Perceiver Is All You Need?

Atharv Bhat
New York University
arb881@nyu.edu

Saumyaa Shah
New York University
sns9906@nyu.edu

Sourabh Kumar Bhattacharjee
New York University
skb5275@nyu.edu

Yash Thesia
New York University
yt2188@nyu.edu

Abstract

Transformer-based architectures (Vaswani et al., 2017) have achieved state-of-the-art performance on several modern natural language processing(NLP) tasks. However, due to the quadratic space and time complexity of the attention mechanism, their use for large input sequences remains limited. In recent years, many architectures have proposed approximations of the vanilla attention mechanism which scale linearly with respect to the input size. However, as shown in (Tay et al., 2020a), these *X-formers* also introduce inductive biases, which prevent them from performing well on certain long-range NLP tasks, thus raising questions on generalizability. The *Perceiver* (Jaegle et al., 2021) formulates a transformer-based framework which has been empirically shown to contain minimal inductive biases for long-range vision tasks, with limited assumptions about the input. However, it has not been tested on text data, especially for long-range tasks. We conducted the experiments on Long Range Arena proposed by (Tay et al., 2020a) and observed that *Perceiver* performs well for long context tasks.

1 Introduction

Since the advent of the vanilla Transformer (Vaswani et al., 2017), the field of Natural Language Processing has seen a significant boost in terms of performance in Natural Language Understanding(NLU) and related downstream tasks. The transformer architecture when combined with large corpus pre-training (BERT (Devlin et al., 2018), GPT-2 (Bahdanau et al., 2015)) followed by task specific transfer learning has shown to achieve state-of-the-art results on many NLU tasks. These performances are primarily attributed to the presence of the *Multi-Head Self Attention* module, used to process and attend to the inputs all at once (unlike attention-based Recurrent Neural Networks (Bahdanau et al., 2015)), thereby circumventing the long-range dependency issues suffered by the latter.

While these seminal models perform well on downstream tasks in general, they are also prohibitively expensive to scale to long input sequences due to the quadratic time and memory complexities of the self-attention modules (Vaswani et al., 2017). This restricts their potential applications in domains that require longer input sequence lengths. To tackle this issue, a significant number of so-called *efficient transformers* or *X-formers* (such as Reformers (Kitaev et al., 2020), Performers (Choromanski et al., 2021) etc.) have been introduced in the past few years, which provide better scaling complexities for applications for longer input sequences.

Since the release of the benchmark *Long-Range Arena(LRA)* (Tay et al., 2020a), long-range performance of these *efficient* transformers have been studied on both text and images. The results on LRA showcases that most of these *X-formers* perform well on some of the tasks while showing sub-par performance on others. This suggests that some architectural, efficiency-induced inductive biases may be at play which make inherent assumptions on the input data and prevents these models from generalizing well over a range of tasks.

The *Perceiver* (Jaegle et al., 2021) is an efficient transformer-based model which makes few architectural, as well as input assumptions and is designed to handle high-dimensional inputs, making it suitable for long-range tasks. While this model claims to be input-agnostic, it has not been tested on text data, especially in the long-context setting. So in our work[1] we attempt to test the following question:

- Does the *Perceiver* provide better task-agnostic performance on long-range unified-benchmark text tasks compared to efficient

[1]Codebase: https://github.com/AtharvBhat/PIAYN

transformers(Tay et al., 2020b)? If so, then following from the claim made by (Jaegle et al., 2022), which states that *Perceiver* works well on shorter inputs as well, *Perceiver* can be used as a general Language Model capable of being scaled for longer inputs.

2 Related Work

Efficient Transformers/X-formers: In recent years, many methods have been introduced for dealing with the quadratic cost of full attention. In general, they can be categorized into the following taxonomies (as specified in (Tay et al., 2020b)):

- **Sparse Attention Transformers**: These models *sparsely* activate a small subset of the total parameters thereby improving the parameter-to-floating point operations per second(FLOPs) ratio.

- **Low-Rank/Kernel Transformers**: Low-Rank methods have an underlying assumption of the $N \times N$ full attention matrix having a low-rank structure to project it to a smaller $N \times k$ (where $k \ll N$) dimension. Kernel methods (similar to Low-Rank methods) use mathematical transformations to approximate the full attention matrix. ***Linformer*** (Wang et al., 2020), ***Linear Transformer*** (Katharopoulos et al., 2020), ***Performer*** (Choromanski et al., 2021), ***Transformer-LS*** (Zhu et al., 2021) and ***Synthesizer*** (Tay et al., 2021) use these methods.

- **Fixed/Factorized/Random Patterns**: Models such as ***Sparse Transformer*** (Child et al., 2019) and ***Longformer*** (Beltagy et al., 2020) compute a sparse version of the full attention matrix by restricting the field-of-view to fixed and pre-defined patterns such as local windows and block patterns of pre-set strides.

- **Recurrence Transformers**: Builds on the formulation of block-based fixed pattern methods where the blocks are recursively connected. For instance, ***Transformer-XL*** (Dai et al., 2019) uses a segment-level recurrence which connects multiple segments and blocks.

- **Memory/Downsampling**: These methods reduce computing costs by reducing the resolution of input or intermediate sequences. ***Nystromformer*** (Xiong et al., 2021), ***Perceiver*** (Jaegle et al., 2021) and ***BigBird*** (Zaheer et al., 2020) use these methods.

- **Learnable Patterns**: Builds on the fixed, predetermined pattern methods in which models try to learn these patterns from the data. The main idea is to exploit these fixed patterns but by learning to cluster the inputs for a more optimal view during attention while maintaining efficiency. ***Reformer*** (Kitaev et al., 2020) and ***Sinkhorn*** (Cuturi, 2013) use these methods.

Non-transformer Models: Another contemporary approach to tackle the long-range arena tasks is **state space models(SSM)**, that use special state matrices to capture long-range dependencies(Gu et al., 2021). The **Structured State Space(S4)** sequence model, proposed by (Gu et al., 2022), improves upon the results of previous state-of-the-art methods, as well as resolves the computational impracticality of previous SSMs by re-parameterizing the special state matrix into low-rank and skew-symmetric terms, thus bolstering the theory of exceptional performance of deep SSMs in long-context scenarios.

In this work, our sole focus is on transformer-based architectures.

3 Methodology

3.1 Datasets

To evaluate the performance of *Perceiver* under scenarios involving long-context, we employ three textual datasets, proposed under the Long-Range Arena benchmark proposed by Tay et al. (2020a).

The **Long ListOps** task aims at assessing a natural language model's ability of parsing hierarchically structured data sequences in long-context scenario. The dataset contains arrays of arithmetic operations on single-digit numbers, formatted using prefix notation and structured such that the summary operators **MAX, MEAN, MEDIAN** and **SUM_MOD** are enclosed with delimiters. Each input sequence has a corresponding output between 0-9, thus creating a 10-way classification task. For our experiments, we use a sequence length of `2K`. An example input sequence is as follows:

```
INPUT: [MAX 3 8 [MIN 1 5 ] 0 ]
OUTPUT: 8
```

The **byte-level text classification** task focuses on examining the capability of the model in dealing with **compositionality** i.e. creating words from bytes/characters and advanced phrases from words. To test on lengthy input sequences, a character-level model framework is used. The dataset employed for this task is the **IMDb reviews dataset**, a popular dataset for binary sentiment analysis. It consists of 50,000 reviews of movies, split equally into training and testing set, derived from Internet Movie Database(IMDb). For our experiments, we use sequence length of `4K`.

The **byte-level document matching** task targets assessing the model's capacity to learn a similarity metric between two documents. Compressed representations suitable for retrieval are generated by encoding the long input sequences. We use the **ACL Anthology Network(AAN)** dataset(Bird et al., 2008) to simulate a popular document matching scenario: recognizing a common citation link between two articles. For our experiments, we use a sequence length of `4K` per document, bringing the total sequence length to `8K`.

3.2 Model

The distinguishing feature of ***Perceiver*** is its use of alternating latent self-attention transformer blocks and cross-attention modules to **iteratively attend** to the input sequence. Applying cross attention on the input array brings down the computation complexity to linear from quadratic in terms of input size. This removes the dependency of the depth of the network on input size, allowing us to design deeper networks, which in turn allows us to model more challenging tasks.

To ensure unbiased comparison with the results of models tested in Long-Range Arena, such as (Choromanski et al., 2021) and (Katharopoulos et al., 2020), for our experiments, we use the same self-attention parameters and training configuration as all models implemented in (Xiong et al., 2021). We then proceed to test if *Perceiver*'s minimal inductive bias succeeds in improving upon the results obtained by state-of-the-art transformer models on text-based long-context tasks.

We evaluate the performance of our model using classification accuracy on test set, following the evaluation strategy from previous works.

Model	ListOps	Text	Retrieval
Linear Attention	18.35	64.27	79.34
Linformer	37.25	55.91	79.37
Performer	18.35	64.10	78.62
Perceiver	37.15	63.76	76.25

Table 1: Comparison of baseline results of *Perceiver* to previous models

4 Experiments

For our experimentation, we train our model on the datasets for each of the three LRA tasks, validating performance on the held-out dataset while training. We fine-tune the Perceiver-specific parameters and report the best performance for each task. Our experiments and results are discussed below.

4.1 Baselines

We implement three self-attention based architectures, namely linear transformer(Katharopoulos et al., 2020), Linformer (Wang et al., 2020) and Performer(Choromanski et al., 2021) as our baselines, and compare their performance with our model, which, in addition to self-attention, also employs cross-attention.

4.2 Implementation Details

Model Architecture: Similar to (Xiong et al., 2021), the baselines as well as Perceiver use a transformer depth of 2, an embedding dimension of 64, a latent dimension of 128 and 2 self-attention heads. We vary the number of cross-attention heads, the size of self-attention and cross-attention heads and the number of latents in Perceiver, and observe their effects on the model accuracy, while keeping the architecture size small.

Training parameters: To train the model on the LRA benchmark datasets, we set the training parameters as the following: (i) for text classification, we use a learning rate of 0.0001 with 20K training steps and a warmup of 8K, (ii) for ListOps, we set the learning rate to 0.0001, and train the model for 5K steps, and (iii) for byte-level document matching, we use a learning rate of 0.0001, and train for 30K steps, with a warmup of 800. The feed-forward layers are followed by ReLU activations, with cross-entropy loss and Adam with warmup used for optimization and a batch size of 32.

We use the PyTorch implementation of Perceiver language model, developed by (Wang, 2021). We employ data pre-processing scripts from the Nystromformer code base to generate our train-

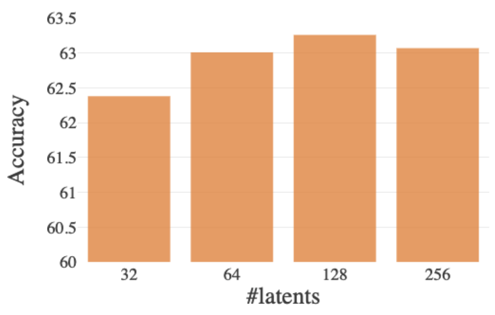

Figure 1: Effect of num_latents on Perceiver for Text Classification

ing/validation/test splits, as well as integrate our model into its configuration and training scripts for solving the LRA tasks. Experiments are run on Google Colab as well as the Nvidia RTX8000 GPU on Greene.

5 Results and Analysis

Table 1 shows the comparison of Perceiver with the self-attention baselines. We observe that Perceiver's classification accuracy closely matches the performance of the baselines for all 3 LRA tasks. It also outperforms some of the methods for certain tasks, such as getting 2x accuracy over Linear Attention and Performer for ListOps, and approximately an 8% increase in accuracy over Linformer in text classification.

We can see that *Perceiver* isn't the best at one particular task. But its performance is on par with current state-of-the-art(SOTA) models at all the tasks. This supports our hypothesis that some models are not good for long context tasks because of their architectural inductive biases, and Perceiver's lack of inherent inductive bias allows it to have adequate performance on all three text-based tasks.

5.1 Parameter Search

We select four cross-attention parameters in Perceiver, namely `cross_heads` i.e. number of heads for cross attention, `cross_dim_head` i.e. number of dimensions per cross attention head, `latent_dim_head` i.e. number of dimensions per self attention head and `num_latents` i.e. the number of latents in the model.

Looking at the model size on varying the cross-attention parameters, we observe that on increasing the number of latents or the number of cross-attention heads(Figures 1,2), the number of parameters doesn't become exceedingly large, with the largest model's size being close to BERT. Despite

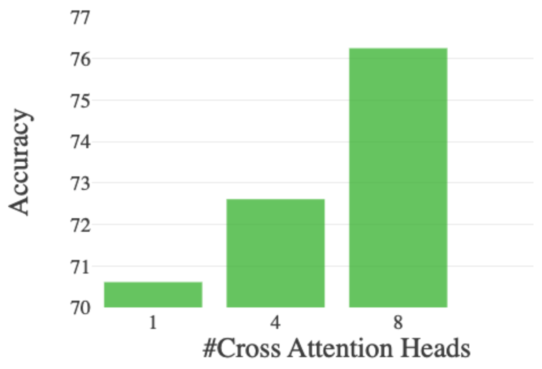

Figure 2: Effect of cross_heads on Perceiver for Retrieval

the lesser parameters, our model is able to capture the long context in the inputs. A similar pattern is observed on increasing the dimensions of the attention heads.

6 Conclusion

In this work, we attempted to test the performance of *Perceiver* on long sequence text tasks such as those given in *Long-Range-Arena*, and observed that while *Perceiver* doesn't achieve SOTA performance in any of the LRA tasks individually, it does however perform almost as well as the best baselines for each of the three tasks we've considered. This task-agnostic performance is indicative of its capability to scale on input lengths due to absence of any assumptions it makes on the input data. Further, we also observe that varying parameters such as number of *cross attention heads* and number of *latents* helps it perform better than the baselines. Hence, we argue that the *Perceiver* MIGHT be all you need in terms of a general language model capable of scaling on longer input sequences.

7 Future Work

In the future, we can change the model parameters and hyper-parameters to analyze their effects on the LRA tasks, as well as improve upon the current test accuracy. Since Perceiver performs well on all LRA tasks, we can train the Perceiver language model with a higher sequence length and understand its performance on various NLU tasks such as long context question answering and document translation, as well as test its performance on other benchmarks such as SCROLLS(Shaham et al., 2022), MuLD(Hudson and Moubayed, 2022) and GLUE(Wang et al., 2018).

Collaboration Statement

All authors have contributed equally to this paper.
Atharv Bhat (arb881): Project idea formation, Literature Review, Huggingface model implementation and experiments for LRA, Proof-reading, iterative Perceiver model.
Saumyaa Shah (sns9906): Project idea formation, Literature Review, Paper Write-up, Baseline implementations and parameter search experiments in Pytorch, Proof-reading.
Sourabh Kumar Bhattacharjee (skb5275): Project idea formation, Literature Review, Implementation of Hugging-face experiments, Pytorch Codebase development/alignment, Write-up, Proof reading.
Yash Thesia (yt2188): Project idea formation, Literature Review, Huggingface Implementation for Document Retrieval and Pytorch implementation of Perceiver for LRA, Write-up, Proof-reading.

Ethical Considerations

Just last year, Google observed that its trillion-parameter Switch Transformer raised several ethical questions regarding the "language" it was learning. As a model is made larger and trained on more and more human-generated natural language data, it gets closer to mimicking human speech and thus, successfully solving several inferential tasks. In the process, it also mimics the inherent biases and toxicities of people such as racism, sexism and homophobia. In our experiments, all our models have less number of parameters, with our largest model having around 1.4 million parameters. So, we believe that our model isn't inherently unethical. To ensure the fairness of our model, we would consider analysing the biases(if any) in the current LRA datasets, as well as future datasets to be used to train the models. In the future, when we extend our experiments to tasks such as document translation or question answering, we need to filter out any biased text from the data before we train our language models.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *CoRR*, abs/2004.05150.

Steven Bird, Robert Dale, Bonnie Dorr, Bryan Gibson, Mark Joseph, Min-Yen Kan, Dongwon Lee, Brett Powley, Dragomir Radev, and Yee Fan Tan. 2008. The ACL Anthology reference corpus: A reference dataset for bibliographic research in computational linguistics. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *URL https://openai.com/blog/sparse-transformers*.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. 2021. Rethinking attention with performers. In *International Conference on Learning Representations*.

Marco Cuturi. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Albert Gu, Karan Goel, and Christopher Re. 2022. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*.

Albert Gu, Isys Johnson, Karan Goel, Khaled Kamal Saab, Tri Dao, Atri Rudra, and Christopher Re. 2021. Combining recurrent, convolutional, and continuous-time models with linear state space layers. In *Advances in Neural Information Processing Systems*.

G Thomas Hudson and Noura Al Moubayed. 2022. Muld: The multitask long document benchmark. *arXiv preprint arXiv:2202.07362*.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier J Henaff, Matthew Botvinick, Andrew Zisserman, Oriol Vinyals, and Joao Carreira.

2022. Perceiver IO: A general architecture for structured inputs & outputs. In *International Conference on Learning Representations*.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. 2021. Perceiver: General perception with iterative attention. In *International Conference on Machine Learning*, pages 4651–4664. PMLR.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and Franccois Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *ICML*.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *International Conference on Learning Representations*.

Uri Shaham, Elad Segal, Maor Ivgi, Avia Efrat, Ori Yoran, Adi Haviv, Ankit Gupta, Wenhan Xiong, Mor Geva, Jonathan Berant, et al. 2022. Scrolls: Standardized comparison over long language sequences. *arXiv preprint arXiv:2201.03533*.

Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. 2021. Synthesizer: Rethinking self-attention for transformer models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10183–10192. PMLR.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2020a. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020b. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Phil Wang. 2021. https://github.com/lucidrains/perceiver-pytorch.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *CoRR*, abs/2006.04768.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. 2021. Nyströmformer: A nyström-based algorithm for approximating self-attention.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33.

Chen Zhu, Wei Ping, Chaowei Xiao, Mohammad Shoeybi, Tom Goldstein, Anima Anandkumar, and Bryan Catanzaro. 2021. Long-short transformer: Efficient transformers for language and vision. In *Advances in Neural Information Processing Systems*.

A Appendix

A.1 Iterative Attention

Model	ListOps	Text
Perceiver	37.15	63.76
Perceiver$_{it}$	37.08	65.14

Table 2: Comparison of baseline results of *Perceiver* to Perceiver with iterative Attention.

We do not see a significant performance difference between *Perceiver* and *Perceiver* with iterative attention. This might be because of a shallower model which is not able to benefit from iterative attention.

A.2 Experiments Using Original Config

Model	ListOps	Text	Retrieval
Perceiver	37.50	65.2	50.4

Table 3: Comparison of baseline results of *Perceiver* to Perceiver with iterative Attention.

Our initial experiments were based on the configuration mentioned in the original Long Range Arena paper(Tay et al., 2020a). But, on exploring their codebase, we found inconsistencies in the parameter values and training settings used in the code and the ones mentioned in the paper. Additionally, when we tried to use their Flax implementation to reproduce their baseline results, we found many unresolved dependencies and problems with the Flax functions used in the code. So, we switched to using PyTorch for our experiments with Perceiver.