Title: LambdaNetworks: Modeling long-range Interactions without Attention
https://openreview.net/pdf?id=xTJEN-ggl1b
Authors: Under double blind review
Field: Computer Vision
Submitted: Sep 28

Modelling Long-Range Interactions:
Notation: q denotes a query**, c** denotes a context.
x is a scalar, $\mathbf{x}$ is a vector , X is a tensor.
dim $\mathbf{q_n}$ = k (key depth, eg. embedding dim)
dim value $(\mathbf{y_n})$ = v (value depth, eg. embedding dim)
intra depth = u =1 for classical transformers.

$|m|$ is the dimensionality of a set indexed by m
$(\mathbf{q_n}, \mathbf{c_m})$ denotes a content based relation (function or assignment)
$(\mathbf{q_n}, (\mathbf{n}, \mathbf{m}))$ denotes a position based relation.

Table 1: Hyperparameter, parameters and quantities of interest describing our lambda layer.

Name	Type	Description
$\|k\|$, $\|v\|$, $\|u\|$	hyperparameter	key/query depth, value depth, intra-depth
$\boldsymbol{W}_Q \in \mathbb{R}^{d\times\|k\|}$ $\boldsymbol{W}_K \in \mathbb{R}^{d\times\|k\|\times\|u\|}$ $\boldsymbol{W}_V \in \mathbb{R}^{d\times\|v\|\times\|u\|}$ $\boldsymbol{E}_{nm} \in \mathbb{R}^{\|k\|\times\|u\|}$	parameter	a tensor that linearly projects the inputs a tensor that linearly projects the context a tensor that linearly projects the context a positional embedding for the relation (n, m).
$\boldsymbol{X} \in \mathbb{R}^{\|n\|\times d}$ $\boldsymbol{C} \in \mathbb{R}^{\|m\|\times d}$	input	the inputs the context
$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_Q \in \mathbb{R}^{\|m\|\times\|k\|\times\|u\|}$ $\boldsymbol{K} = \boldsymbol{C}\boldsymbol{W}_K \in \mathbb{R}^{\|m\|\times\|k\|\times\|u\|}$ $\boldsymbol{V} = \boldsymbol{C}\boldsymbol{W}_V \in \mathbb{R}^{\|m\|\times\|v\|\times\|u\|}$ $\bar{\boldsymbol{K}} = \text{softmax}_m(\boldsymbol{K})$	activation	the queries the keys the values the normalized keys
$\boldsymbol{\mu}^c_m = \boldsymbol{K}_m\boldsymbol{V}^T_m \in \mathbb{R}^{\|k\|\times\|v\|}$ $\boldsymbol{\mu}^p_{nm} = \boldsymbol{E}_{nm}\boldsymbol{V}^T_m \in \mathbb{R}^{\|k\|\times\|v\|}$		*content* contribution from context element m *position* contribution from context element m
$\boldsymbol{Y} \in \mathbb{R}^{\|n\|\times d}$	outputs	the outputs

Keys:
Content based relations requires a k dimensional vector $\mathbf{k_m}$ (key) for each context
Position based relations require a k dimensional embedding vector $\mathbf{e_{mn}}$ for each position (m,n).

The output $Y = \{\mathbf{y_n}\}$ does not have a dimension of key depth v or position index m; these indices must be contracted (matrix multiply, dot product, diag(matrix) etc) away.

Classical attention contracts key and query over a common depth dimension.

Lambda layer maps each query to an output via a linear function $\mathbf{y_n} = \lambda(\mathbf{C})_\mathbf{n}(\mathbf{q_n})$.
Each lambda function is computed once, then acts independently of the context. It is then discarded after being applied to its query.

Lambda layer:

The lambda function has both a context and position part (equivalent to position embedding)
It takes the form of a k by v matrix.

$$\lambda^c = \sum_m softmax(K_m)V_m$$
$$\lambda_n^p = \sum_m E_{nm}V_m$$
$$\lambda_n = \lambda^c + \lambda_n^p$$

The input is transformed to a query via a weight matrix $\mathbf{q_n} = \mathbf{W_q x_n}$.
Then $\mathbf{y_n} = \mathbf{\lambda_n q_n}$ via a matrix multiplication.

Translation Equivalence:
To implement translation equivalence, define a relative position embedding for each (n,m) $R_r(m,n) = E_{m,n}$ where R is a (k,|r|,u) tensor and E is a (k,|n|,|m|,u) tensor. Where r indexes relative positions only.

Lambda Convolution:
For local attention, can generate a convolution with position lambda λ_p kernel. This can be done for Conv2D given 2d context, and is highly optimized.

Lambda convolution has linear time and space complexity wrt. to input length |m|.

Complexity
Batch size b
Time O(bknmv)
Space O(bknv + kmn)
Still quadratic (mn term). Note however the space complexity has no bmn term; the quadratic term does not scale with batch size, this allows for the processing of large batches.

Multiquery lambda:

Analogous to multi-head attention. However, an important distinction; multi-head attention increases representation power and complexity, multiquery lambda decreases representation power and complexity.

Procedure: split model dimension d into hv.

Results:

Table 3: **Comparison of the lambda layer and attention mechanisms on ImageNet classification with a ResNet50 architecture.** The lambda layer strongly outperforms alternatives at a fraction of the parameter cost. We include the reported improvements compared to the ResNet50 baseline in subscript to account for training setups that are not directly comparable. †: Our implementation.

Layer	Params (M)	top-1
Conv (He et al., 2016)†	25.6	$76.9_{+0.0}$
Conv + channel attention (Hu et al., 2018b)†	28.1	$77.6_{+0.7}$
Conv + double attention (Chen et al., 2018)	33.0	77.0
Conv + efficient 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**	$\mathbf{78.4}_{+1.5}$
Lambda layer ($\|u\|$=4)	**16.0**	$\mathbf{78.9}_{+2.0}$

Table 4: **The lambda layer reaches higher accuracies while being faster and more memory-efficient than self-attention alternatives.** Inference throughput is measured on 8 TPUv3 cores for a ResNet50 architecture with input resolution 224x224.

Layer	Complexity	Memory (GB)	Throughput	top-1
Global self-attention	$\Theta(blhn^2)$	120	OOM	OOM
Axial self-attention	$\Theta(blhn\sqrt{n})$	4.8	960ex/s	77.5
Local self-attention (7x7)	$\Theta(blhnm)$	-	440ex/s	77.4
Lambda layer	$\Theta(lkn^2)$	0.96	1160ex/s	**78.4**
Lambda layer (shared embeddings)	$\Theta(kn^2)$	0.31	1210ex/s	78.0
Lambda layer ($\|k\|$=8)	$\Theta(lkn^2)$	0.48	**1640**ex/s	77.9
Lambda convolution (7x7)	$\Theta(lknm)$	-	1100ex/s	78.1

Table 5: LambdaResNets improve upon the parameter-efficiency of large EfficientNets.

Architecture	Params (M)	top-1
EfficientNet-B6	43	84.0
LambdaResNet152	**35**	84.0
LambdaResNet200	42	**84.3**

Table 6: LambdaResNets improve upon the flops-efficiency of large EfficientNets.

Architecture	Flops (G)	top-1
EfficientNet-B6	38	**84.0**
LambdaResNet-270	**34**	**84.0**

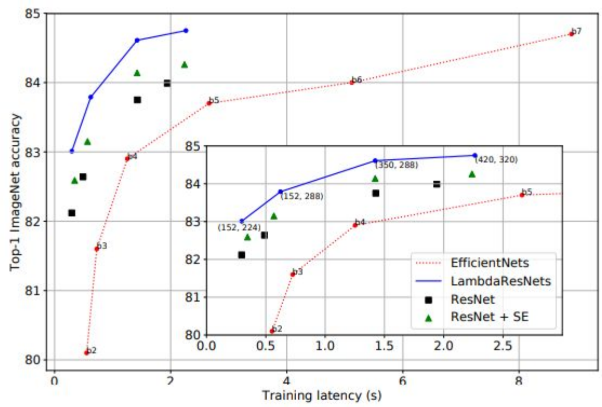

Figure 2: LambdaResNets are ~4.5x faster than EfficientNets and substantially improve the speed-accuracy tradeoff of image classification models[3]across different (depth, image size) scales.

Implementation:

$$\begin{aligned}
\boldsymbol{\lambda}^c_{bkv} &= einsum(\bar{\boldsymbol{K}}_{bmku}, \boldsymbol{V}_{bmvu}) \\
\boldsymbol{\lambda}^p_{bnkv} &= einsum(\boldsymbol{E}_{knmu}, \boldsymbol{V}_{bmvu}) \\
\boldsymbol{Y}^c_{bnhv} &= einsum(\boldsymbol{Q}_{bnhk}, \boldsymbol{\lambda}^c_{bkv}) \\
\boldsymbol{Y}^p_{bnhv} &= einsum(\boldsymbol{Q}_{bnhk}, \boldsymbol{\lambda}^p_{bnkv}) \\
\boldsymbol{Y}_{bnhv} &= \boldsymbol{Y}^c_{bnhv} + \boldsymbol{Y}^p_{bnhv}
\end{aligned}$$

Masked Context Lambda:
Masked attention is achieved by zeroing out certain elements of the attention tensor.
For lambda:

$$\begin{aligned}
\boldsymbol{\mu}^c_{bmkv} &= einsum(\boldsymbol{K}_{bmku}, \boldsymbol{V}_{bmvu}) \\
\boldsymbol{\lambda}^c_{bnkv} &= einsum(\boldsymbol{P}_{nm}, \boldsymbol{\mu}_{bmkv}) \\
\boldsymbol{\lambda}^p_{bnkv} &= einsum(\boldsymbol{E}_{knmu} * \boldsymbol{P}_{nm}, \boldsymbol{V}_{bmvu})
\end{aligned}$$

where $p_{nm} = 1[m \in \mathcal{C}_n]$ and $*$ is a broadcasted element-wise multiplication.

p_{mn} is a masking tensor (forward or causal) i.e. is 1 when in context, 0 when out of context(masked).