16
+ TESA	<u>91.67</u>	<u>91.25</u>	<u>88.87</u>
+ Latent-GNN	91.46	91.02	88.61
+ Efficient-NL	91.52	91.09	88.65
+ Poly-NL	**91.98**	**91.41**	**89.05**

(b) **Face Detection on ResNet-18 (x0.25)**

Table 1: **Additional Results**. Tables report results on two different architectures for a) the Face Detection task on WIDER-FACE and b) Instance Segmentation on COCO. Experiments are executed using an additional attention block at stage Res4.

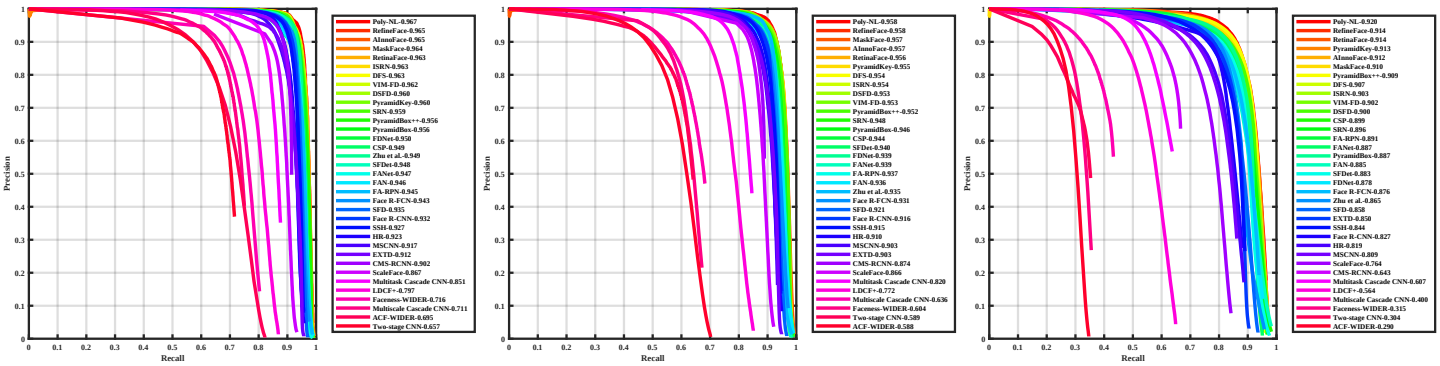


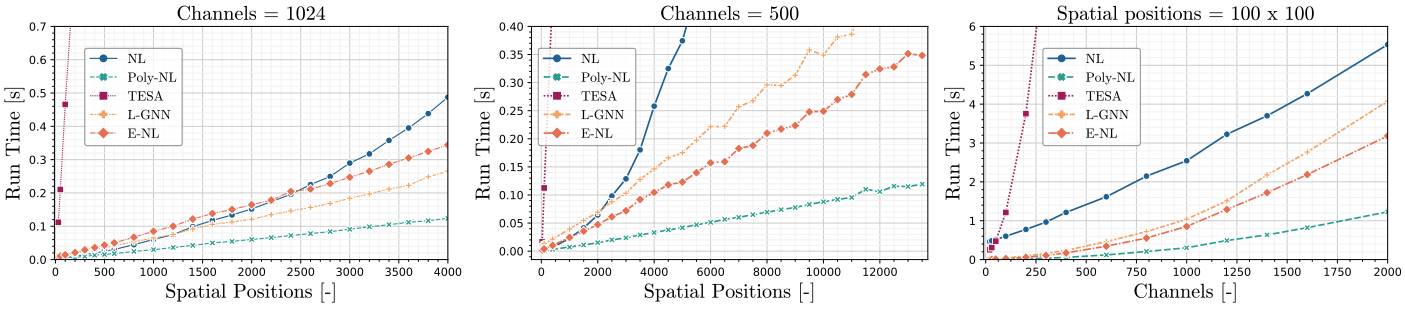
Figure 1: **Precision-recall curves** on the WIDER FACE test subsets of Easy, Medium, and Hard tracks. Different methods are presented in different colors. Images are better seen using an electronic display.

2. Face Detection

Poly-NL Applied on Lightweight Face Detector. Besides using ResNet-50 as backbone (in the main paper), we also apply the proposed Poly-NL to the lightweight face detector. More specifically, we use ResNet-18 with 1/4 of the traditional channels (*i.e*. the channel numbers of C2, C3, C4, and C5 are 16, 32, 64, and 128.) as the backbone. In Table 1b, We placed one additional attention block at stage Res4 and compared our Poly-NL method against other popular non-local blocks. The experiment shows consistent results with respect to the ResNet-50 architecture. Poly-NL outperforms all competitors on Easy, Medium, and Hard tracks, followed by TESA.

Efficiency. The proposed Poly-NL face detector using ResNet-50 as the backbone is efficient and only takes 39.5 milliseconds on 2080Ti GPU to perform inference on an image of VGA resolution (640×480). By using lower precision like 16 or 8 bits, the speed can be further improved by two to four times at the same cost. After we change the backbone from ResNet-50 to ResNet-18 (1/4 channels), the inference speed significantly improves to 200FPS on 2080Ti GPU and the lightweight face detector can easily run in real-time on mobile devices.

Precision-Recall Curves. To obtain the evaluation results from the WIDER FACE test set, we submitted the detection results of the proposed Ploy-NL face detector (ResNet-50) to the organizers. Different from the single-scale testing used in the main paper and Table 1b, we employ Test Time Augmentation(TTA) on the test set. More specifically, TTA includes multi-scale (the short edge of image is $[500, 800, 1100, 1400, 1700]$), shift (the direction is $[(0, 0), (0, 1), (1, 0), (1, 1)]$), horizontal flip and box voting. As shown in Figure 1, we compared our method to other recent state-of-the-art face detection algorithms (*e.g*. RefineFace [9], RetinaFace [2], PyramidBox [6], DSFD [4], SFDet [10], SFD [11], SSH [5], *etc*.). Our approach outperforms these state-of-the-art methods in terms of AP in all subsets, *i.e*. 96.7% (Easy), 95.8% (Medium) and 92.0% (Hard) for test set. On the "hard" subset, which contains a large number of tiny faces, our performance (average precision) is the best among all methods, indicating that the proposed Poly-NL can effectively enhance the context modeling to find tiny faces.



(a) (b) (c)

Figure 2: **Additional Runtime** comparison between Poly-NL and other non-local methods executed on a Intel(R) Core(TM) i9-9900X CPU.

Method	AP_b	AP_{b50}	AP_{b75}	AP_m	AP_{m50}	AP_{m75}
MaskR-CNN (R50)	37.9	59.2	41.0	34.6	56.0	36.9
+SE	38.1	59.5	40.8	34.8	56.3	36.8
+*Ours-2nd ord.*	38.1	59.3	41.3	34.7	56.1	36.9
+GC	38.8	60.3	42.1	35.2	56.9	37.4
+*Poly-NL*	**39.2**	**60.8**	**42.2**	**35.4**	**57.4**	**37.6**

Table 2: **Comparison** between Poly-NL and various $2nd$ order methods on Instance Segmentation (COCO). Poly-NL outperforms competitors thanks to the use of $3rd$ order interactions.

Visualisation Results. Figure 3 shows qualitative results on the WIDER FACE validation subset. We randomly pick 28 diverse images to highlight the results generated by the proposed Poly-NL face detector (ResNet-50). Detection results are shown by yellow rectangles and the brightness encodes the confidence. As can be seen, our face detector works very well in crowded scenes and can find hundreds of small faces in a wide variety of images, which indicates that the proposed Poly-NL face detector generalizes well in both indoor and outdoor scenes and under different lighting conditions, poses, and occlusions.

3. Instance Segmentation

Ablation on a deeper architecture. We provide an additional ablation study on the Poly-NL placement for the task of Instance Segmentation. We consider a MaskR-CNN architecture and use as backbone a ResNet-101 network pretrained on ImageNet. As presented in the main paper, we add one Poly-NL block at different stages of the baseline architecture and report results on Table 1a. Consistently with a ResNet-50 architecture, the use of Poly-NL improves the baseline performance if placed at stage Res3, Res4, and Res5. The use of a Poly-NL block at stage Res4 produces the best performance on both detection and segmentation, with an improvement of 1.4% $\uparrow$ in AP_{box} and 1% $\uparrow$ in AP_{mask}. Results show how Poly-NL is robust to different placement and architectures.

Comparison with $2nd$ order methods. Differently from our methods, CBAM [8], GCNet [1], and SE [3] process only $2nd$ order interactions by design. While $2nd$ order methods avoid the quadratic complexity found in NL, they consider a smaller set of interactions (i.e. $\sum_{c,e}^{N}\sum_{d,f}^{C} x_{(c,d)}x_{(e,f)}$). To highlight the importance of higher-level interactions we provide an ablation on instance segmentation on COCO in Tbl. 2. It is evident that in this task, where non-local patterns are crucial, substituting Poly-NL with $2nd$ order methods causes a drop in performance. We consider Eq. (6) of the main paper. Replacing $(\mathbf{XW_1} \odot \mathbf{XW_2})$ by only $\mathbf{XW_1}$ makes our method a $2nd$ order polynomial (*Ours-2nd ord.*) close to SE, that uses non-linearities and deploys a different pooling function. Our $2nd$ order results are on par with SE, lower than Poly-NL, and highlight the importance of transitioning to $3rd$ order interactions for performance improvements. GCNet proposes to use the same contribution for every position, making the conscious choice to lose part of the full interaction patterns. Compared to this method, Poly-NL retains access to every triplet of the original NL providing in return better performance.

4. On the Use of Non-Linearities.

Poly-NL starts with the assumption of an embedded dot-product similarity and doesn't use any activation in its implementation. As pointed out in Sec 3.2 and Tbl. 2.a of [7]

or Tbl. 1 in [1], non-local models are not sensitive to the choices of the similarity function therefore for the Non-Local block the use of softmax is not essential. The lack of in-place non-linearities does not impact efficiency trends nor prevent Poly-NL to outperform competitors (which make use of non-linear mappings). Figure 2 reports additional efficiency trends for various spatial-attention blocks, measured as CPU run time. As visible, Poly-NL exhibits significantly less overhead when compared to prior work.

5. Additional Attention Patterns.

We provide additional visualizations of Poly-NL attention patterns. We consider a single Poly-NL block inserted in a ResNet-50 architecture at stage Res4. We compare visual results of attention's features learned for the task of large-scale classification on Imagenet (Fig. 4) and instance-segmentation on COCO (Fig. 5). In both cases, we visualize the norm of the extracted features per spatial location over the input image. The output of the Poly-NL block, $\mathbf{Z}$, is presented together with the input $\mathbf{X}$ and the attention contribution $\mathbf{Y}$. The images show the versatility of the attention patterns learned by Poly-NL. Figure 4 shows results for the classification task, where the attention contribution highlights details overlooked by the input. The non-local information captured by the attention block is then merged with the input to produce the output $\mathbf{Z}$. Analogously, Figure 5 visualize results for the task of instance segmentation. In this case, the module learns to isolate noisy patterns in the image. The output of Poly-NL returns a cleaner version of the input, where the object instances are clearly visible.

References

[1] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. 3, 4

[2] Jiankang Deng, Jia Guo, Evangelos Ververas, Irene Kotsia, and Stefanos Zafeiriou. Retinaface: Single-shot multi-level face localisation in the wild. In *CVPR*, 2020. 2

[3] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 3

[4] Jian Li, Yabiao Wang, Changan Wang, Ying Tai, Jianjun Qian, Jian Yang, Chengjie Wang, Jilin Li, and Feiyue Huang. Dsfd: dual shot face detector. In *CVPR*, 2019. 2

[5] Mahyar Najibi, Pouya Samangouei, Rama Chellappa, and Larry S Davis. Ssh: Single stage headless face detector. In *ICCV*, 2017. 2

[6] Xu Tang, Daniel K Du, Zeqiang He, and Jingtuo Liu. Pyramidbox: A context-assisted single shot face detector. In *ECCV*, 2018. 2

[7] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7794–7803, 2018. 3

[8] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 3

[9] Shifeng Zhang, Cheng Chi, Zhen Lei, and Stan Z Li. Refineface: Refinement neural network for high performance face detection. *TPAMI*, 2019. 2

[10] Shifeng Zhang, Longyin Wen, Hailin Shi, Zhen Lei, Siwei Lyu, and Stan Z Li. Single-shot scale-aware network for real-time face detection. *IJCV*, 2019. 2

[11] Shifeng Zhang, Xiangyu Zhu, Zhen Lei, Hailin Shi, Xiaobo Wang, and Stan Z Li. S3fd: Single shot scale-invariant face detector. In *ICCV*, 2017. 2

Figure 3: Qualitative results on the validation set of the WIDER FACE dataset, where rectangles show the detection results. The detection confidence is represented by the brightness of the box (the color map at the end of the first row).

Figure 4: **Non-local dependencies for large-scale classification.** The image reports additional visualizations of the features learned by the Poly-NL module. Poly-NL captures long-range correlations and sums their contribution to the input features.

Figure 5: **Non-Local dependencies for instance-segmentation.** Poly-NL extrapolates background information and uses the attention contribution to highlight object's instances.