LambdaNetworks: Efficient & accurate, but also accessible? A reproducibility project with CIFAR-10.

de Alvear Cárdenas, J.I. † and de Vries, W.A.J.G.‡

†Department of Control & Simulation, Faculty of Aerospace Engineering, Delft University of Technology
‡Department of Space Systems Engineering, Faculty of Aerospace Engineering, Delft University of Technology

Introduction

This work builds on the foundations laid by Irwan Bello in "LambdaNetworks: Modeling long-range interactions without attention" [1]. Bello proposes a method where long-range interactions are modelled by layers which transform contexts into linear functions called lambdas, in order to avoid the use of attention maps. The great advantage of lambda layers is that they require much less compute than self-attention mechanisms. This is fantastic, because it does not only provide results faster, but also saves money and has a more favourable carbon footprint! However, Bello still uses 32 TPUv3s and the 200 GB sized ImageNet classification dataset. Therefore, we started this reproducibility project wondering: Could lambda layers be scaled to mainstream computers while keeping its attractive properties?

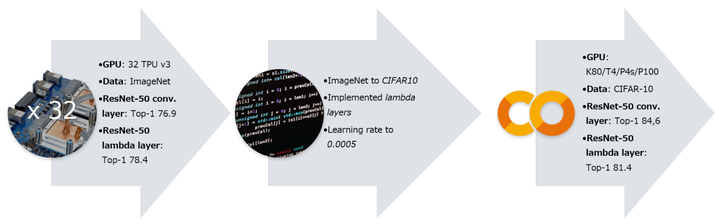
In 2021 the world did not only have to deal with the COVID-19 epidemic but was struck by chip shortages as well due to increase in consumer electronics for working at home, shut down factories in China and the rising prices of crypto-currencies. This has decreased supply to record lows and prices to record highs. Resulting in a situation, whereby researchers, academics, and students (who are all usually on a budget) are no longer able to quickly build a cluster out of COTS (commercial off-the-shelf) GPUs resulting in having to deal with older, less, and less efficient hardware.

No official code was released at the time of starting the project. Therefore, it is up to us to reproduce the paper by Bello as accurately as possible while trying to scale it down such that it can be run on an average consumer computer.

Goals of this reproducibility project

They are twofold. Firstly, attention seems to be state-of-the-art at the moment with a lot of ongoing research. However, attention has some shortcomings that lambda layers aims to fix while, at the same time, slightly increasing the accuracy. Therefore, reproducing the lambda layers can contribute to the early future adoption of this potentially superior algorithm by the community. Additionally, its implementation on a lower dimensional dataset proves the robustness of the algorithm, as well as its potential implementation on resource constrained devices (TinyML).

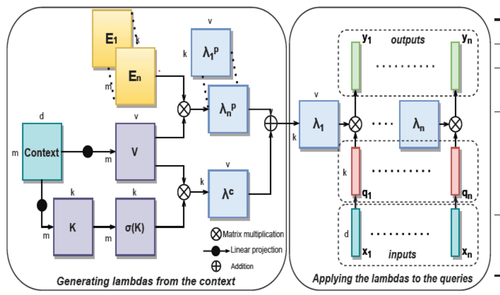
The second reason is the enhancement and broadening of the deep learning knowledge and skills of the authors of the here presented reproducibility project. In order to get a feel for Deep-Learning it is not only important to read from the great number of online resources, but also to have hands-on experience with some groundbreaking papers. After having read the well-received paper called "Attention Is All You Need" [2], we were excited by the whole pool of papers that could follow-up. From the abstract, lambda layers promised to be a great advancement from Transformers, targeting multiple weakness and leading to greater performance and lower computational load. Eager to learn more about attention and thrilled by the potential of lambda layers, choosing this reproducibility project was an obvious choice for us.



Lambda layer implementation

It consists of three main computation components which can be visualised in Figure 2:

1. **Content lambda** which encapsulates the context content.
2. **Position lambda** which encapsulates the query-context relative position information.
3. **Lambdas applied to queries** for the computation of the output.



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

Fig. 2: Computational graph of the lambda layer [1]

Apart from these steps, in the original paper it was observed that the reduction of the value dimension |v| could greatly reduce the computational cost, as well as the space and time complexities. Therefore, the author decides to decouple these complexities from this dimension by manipulating its value at will. Bello proposes using |h| queries q^h_n for each (pixel) input to which the same lambda is applied. Then the output for a single (pixel) input is the result of the concatenation of each of the h outputs as $y_n = \text{concat}(\lambda_n q^1_n, \cdots, \lambda_n q^{|h|}_n)$. Consequently, |v| is now equal to d/|h|, which reduces the complexity by a factor of |h|. This reduction in the dimensionality of the values is called by the author **multi-query lambda layers**.

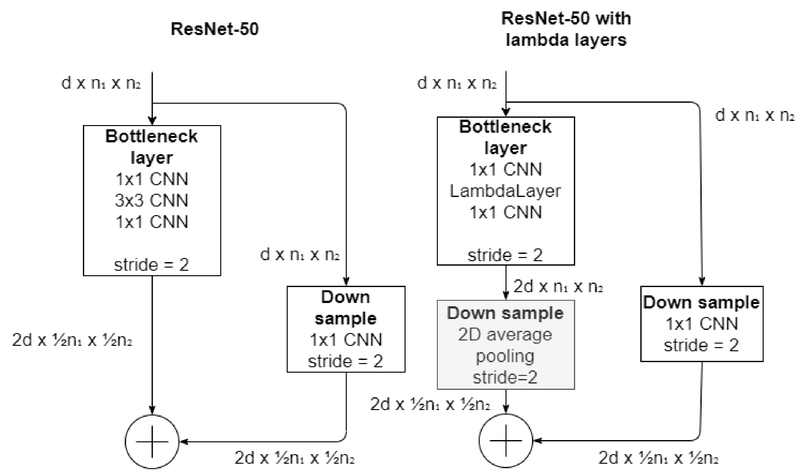


Fig. 3: Lambda layer integration within ResNet-50.

Results

The main claims of the original paper on lambda layers are their superior performance and higher computationally efficiency when compared to their convolutional and attention counterparts. Therefore, we compared the accuracy of the original ResNet-50 to its modified version with lambda layers (left Figure 4), as well as their required training computation times and throughput (Figure 5). Additionally, a brief sensitivity analysis was performed on the initial learning rate in order to tune this hyperparameter to the new architecture-dataset combination (right Figure 4). The best initial learning rate found is 0.0005.

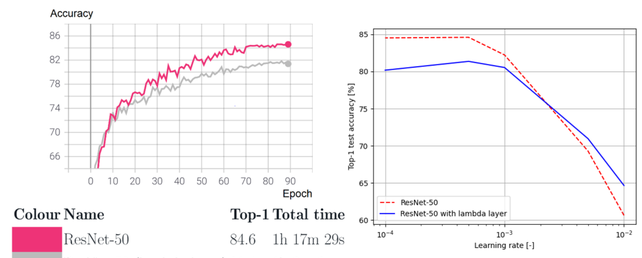


Fig. 4: (Left) The accuracy diagram with the top-1 vs epochs for the test dataset of the ResNet-50 and lambda layer architectures. The initial learning rate equals 0.0005. (Right) Accuracy as a function of the initial learning rate for the ResNet-50 and lambda layer architectures.

	Baseline	Lambda
Throughput (ex/s)	981.83	1283.26

Fig. 5: Throughput of the baseline and lambda models.

Conclusions

1. When trained on a lower dimensional dataset as CIFAR-10, lambda layers do not outperform the convolutional counterparts; however, they still reach competitive results.
2. On the ImageNet dataset, Bello reports a baseline accuracy of 76.9% and a lambda layer accuracy of 78.4%. The accuracy of both architectures increases on CIFAR-10. This observation is alluded to the lower difficulty of classifying an image among 10 classes versus 1000.
3. The lambda layer has a higher throughput than the convolutional counterpart, namely 23.5% higher. This results in lower training times.
4. The best initial learning rate is 0.0005 for both architectures.

References

[1] Irwan Bello. "LambdaNetworks: Modeling Long-Range Interactions Without Attention". In: (2021). arXiv: 2102.08602 [cs.CV].

[2] Ashish Vaswani et al. "Attention is all you need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Long Beach, CA: Curran Associates, Inc., 2017.