LAUREL: Learned Augmented Residual Layer

Gaurav Menghani [1] **Ravi Kumar** [1] **Sanjiv Kumar** [2]

Abstract

One of the core pillars of efficient deep learning methods is architectural improvements such as the residual/skip connection, which has led to significantly better model convergence and quality. Since then the residual connection has become ubiquitous in not just convolutional neural networks but also transformer-based architectures, the backbone of LLMs.

In this paper we introduce *Learned Augmented Residual Layer* (LAUREL)—a novel generalization of the canonical residual connection—with the goal to be an in-situ replacement of the latter while outperforming on both model quality and footprint metrics. Our experiments show that using LAUREL can help boost performance for both vision and language models. For example, on the ResNet-50, ImageNet 1K task, it achieves 60% of the gains from adding an extra layer, while only adding 0.003% more parameters, and matches it while adding $2.6\times$ fewer parameters.

1. Introduction

Model efficiency is of critical importance in the age of extremely large language and vision models. Even if a given model's quality is good, its footprint metrics such as train-time compute required, inference latency, resident memory size, etc. dictate if it can be experimented with and/or deployed in real-world settings. These metrics are directly tied to the financial costs of deploying the model in production and user-perceived responsiveness of systems dependent on these models.

Consequently, improving the Pareto-frontier of model quality vs footprint, via efficient deep learning methods has been an area of active research in the past few years. Areas of interests span from algorithmic techniques (Menghani, 2023), to efficient hardware (Sze et al., 2017), to best practices around model efficiency (Dehghani et al., 2022), etc.

One of the core pillars of efficient deep learning methods is architectural improvements such as the residual/skip connection, which had led to significantly better model convergence and quality (He et al.). Since then the residual connection has become ubiquitous in not just convolutional neural networks but also transformer-based architectures (Vaswani et al., 2017), the backbone of LLMs.

In this paper we introduce *learned augmented residual layer*, LAUREL, which generalizes the canonical residual connection. Recall that deep-learning models with residual connections have a 'block' structure, with many blocks chained together between the input and final output; these could be convolution/identity blocks within a ResNet, a transformer block in a transformer encoder/decoder, etc. Within a block, a typical residual connection is given by:

$$x_{i+1} = f(x_i) + x_i. \tag{1}$$

Here, $f(\cdot)$ can be any non-linear function such as attention, MLP, multiple non-linear layers, etc., x_i is the input to the said non-linear function, and x_{i+1} is the combined output of the non-linear function and the residual component. Refer to Figure 1 for an illustration. To simplify exposition, we ignore pre-processing functions such as layer norm, which can be folded into $f(\cdot)$ without loss of generality.

2. Learned Augmented Residual Layer

In this section we describe the main idea behind LAUREL. In its most general form, we reformulate the residual connection to be the following:

$$x_{i+1} = \alpha \cdot f(x_i) + g(x_i, x_{i-1}, \ldots, x_0). \tag{2}$$

Here α is a learned scalar parameter, and $g(\cdot)$ is a learned linear function with $x_i, x_{i-1}, \ldots, x_0$ as inputs, where x_j is the output of the jth residual connection. The intuition behind LAUREL is that one can learn a richer set of (linear) functions than just using x_i as the residual component. One motivation behind seeking these richer linear functions is the concept of a "residual stream" (Elhage et al., 2021), where the residual connection is considered to be part of a stream of information that passes through each layer without being exposed to any non-linearities. This allows the learning process to focus on the non-linear components better.

Each layer / operation can read from, and subsequently write to this residual stream based on what it read. Given that

[1]Google Research, Mountain View, CA. gmenghani@google.com, ravi.k53@gmail.com
[2]Google Research, New York, NY. sanjivk@google.com.

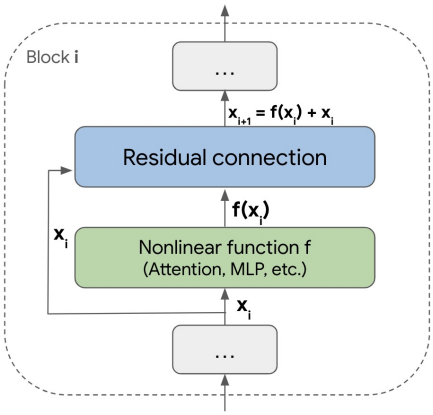

Figure 1. A standard residual connection. We assume the model to be divided into logical 'blocks', which is true for most modern architectures including transformers. The residual connection combines the output of a non-linear function f and the input to the said non-linear function. Here, f can be attention, MLP, or any other combination of non-linear layers.

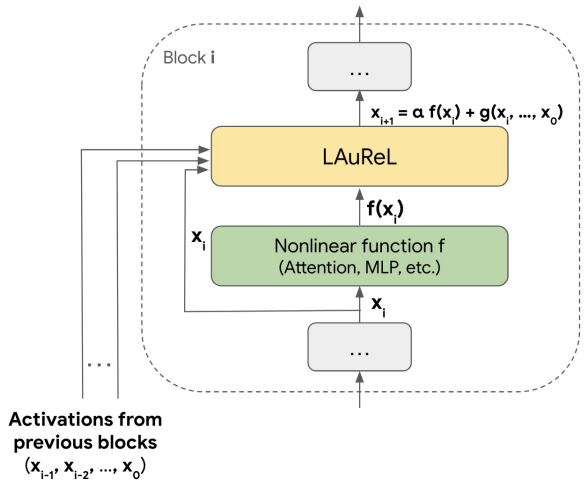

Figure 2. An illustration of the LAUREL framework; see (2). LAUREL can be used to replace the regular residual connection in Figure 1. Again, f can be any non-linear function such as attention, MLPs, and groups of multiple non-linear layers.

the residual connection has been shown to be important for model quality and convergence, we designed LAUREL to operate on this residual stream in a learned fashion, while being light-weight in terms of the model size and latency changes.

In this paper we study three specific versions of the LAUREL framework; although as described in (2), the framework can be generalized beyond these versions.

2.1. Residual Weights Version (LAUREL-RW)

In this version, we keep α learnable and set $g(x_i, \ldots, x_0) = \beta x_i$. Therefore, (2) can be rewritten as:

$$x_{i+1} = \alpha f(x_i) + \beta x_i.$$

Notice that this version assigns learnable weights to the $f(x_i)$ and x_i from (1). In practice, we found that we cannot let α and β grow unbounded, and using a normalization function such as `softmax` helps. Clearly, this version will add only two new parameters per LAUREL layer. If necessary, we can always replace these two parameters by a single learnable parameter and use the `sigmoid` function to define α, β in terms of this single parameter.

2.2. Low-Rank Version (LAUREL-LR)

In this version, we fix $\alpha = 1$, and $g(x_i) = Wx_i$ in (2) to obtain

$$x_{i+1} = f(x_i) + Wx_i,$$

where W is learnable. Note that, as written, W is a $D \times D$ matrix, where D is the model dimension; hence this will add D^2 new parameters (per LAUREL layer) to the model.

In practice, to reduce the number of new parameters added to the model and to help with convergence, we consider a low rank version of W. In particular, let $W = A \times B + I$, where A and B^T are $D \times r$ matrices and $r \ll D$. Thus, we can rewrite (2) as:

$$x_{i+1} = f(x_i) + ABx_i + x_i. \tag{3}$$

Here, both A and B matrices are learnable. The number of new parameters is $2rD$.

2.3. Previous Activations Version (LAUREL-PA)

In this version, we use use activations from previous blocks. In particular, we set $g(x_i) = \sum_{j=0}^{j=i} \gamma_j h(x_j)$, where $\gamma_0, \ldots, \gamma_j$ are learned parameters and h is another linear function.[1] This allows us to rewrite (2) as:

$$x_{i+1} = f(x_i) + \sum_{j=0}^{i} \gamma_j \cdot h(x_j).$$

In practice, we replace h by a low-rank product similar to the LAUREL-LR version. The number of new parameters is $2rD + N$, where N is the number of layers.

Note that, all three versions above are a combination of scalar and/or low-rank products on top of the vanilla residual connection in (1). This makes LAUREL especially light-weight in terms of its impact on model size and latency. Moreover, the framework is generic enough to allow combinations of the above versions, as well as new versions.

[1] For simplicity, we fix $\alpha = 1$.

3. Experiments

We experiment with LAUREL in two domains, namely, vision and language. For the first case, our goal is to improve the image classification accuracy of the ResNet-50 model on the ImageNet-1K dataset (Deng et al., 2009). For the second case our goal is to improve the performance of an LLM, evaluated after the pre-training stage, on common benchmarks.

The underlying motivation behind these experiments is not necessarily to improve on the SOTA results, but to show how LAUREL can be easily integrated on top of common model architectures with residual/skip connections in order to achieve a better model quality and footprint trade off.

3.1. ResNet-50 on ImageNet-1K

In this setup we train a standard ResNet-50 model on the ImageNet 1K dataset (Deng et al., 2009) using 16 Cloud TPUv5e chips over one epoch with data-augmentation turned on. In order to obtain a strong baseline, we fine-tuned the model learning rate schedule and picked a schedule that maximized the average of the best accuracy@1 values over 5 trials (which we simply refer to as accuracy in this subsection). The baseline model that we obtained achieves an accuracy of $74.95 \pm 0.016\%$.

In addition, we also find that if we simply add another layer to the ResNet-50 model (i.e., naive scaling), we can increase the model's accuracy by 0.25% to reach 75.20%, while adding 4.37% new parameters. With that in context, applying LAUREL on the model leads to better results (see Table 1).

If we only use the LAUREL-RW version, we get an improvement of 0.15% on average with only 0.003% extra parameters, which is essentially negligible. When we try the LAUREL-RW+LR version with $r = 16$, we achieve an accuracy of 75.20% while adding only 1.68% extra parameters; this matches the performance of the baseline with an extra layer, while using $2.6\times$ fewer extra parameters. Additionally, when we use the combined LAUREL-RW+LR+PA version we improve the accuracy to 75.25% while still using $1.82\times$ fewer extra parameters than the baseline with one extra layer, demonstrating that LAUREL is superior to naively scaling the model. Notably even though we make fundamental changes to the residual connection we did not find any training instabilities when using LAUREL.

3.2. Decoder-only LLM Pre-training

In this setup, our goal is to test the performance of LAUREL with Large Language Models (LLMs). For our baseline, we chose a 3B parameter decoder-only model based on the Transformer architecture. We pre-trained both the baseline, and our experiment with LAUREL, from scratch; we use the LAUREL-RW and LAUREL-LR versions (with $r = 4$). Both the models were trained using 1024 Cloud TPU v5e chips for approximately two weeks each, using a pre-training mixture consisting of only text tokens.

Table 1. Applying LAUREL on a ResNet-50 trained on the ImageNet 1K classification dataset.

MODEL	AVG. BEST ACCURACY@1 (%), 5 TRIALS	PARAMS ADDED VS BASELINE (%)
BASELINE	74.95 ± 0.01	-
BASELINE + 1 LAYER (NAIVE SCALING)	75.20 ± 0.12	4.37
LAUREL-RW	75.10 ± 0.10	0.003
LAUREL-RW+LR	75.20 ± 0.07	1.68
LAUREL-RW+LR+PA	75.25 ± 0.09	2.40

It is worth noting that the combined LAUREL-RW+LR variant adds only 0.012% more parameters as compared to the baseline model. Since we chose $r = 4$, the total number of parameters added by LAUREL-LR is $8ND$. Typically $N \in [10, 100]$ and $D \in [500, 5000]$. Thus, the number of new parameters is dwarfed by that of the original model.

We evaluated both the pre-trained baseline and LAUREL models on a host of common LLM tasks such as Q&A, NLU, Math, Code, etc; see Table 2 for the results. LAUREL outperforms the baselines on all tasks except on the MBPP dataset where it was neutral. Interestingly, these can be achieved with only 0.012% extra parameters.

Since the pre-training was computationally expensive, we tried a single value of r and did not try the LAUREL-PA version. It is possible that trying different variants similar to the ResNet experiments might improve the results further.

3.3. LAUREL-LR: Rank vs Accuracy

We note that for the LAUREL-LR version on the ResNet-50/ImageNet combination, there is a pattern in terms of the best accuracy achieved with different values of r. In the combined LAUREL-RW+LR version we experimented with different values of r, and computed the average of the best accuracy@1 achieved over 5 trials; see Figure 3. From Table 1, with the LAUREL-RW version alone we already achieve an average best accuracy@1 of 75.10%, therefore for the combined LAUREL-RW+LR version we would like to see the accuracy exceeding that.

We observe that when r is small ($r \in \{4, 8\}$), there is not a significant improvement over the baseline LAUREL-RW experiment. This could be because a very small r acts as an information bottleneck in the low-rank product in

Table 2. Evaluation results of a Decoder-only LLM pre-trained from scratch with (a) the baseline model architecture, and (b) using LAUREL. We evaluated both the models on a number of common eval benchmarks (higher is better for all task and dataset combinations listed below). The LAUREL variant outperforms the baseline on all but one dataset while adding only 0.012% extra parameters.

TASK TYPE	DATASET	BASELINE	LAUREL
Q&A	BOOLQ	58.07	**65.66**
	TYDI QA (GOLDP)	67.98	**72.58**
MULTI-TASK NLU	MMLU	25.72	**25.89**
MATH	MATH	3.54	**3.70**
	GSM8K-COT	8.34	**8.79**
SENTENCE COMPLETION	HELLASWAG	64.84	**65.06**
CODE	HUMANEVAL	18.29	**18.90**
	MBPP	27.00	27.00
	GSM8K-PAL	10.31	**11.37**

(3). As r increases, the accuracy reaches the maximum for $r = 16, 32$; beyond this, the accuracy seems to drop though still higher than the LAUREL-RW baseline. We believe this unimodal phenomenon could be due to the number of parameters added to the model, which increases linearly in r, which would require appropriate tuning of hyper-parameters such as the learning rate as well as the regularization penalty.

4. Related Work

Since our larger goal is to improve the training and inference efficiency of deep learning models, we briefly discuss some research directions aimed at improving model efficiency.

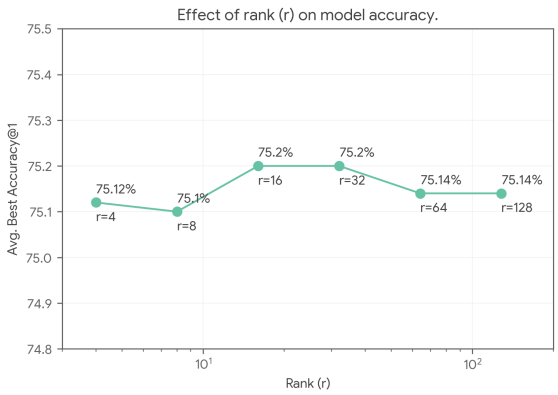

Figure 3. Trend of accuracy as r is varied in the LAUREL-RW+LR version.

Architectural Changes: Our work is inspired by recent model architecture improvements such as LoRA (Hu et al., 2022) and AltUp (Baykal et al., 2023) amongst others. However, they are not directly relevant to LAUREL. Indeed, LoRA is designed to efficiently fine-tune large pre-trained models and it works directly on the model weight matrices level by introducing low-rank 'adapter' weights that are learned during the fine-tuning stage, while other model weights are held constant. In contrast, LAUREL works at the residual connection level, which likely spans multiple weight matrices involved in the function f; furthermore, it is applied during the pre-training stage.

AltUp (Baykal et al., 2023) is designed to replicate the quality improvements of a model with a large model dimension, without having to pay the additional cost. It operates at the transformer-block level, constructing parallel 'lightweight' transformer blocks to approximate the model dimension scaling effect. In contrast, LAUREL operates at the residual connection level and does not aim to replicate the dimension scaling effect.

Interestingly, LAUREL can be applied in conjunction with both LoRA (during fine-tuning) and AltUp (during pre-training and fine-tuning).

He & Hofmann (2023) proposes several changes to transformer blocks to help improve model convergence; however these proposals are limited to transformer blocks.

Compression Techniques: Model compression techniques (Buciluǎ et al., 2006) such as quantization (Krishnamoorthi, 2018; Jacob et al., 2018), including ternary networks (Li et al., 2016; Ma et al., 2024) are commonly used to reduce model size and inference latency. Similarly, pruning and model sparsity (Gale et al., 2019; Liu et al., 2019) techniques have also been explored and implemented in hardware.

Learning Techniques: Distillation (Hinton et al., 2014) is a popular technique for improving a smaller (student) model quality using "soft-labels" from a larger, impractical (teacher) model (Sanh et al., 2019). Some distillation variants propose learning intermediate representations as well (Zagoruyko & Komodakis, 2016; Kim et al., 2023). Other techniques include works like Stacking (Reddi et al., 2023) and RaPTr (Panigrahi et al., 2024), which progressively grow and train the network to achieve an improved model quality while reducing model training time.

5. Conclusion

In this paper we introduce the LAUREL framework, which is a novel architectural change and a generalization of the residual / skip connection aimed at improving the model quality without significantly increasing the model size or latency. We study three versions (LAUREL-RW, LAUREL-

LR, LAUREL-PA) that can be mixed-and-matched together, as we show in our experiments.

Through experiments, we demonstrate the efficacy of replacing the conventional residual connection with LAUREL on both vision and language tasks, while also providing evidence for its advantages over naive model scaling methods. In the future, we would like to try LAUREL and its variants on other architectures such as Vision Transformers (ViT) (Dosovitskiy et al., 2020) and related tasks.

References

Baykal, C., Cutler, D., Dikkala, N., Ghosh, N., Panigrahy, R., and Wang, X. Alternating updates for efficient transformers. In *NeurIPS*, pp. 76718–76736, 2023.

Buciluǎ, C., Caruana, R., and Niculescu-Mizil, A. Model compression. In *KDD*, pp. 535–541, 2006.

Dehghani, M., Tay, Y., Arnab, A., Beyer, L., and Vaswani, A. The efficiency misnomer. In *ICLR*, 2022.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *CVPR*, pp. 248–255, 2009.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations*, 2020.

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., DasSarma, N., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Gale, T., Elsen, E., and Hooker, S. The state of sparsity in deep neural networks. *arXiv*, 1902.09574, 2019.

He, B. and Hofmann, T. Simplifying Transformer Blocks. *arXiv*, 2311.01906, 2023.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. In *CVPR*, pp. 27–30.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. In *Deep Learning Workshop (NeurIPS)*, 2014.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *CVPR*, pp. 2704–2713, 2018.

Kim, S., Rawat, A. S., Zaheer, M., Jayasumana, S., Sadhanala, V., Jitkrittum, W., Menon, A. K., Fergus, R., and Kumar, S. EmbedDistill: A Geometric Knowledge Distillation for Information Retrieval. *arXiv*, 2301.12005, 2023.

Krishnamoorthi, R. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv*, 1806.08342, 2018.

Li, F., Zhang, B., and Liu, B. Ternary weight networks. In *NIPS*, 2016.

Liu, Z., Sun, M., Zhou, T., Huang, G., and Darrell, T. Rethinking the value of network pruning. In *ICLR*, 2019.

Ma, S., Wang, H., Ma, L., Wang, L., Wang, W., Huang, S., Dong, L., Wang, R., Xue, J., and Wei, F. The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits. *arXiv*, 2402.17764, 2024.

Menghani, G. Efficient deep learning: A survey on making deep learning models smaller, faster, and better. *ACM Computing Surveys*, 2023.

Panigrahi, A., Saunshi, N., Lyu, K., Miryoosefi, S., Reddi, S., Kale, S., and Kumar, S. Efficient Stagewise Pretraining via Progressive Subnetworks. *arXiv*, 2402.05913, 2024.

Reddi, S. J., Miryoosefi, S., Karp, S., Krishnan, S., Kale, S., Kim, S., and Kumar, S. Efficient Training of Language Models using Few-Shot Learning. In *ICML*, pp. 14553–14568. 2023.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In *Workshop on Energy Efficient Machine Learning and Cognitive Computing (NeurIPS)*, 2019.

Sze, V., Chen, Y., Yang, T., and Emer, J. S. Efficient processing of deep neural networks: A tutorial and survey. *Proc. IEEE*, 105(12):2295–2329, 2017.

Vaswani, A., Shazeer, N. M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *NIPS*, 2017.

Zagoruyko, S. and Komodakis, N. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *ICLR*, 2016.