LambdaNetworks: Modeling Long-Range Interactions Without Attention

Irwan Bello

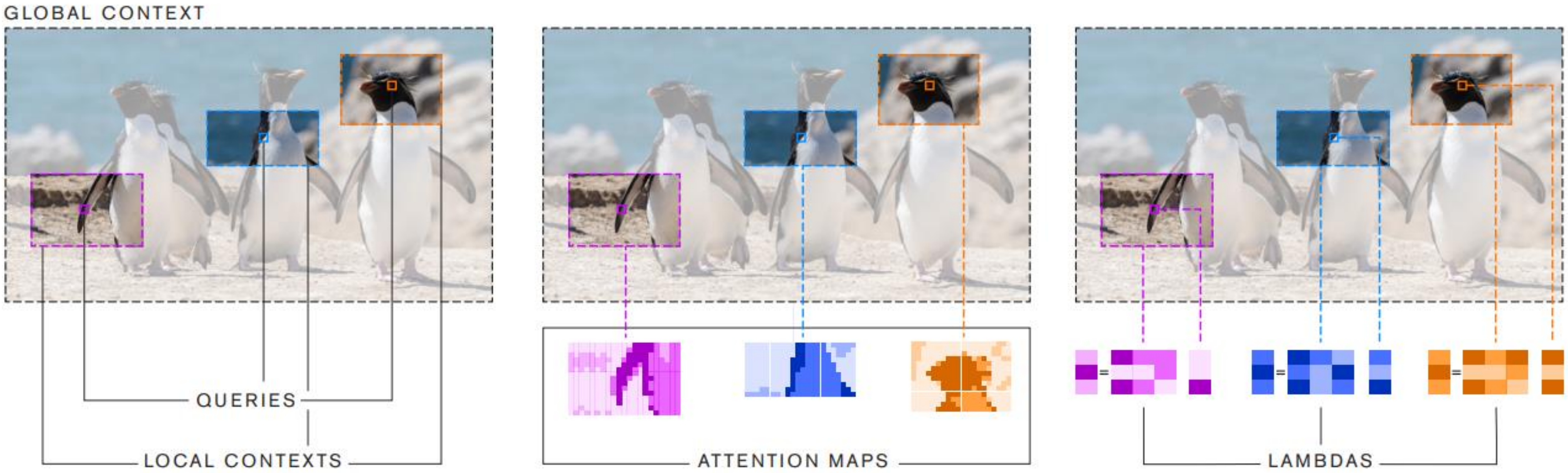
ICLR 2021

2021.03.29 윤주열

Long-range Interaction

- Allow long-range interaction without materializing attention maps

- Instead offer a summarization of the context termed "Lambda"



Lambda Layers

- Notations

Name	Description
$\|k\|, \|v\|$	query, value depth
$\boldsymbol{X} \in \mathbb{R}^{\|n\| \times d}$ $\boldsymbol{C} \in \mathbb{R}^{\|m\| \times d}$	inputs context
$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_Q \in \mathbb{R}^{\|n\| \times \|k\|}$ $\boldsymbol{K} = \boldsymbol{C}\boldsymbol{W}_K \in \mathbb{R}^{\|m\| \times \|k\|}$ $\boldsymbol{V} = \boldsymbol{C}\boldsymbol{W}_V \in \mathbb{R}^{\|m\| \times \|v\|}$ $\sigma(\boldsymbol{K}) = \text{softmax}(\boldsymbol{K}, \text{axis=}m)$ $\boldsymbol{E}_n \in \mathbb{R}^{\|m\| \times \|k\|}$	queries keys values normalized keys relative position embeddings
$\boldsymbol{\lambda}^c = \bar{\boldsymbol{K}}^T \boldsymbol{V} \in \mathbb{R}^{\|k\| \times \|v\|}$ $\boldsymbol{\lambda}^p_n = \boldsymbol{E}^T_n \boldsymbol{V} \in \mathbb{R}^{\|k\| \times \|v\|}$ $\boldsymbol{\lambda}_n = \boldsymbol{\lambda}^c + \boldsymbol{\lambda}^p_n \in \mathbb{R}^{\|k\| \times \|v\|}$	*content* lambda *position* lambdas lambdas

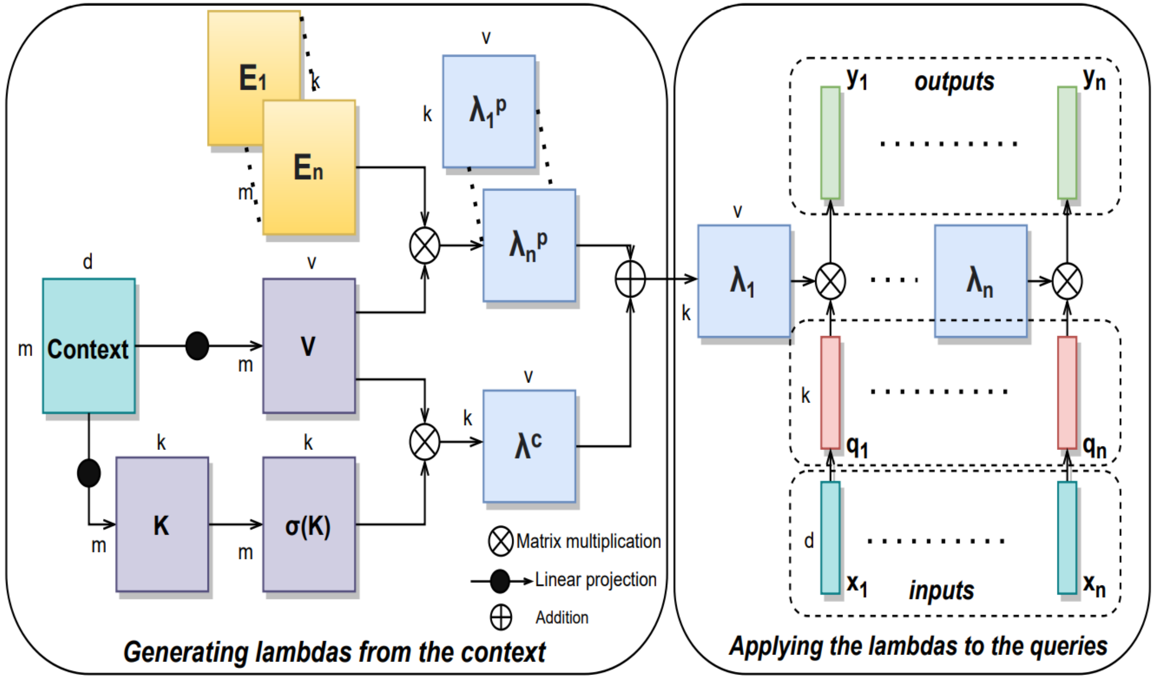
Lambda Layers

- Computation

$$\boldsymbol{\lambda}_n = \sum_m (\bar{\boldsymbol{k}}_m + \boldsymbol{e}_{nm})\boldsymbol{v}_m^T = \underbrace{\bar{\boldsymbol{K}}^T\boldsymbol{V}}_{\text{content lambda}} + \underbrace{\boldsymbol{E}_n^T\boldsymbol{V}}_{\text{position lambda}} \in \mathbb{R}^{|k|\times|v|}$$

$$\boldsymbol{y}_n = \boldsymbol{\lambda}_n^T \boldsymbol{q}_n = (\boldsymbol{\lambda}^c + \boldsymbol{\lambda}_n^p)^T \boldsymbol{q}_n \in \mathbb{R}^{|v|}$$

Name	Description
$\lvert k\rvert, \lvert v\rvert$	query, value depth
$\boldsymbol{X} \in \mathbb{R}^{\lvert n\rvert\times d}$ $\boldsymbol{C} \in \mathbb{R}^{\lvert m\rvert\times d}$	inputs context
$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_Q \in \mathbb{R}^{\lvert n\rvert\times\lvert k\rvert}$ $\boldsymbol{K} = \boldsymbol{C}\boldsymbol{W}_K \in \mathbb{R}^{\lvert m\rvert\times\lvert k\rvert}$ $\boldsymbol{V} = \boldsymbol{C}\boldsymbol{W}_V \in \mathbb{R}^{\lvert m\rvert\times\lvert v\rvert}$ $\sigma(\boldsymbol{K}) = \text{softmax}(\boldsymbol{K}, \text{axis}=m)$ $\boldsymbol{E}_n \in \mathbb{R}^{\lvert m\rvert\times\lvert k\rvert}$	queries keys values normalized keys relative position embeddings
$\boldsymbol{\lambda}^c = \bar{\boldsymbol{K}}^T\boldsymbol{V} \in \mathbb{R}^{\lvert k\rvert\times\lvert v\rvert}$ $\boldsymbol{\lambda}_n^p = \boldsymbol{E}_n^T\boldsymbol{V} \in \mathbb{R}^{\lvert k\rvert\times\lvert v\rvert}$ $\boldsymbol{\lambda}_n = \boldsymbol{\lambda}^c + \boldsymbol{\lambda}_n^p \in \mathbb{R}^{\lvert k\rvert\times\lvert v\rvert}$	*content* lambda *position* lambdas lambdas



Experiments

- Time and Space Complexity

(ResNet-50 Baseline)

Layer	Params (M)	top-1
Conv (He et al., 2016)†	25.6	76.9
Conv + channel attention (Hu et al., 2018c)†	28.1	77.6 (+0.7)
Conv + linear attention (Chen et al., 2018)	33.0	77.0
Conv + linear attention (Shen et al., 2018)	-	77.3 (+1.2)
Conv + relative self-attention (Bello et al., 2019)	25.8	77.7 (+1.3)
Local relative self-attention (Ramachandran et al., 2019)	18.0	77.4 (+0.5)
Local relative self-attention (Hu et al., 2019)	23.3	77.3 (+1.0)
Local relative self-attention (Zhao et al., 2020)	20.5	78.2 (+1.3)
Lambda layer	**15.0**	**78.4 (+1.5)**
Lambda layer ($\|u\|$=4)	**16.0**	**78.9 (+2.0)**

Architecture	Params (M)	Throughput	top-1
$\mathbf{C} \to \mathbf{C} \to \mathbf{C} \to \mathbf{C}$	25.6	7240 ex/s	76.9
$\mathbf{L} \to \mathbf{C} \to \mathbf{C} \to \mathbf{C}$	25.5	1880 ex/s	77.3
$\mathbf{L} \to \mathbf{L} \to \mathbf{C} \to \mathbf{C}$	25.0	1280 ex/s	77.2
$\mathbf{L} \to \mathbf{L} \to \mathbf{L} \to \mathbf{C}$	21.7	1160 ex/s	77.8
$\mathbf{L} \to \mathbf{L} \to \mathbf{L} \to \mathbf{L}$	15.0	1160 ex/s	78.4
$\mathbf{C} \to \mathbf{L} \to \mathbf{L} \to \mathbf{L}$	15.1	2200 ex/s	78.3
$\mathbf{C} \to \mathbf{C} \to \mathbf{L} \to \mathbf{L}$	15.4	4980 ex/s	78.3
$\mathbf{C} \to \mathbf{C} \to \mathbf{C} \to \mathbf{L}$	18.8	7160 ex/s	77.3

Layer	Space Complexity	Memory (GB)	Throughput	top-1
Global self-attention	$\Theta(blhn^2)$	120	OOM	OOM
Axial self-attention	$\Theta(blhn\sqrt{n})$	4.8	960 ex/s	77.5
Local self-attention (7x7)	$\Theta(blhnm)$	-	440 ex/s	77.4
Lambda layer	$\Theta(lkn^2)$	1.9	1160ex/s	**78.4**
Lambda layer ($\|k\|$=8)	$\Theta(lkn^2)$	0.95	**1640** ex/s	77.9
Lambda layer (shared embeddings)	$\Theta(kn^2)$	**0.63**	1210 ex/s	78.0
Lambda convolution (7x7)	$\Theta(lknm)$	-	1100 ex/s	78.1

Table 4: **The lambda layer reaches higher ImageNet accuracies while being faster and more memory-efficient than self-attention alternatives.** Memory is reported assuming full precision for a batch of 128 inputs using default hyperparameters. The memory cost for storing the lambdas matches the memory cost of activations in the rest of the network and is therefore ignored. b: batch size, h: number of heads/queries, n: input length, m: context length, k: query/key depth, l: number of layers.

Experiments

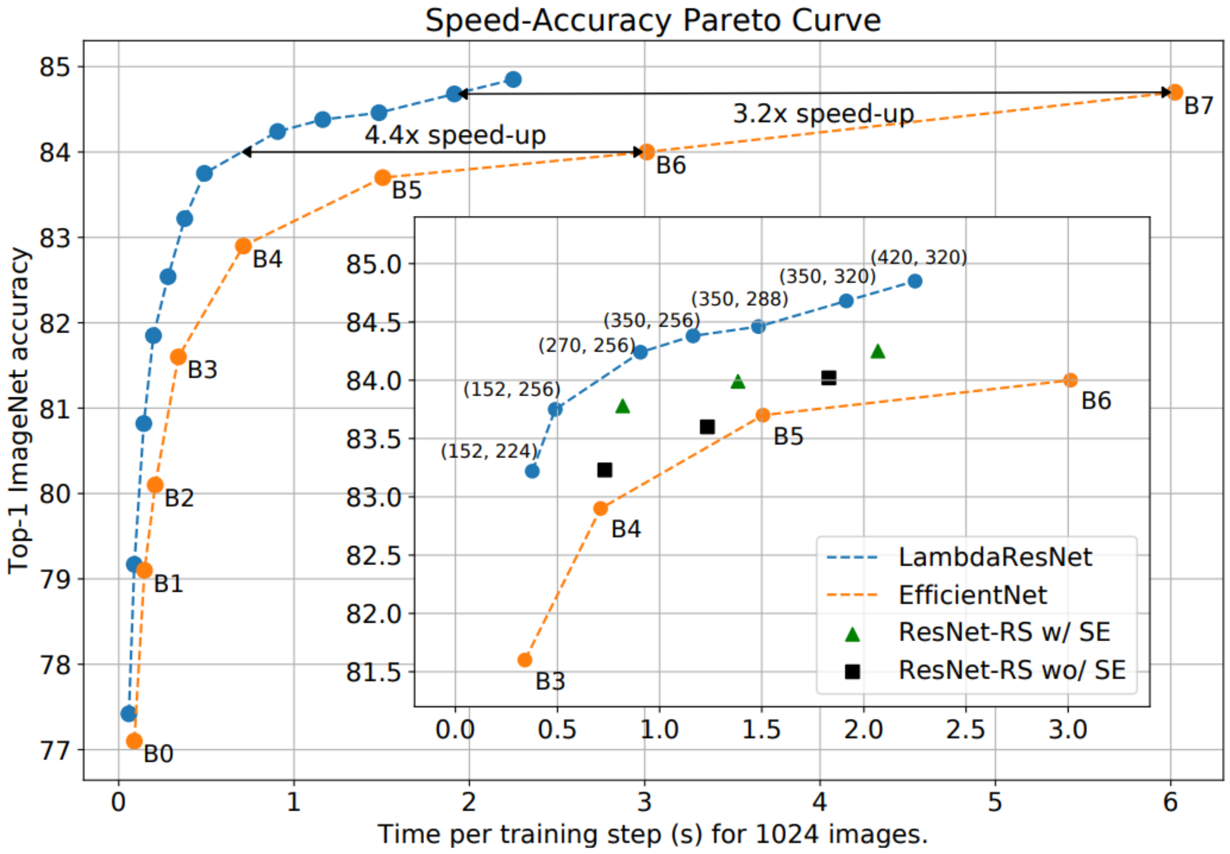
- Performance

Architecture	Params (M)	Train (ex/s)	Infer (ex/s)	ImageNet top-1
LambdaResNet-152	**51**	**1620**	**6100**	86.7
EfficientNet-B7	66	170 (9.5x)	980 (6.2x)	86.7
ViT-L/16	307	180 (9.0x)	640 (9.5x)	**87.1**

Table 5: **Comparison of models trained on extra data**. ViT-L/16 is pre-trained on JFT and fine-tuned on ImageNet at resolution 384x384, while EfficientNet and LambdaResNet are co-trained on ImageNet and JFT pseudo-labels. Training and inference throughput is shown for 8 TPUv3 cores.

Architecture	Params (M)	FLOPS (M)	top-1
MobileNet-v2	3.50	603	72.7
MobileNet-v2 with 2 lightweight lambda blocks	**3.21**	**563**	**73.3**

Table 17: **Lambda layers improve ImageNet accuracy in a resource-constrained scenario.** Replacing the 10-th and 16-th inverted bottleneck blocks with lightweight lambda blocks in the MobileNet-v2 architecture reduces parameters and flops by $\sim$10% while improving ImageNet accuracy by 0.6%.



Experiments

- Performance

Backbone	AP^{bb}_{coco}	$AP^{bb}_{s/m/l}$	AP^{mask}_{coco}	$AP^{mask}_{s/m/l}$
ResNet-101	48.2	29.9 / 50.9 / 64.9	42.6	24.2 / 45.6 / 60.0
ResNet-101 + SE	48.5 (+0.3)	29.9 (+0.0) / 51.5 / 65.3	42.8 (+0.2)	24.0 (-0.2) / 46.0 / 60.2
LambdaResNet-101	**49.4** (+1.2)	**31.7** (+1.8) / **52.2** / **65.6**	**43.5** (+0.9)	**25.9** (+1.7) / **46.5** / **60.8**
ResNet-152	48.9	29.9 / 51.8 / 66.0	43.2	24.2 / 46.1 / 61.2
ResNet-152 + SE	49.4 (+0.5)	30.0 (+0.1) / 52.3 / 66.7	43.5 (+0.3)	24.6 (+0.4) / 46.8 / 61.8
LambdaResNet-152	**50.0** (+1.1)	**31.8** (+1.9) / **53.4** / **67.0**	**43.9** (+0.7)	**25.5** (+1.3) / **47.3** / **62.0**

Table 14: **COCO object detection and instance segmentation with Mask-RCNN architecture on 1024x1024 inputs**. We compare LambdaResNets against ResNets with or without squeeze-and-excitation (SE) and report Mean Average Precision (AP) for small, medium, large objects ($AP_{s/m/l}$). Using lambda layers yields consistent gains across all object sizes, especially small objects.

Experiments

- Ablation Study

Table 8: **Ablations on the ImageNet classification task when using the lambda layer in a ResNet50 architecture.** All configurations outpeform the convolutional baseline at a lower parameter cost. As expected, we get additional improvements by increasing the query depth $|k|$ or intra-depth $|u|$. The number of heads is best set to intermediate values such as $|h|$=4. A large number of heads $|h|$ excessively decreases the value depth $|v| = d/|h|$, while a small number of heads translates to too few queries, both of which hurt performance.

$\lvert k \rvert$	$\lvert h \rvert$	$\lvert u \rvert$	Params (M)	top-1
ResNet baseline			25.6	76.9
8	2	1	14.8	77.2
8	16	1	15.6	77.9
2	4	1	14.7	77.4
4	4	1	14.7	77.6
8	4	1	14.8	77.9
16	4	1	15.0	78.4
32	4	1	15.4	78.4
2	8	1	14.7	77.8
4	8	1	14.7	77.7
8	8	1	14.7	77.9
16	8	1	15.1	78.1
32	8	1	15.7	78.5
8	8	4	15.3	78.4
8	8	8	16.0	78.6
16	4	4	16.0	78.9

Normalization	top-1
Softmax on keys (default)	78.4
Softmax on keys & Softmax on queries	78.1
L2 normalization on keys	78.0
No normalization on keys	70.0
No batch normalization on queries and values	76.2

Table 11: **Impact of normalization schemes in the lambda layer.** Normalization of the keys along the context spatial dimension m, normalization of the queries along the query depth k.

Content	Position	Params (M)	FLOPS (B)	top-1
✓	×	14.9	5.0	68.8
×	✓	14.9	11.9	78.1
✓	✓	14.9	12.0	78.4

Table 9: **Contributions of content and positional interactions**. As expected, positional interactions are crucial to perform well on the image classification task.