ORCHID: FLEXIBLE AND DATA-DEPENDENT CONVOLUTION FOR SEQUENCE MODELING

Mahdi Karami, Ali Ghodsi
Department of Computer Science School of Computer Science
University of Waterloo, ON, Canada
karami1@ualberta.ca

ABSTRACT

In the rapidly evolving landscape of deep learning, the quest for models that balance expressivity with computational efficiency has never been more critical. Orchid is designed to address the quadratic computational complexity of attention models without sacrificing the model's ability to capture long-range dependencies. At the core of Orchid lies the data-adaptive convolution layers, which conditionally adjust their kernels based on input data using a conditioning neural network. This innovative approach enables the model to maintain scalability and efficiency for long sequence lengths. The adaptive nature of data-adaptive convolution kernel combined with the gating operations allows it to offer a highly expressive neural network. We rigorously evaluate Orchid across multiple domains, including language modeling and image classification, to showcase its generality and performance. Our experiments demonstrate that Orchid not only consistently outperforms traditional attention-based architectures in most scenarios but also extends the feasible sequence length beyond the constraints of dense attention layers. This achievement marks a significant milestone in the pursuit of more efficient and scalable deep learning models for sequence modeling.

1 INTRODUCTION

In the realm of modern deep neural networks, attention mechanisms have emerged as a gold standard, pivotal in domains such as natural language processing, image, and audio processing, and even complex fields like biology Vaswani et al. (2017); Dosovitskiy et al. (2020); Dwivedi & Bresson (2020). Despite their strong sequence analysis capabilities, these mechanisms face challenges, especially the computational complexity of Transformers, which scales quadratically with sequence length, creating significant hurdles in long-context tasks Dao et al. (2022); Chen et al. (2021a;b). In response to these computational challenges, the research community has explored alternatives to traditional dense attention layers. Methods like linear attention, sparse, and low-rank approximations of attention have been developed to reduce the computational complexity of attention layers in deep neural networks, enhancing scalability to larger sequences Child et al. (2019); Wang et al. (2020); Kitaev et al. (2020); Zhai et al. (2021); Schlag et al.. However, while these methods significantly reduce computational overhead, they often have lower expressiveness and performance.

Addressing the need for expressive, sub-quadratic, and hardware-efficient mixing operators is a formidable challenge. Recent studies have introduced sub-quadratic sequence mixing with long convolutions or state space models as potential solutions Gu et al. (2021); Romero et al. (2021); Mehta et al. (2022); Wang et al. (2022); Poli et al. (2023); Fu et al. (2023a;b). Orchid marks a significant advancement by offering an expressive, sub-quadratic primitive based on input-adaptive convolution, providing a robust alternative to the traditional Transformer paradigm and laying the foundation for further advancements in efficient modeling, opening new pathways to address the computational challenges in deep learning.

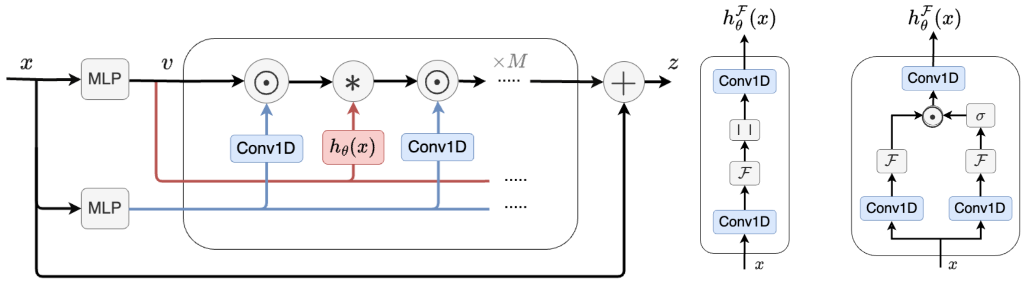

Figure 2.1: Orchid block. $\odot$ and $*$ denote element-wise multiplication and the convolution operator, respectively. The convolutions are implemented in the frequency domain using FFT. On the right two different conditioning networks, introduced in equations (2) and (3) as shift-invariant convolution kernels, are depicted.

2 Preliminaries

Self-Attention Mechanism: Given a length-L sequence of embeddings (of tokens) $x = (x_1, x_2, \dots, x_L)$, self-attention generates a new sequence by computing a weighted sum of these embeddings.[1] It does this by linearly projecting x into three components: queries (Q), keys (K), and values (V), i.e., $Q = xW^Q, \quad K = xW^K, \quad V = xW^V$. Each head of self-attention can be expressed as a dense linear layer as follows:

$$y = \text{SelfAttention}(Q, K, V) = \text{SoftMax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V = A(x)xW^V,$$

where the matrix $A(x)$ is populated with the attention scores between each pair of tokens. This description of the attention layer highlights its notable benefits, including its capability to capture long-range dependencies efficiently, with a sublinear parameter count. The attention mechanism enables direct computation of interactions between any two positions in the input sequence, regardless of their distance, without a corresponding rise in parameter counts. Additionally, the attention layer implements a *data-adaptive* dense linear filter, effectively filtering the input while the filter weights are conditioned by a mapping of the data. However, these merits come at the expense of quadratic computational complexity and memory costs.

This motivates us to develop a scalable and efficient *data-adaptive convolution* mechanism, featuring an adaptive kernel that adjusts based on the input data. The kernel size of this convolution layer is as long as the input sequence length, allowing the capture of long-range dependencies across the input sequence while maintaining high scalability.

Linear Convolution: Discrete-time linear convolution is a fundamental operation in digital signal processing that computes the output as the weighted sum of the finite-length input x with shifted versions of the convolution kernel, h, also known as the impulse response of a linear time-invariant (LTI) system, formally as $\boldsymbol{y}[t] = (\boldsymbol{h} * \boldsymbol{x})[t] \triangleq \sum_{\ell=0}^{L-1} h[t-\ell]x[\ell]$. Circular convolution is defined as $\boldsymbol{y}[t \bmod L] = (\boldsymbol{h} \circledast \boldsymbol{x})[t] \triangleq \sum_{\ell=0}^{L-1} h[t-\ell \mod L]x[\ell]$, which is equivalent to the linear convolution of two sequences when one is padded cyclically.

Fast convolution algorithm: One key advantage of convolution operators is that, according to the *convolution theorem*, they can be performed in the frequency domain, hence can be computed efficiently in $\mathcal{O}(L \log L)$ time using Fast Fourier Transform (FFT) algorithms. Generally speaking, convolution can be performed as $\hat{\boldsymbol{y}} = \mathcal{F}^{-1}(\mathcal{F}(\hat{\boldsymbol{h}}) \odot \mathcal{F}(\hat{\boldsymbol{x}})) = \boldsymbol{T}^{-1}(\boldsymbol{h}_{\mathcal{F}} \odot \boldsymbol{T}\hat{\boldsymbol{x}})$, where $\boldsymbol{T}$ is the DFT matrix, $\mathcal{F}$ denotes the discrete Fourier transformation, $\hat{\boldsymbol{x}} = pad(\boldsymbol{x})$ denotes the zero-padded signal

[1]

3 ORCHID

3.1 DATA-ADAPTIVE CONVOLUTION FILTER

We claim that making the kernel of convolution data dependent, renders the layer more expressive

$$y = h_\theta(x) * x = \mathrm{NN}_\theta(x) * x \tag{1}$$

We call the function $h_\theta(x) = \mathrm{NN}_\theta(x)$, *conditioning network* parameterized by θ. Hence given an input,$\boldsymbol{x}$, this operation defines how each token attends to the entire signal as a weighted sum whose weights are conditioned on the input itself.

In general, a discrete convolution is shift equivariance, that is, ignoring boundary (edge) effects, if the input data is spatially shifted, the output of the model shifts by the same amount. Boundaries doesn't affect this property in circular convolution, hence $\text{shift}_m(\boldsymbol{y}) = \boldsymbol{h} \circledast \text{shift}_m(\boldsymbol{x})$ where $\text{shift}_m(\boldsymbol{x})[t] \triangleq \boldsymbol{x}[t+m]$ (Bronstein et al., 2021). This property ensures that the operation's output is robust to shift of features within the input, thereby enhancing the model's generalization capabilities. This capability (which induces an inductive bias on the model) is at the core of the widespread success of convolution operations Thomas et al.. Therefore it is desirable to design conditioning network in the data-adaptive convolution in (1) to preserve shift equivariance property. It can be shown that to hold this property for convolution operation it is sufficient to design filter kernel to be *shift invariant*. In the following, we present two methods for designing a shift-invariant conditioning network.

I) Suppressing the Phase of Frequency Components: A circular shift of a sequence $\boldsymbol{u}$ corresponds to multiplying its frequency components by a linear phase, *i.e.* $\mathcal{F}\left(\text{shift}_m(\boldsymbol{u})\right)[k] = \boldsymbol{u}_\mathcal{F}[k] \cdot e^{-\frac{i2\pi}{L}km}$ Oppenheim (1999). Suppose $g(\boldsymbol{x})$ is a shift-equivariant function (such as a depthwise `Conv1d()`), such that $g(\text{shift}_m(\boldsymbol{x})) = \text{shift}_m(g(\boldsymbol{x}))$. The frequency components of $g(\boldsymbol{x})$, when spatially shifted, can be expressed as: $\mathcal{F}\left(g(\text{shift}_m(\boldsymbol{x}))\right[k] = \mathcal{F}\left(g(\boldsymbol{x})\right)[k] \cdot e^{-\frac{i2\pi}{L}km}$. Subsequently, by applying the absolute value (or the magnitude of complex numbers) non-linearity to the frequency components, we can achieve shift invariance. Defining $h_\mathcal{F}(\boldsymbol{x}) = \left|\mathcal{F}\left(g(\boldsymbol{x})\right)\right|$, it follows that $h_\mathcal{F}(\text{shift}_m(\boldsymbol{x})) = h_\mathcal{F}(\boldsymbol{x})$, thereby satisfying shift invariance.

In our setup, we define $g(\boldsymbol{x})$ as a 1D depth-wise linear convolution, denoted as $\texttt{Conv1d}(\boldsymbol{x})$, with a short kernel length (typically 3-5) for each feature dimension, which is followed by a short convolution in the frequency domain. Consequently, the conditioning neural network is formulated as

$$h_\theta^\mathcal{F}(\boldsymbol{x}) = \texttt{Conv1d}(\left|\mathcal{F}\left(\texttt{Conv1d}(\boldsymbol{x})\right)\right|) \tag{2}$$

This architecture choice minimizes the number of parameters and reduces the computational burden that the conditioning network introduces to the overall model.

II) Using Cross-Correlation to Achieve Shift Invariance An alternative method to attain shift invariance involves computing the cross-correlation between two versions of a signal. Consider $k(\boldsymbol{x})$ and $q(\boldsymbol{x})$ as two shift-equivariant functions, satisfying: $k(\text{shift}_m(\boldsymbol{x})) = \text{shift}_m(k(\boldsymbol{x}))$ and $q(\text{shift}_m(\boldsymbol{x})) = \text{shift}_m(q(\boldsymbol{x}))$. Define $h(\boldsymbol{x})$ as the cross-correlation of $k(\boldsymbol{x})$ and $q(\boldsymbol{x})$, given by: $h(\boldsymbol{x})[t] = (k(\boldsymbol{x}) \star q(\boldsymbol{x}))[t] \triangleq \sum_{\ell=0}^{L-1} k(\boldsymbol{x})[\ell] \cdot q(\boldsymbol{x})[t+\ell \mod L]$. It can be demonstrated that $h(\boldsymbol{x})$ is shift invariant:

$$h(\text{shift}_m(\boldsymbol{x})) = k(\text{shift}_m(\boldsymbol{x})) \star q(\text{shift}_m(\boldsymbol{x})) = \text{shift}_m(k(\boldsymbol{x})) \star \text{shift}_m(q(\boldsymbol{x})) = k(\boldsymbol{x}) \star q(\boldsymbol{x}) = h(\boldsymbol{x})$$

Moreover, according to the convolution theorem, the cross-correlation can be efficiently computed in the frequency domain as $h_\mathcal{F}(\boldsymbol{x}) = \mathcal{F}\left(k(\boldsymbol{x}) \star q(\boldsymbol{x})\right) = k_\mathcal{F}^*(\boldsymbol{x}) \odot q_\mathcal{F}(\boldsymbol{x})$ where $k_\mathcal{F}^*$ denotes the complex conjugate of $k_\mathcal{F}$

Remark 3.1. By using the same function for both k and q, *i.e.* $k(\boldsymbol{x}) = q(\boldsymbol{x}) = g(\boldsymbol{x})$, we derive $h_\mathcal{F}(\boldsymbol{x}) = |g_\mathcal{F}(\boldsymbol{x})|^2$, implying that the cross-correlation-based approach generalizes approach (I).

In a similar manner, we leverage distinct 1D depth-wise short convolutions for $k(\boldsymbol{x})$ and $q(\boldsymbol{x})$ for both $k(\boldsymbol{x})$ and $q(\boldsymbol{x})$, followed by another convolution post cross-correlation in the frequency domain. As a result, the conditioning neural network is defined as

$$h_\theta^\mathcal{F}(\boldsymbol{x}) = \texttt{Conv1d}\Big(\mathcal{F}^*\left(\texttt{Conv1d}(\boldsymbol{x})\right) \odot \sigma\big(\mathcal{F}\left(\texttt{Conv1d}(\boldsymbol{x})\right)\big)\Big). \tag{3}$$

The two conditioning functions (2), (3) are also illustrated in Figure 2.1. For operations involving long convolutions, we add a fixed (non data-adaptive) term which is implicitly parametrized using a positional embedding of time step (token index in the sequence) and a feed forward networks (FFNs) Romero et al. (2021); Poli et al. (2023) $h(t) = \mathsf{FFN}(\mathsf{PositionalEmbedding}(t)))$. Subsequently, the final convolution kernel is $h = h(t) + h_\theta(\boldsymbol{x})$.

3.2 Orchid Block

In contrast to attention layers, convolution filters perform parameter sharing, meaning they utilize the same kernel weights across different positions within the input sequence. By integrating this data-adaptive convolution approach with element-wise multiplications, it's possible to achieve a location-dependent filtering scheme. Through element-wise multiplication, specific locations within the signal can be emphasized by assigning higher weights before applying the location-invariant convolution. The composition of a cascade of circulant and diagonal matrices has been demonstrated to serve as an efficient approximation for dense linear layers Moczulski et al. (2015); Cheng et al. (2015). The overall architecture of a block, incorporating our data-adaptive convolution block, is illustrated in Figure 2.1.

4 Experiments

Our evaluation of Orchid focuses on three different Transformer-based models to assess its expressivity and generalization capabilities as an alternative to attention layers. Firstly, we conduct a set of experiments on a synthetic task to assess the **in-context learning ability** (Liang et al., 2022; Olsson et al., 2022) and scalability of the proposed model. It involves generating a value from a key given a string of key-value tuples from a random dictionary. For instance, given the input ($[a, \mathit{1}, b, e, \mathit{3}, f]$, b), the model is expected to return e, the value associated with the key b. The results, illustrated in Figure 4.1 and Table 4.3, demonstrate that the Orchid model offers superior expressiveness and outperforms existing long convolution models in the associative recall task. Notably, in challenging scenarios with short sequence lengths of 128 and large vocabulary sizes, Orchid significantly improves the model's accuracy.

Subsequently, we evaluate the performance of the proposed architecture on **language modeling tasks**. Orchid is designed to integrate seamlessly with existing BERT-style language models, such as BERT Devlin et al. (2018), As the results outlined in Table 4.1 show, Orchid-BERT-base achieves 1.0 points in average GLUE score performance compared to the BERT-base on the GLUE benchmark with utilizing 30% fewer parameters. Similarly, Orchid-BERT-large outperforms the performance of BERT-large by 1.0 points with a 25% reduction in parameter counts.

Moreover, we extend our experiments to the Vision Transformer (ViT) architecture (Dosovitskiy et al., 2020) for **image classification tasks**, aiming to evaluate the model's generalizability across diverse domains. Our experiments are conducted on two widely recognized image datasets: CIFAR-10 and ImageNet-1K. The performance outcomes, as highlighted in Table 4.2 for both the CIFAR-10 and ImageNet-1K datasets, demonstrate that Orchid significantly outperforms baseline ViT-style models on both datasets. These results affirm the adaptability and effectiveness of the Orchid architecture beyond the realm of language modeling, showcasing its potential advantages in image processing tasks.

Table 4.1: Average GLUE Score of BERT-base and BERT-large (Devlin et al., 2018) in comparison to Orchid-BERT-base and Orchid-BERT-base, and M2-BERT-base and M2-BERT-large Dao et al. (2022). Baseline results are drawn from (Fu et al., 2023a).

Model (size)	**GLUE Score**	**∆ Params**	**∆ GLUE Score**
BERT-base (110M)	79.6	-	-
M2-BERT-base (80M)	79.9	-27.3%	+0.3
Orchid-BERT-base (77M)	**80.6**	**-30.0%**	+1.0
BERT-large (340M)	82.1	-	-
M2-BERT-large (260M)	82.2	-23.6%	+0.1
Orchid-BERT-large (254M)	**82.7**	**-25.3%**	+0.6

Table 4.2: Performance comparison of Orchid with ViT-based models on ImageNet-1k and CIFAR-10 dataset. Baseline results are drawn from (Fu et al., 2023a).

Model (size)	Top-1 (%)	Top-5 (%)
ImageNet-1k		
ViT-b (87M)	78.5	93.6
ViT-b + Monarch (33M)	78.9	94.2
Hyena-ViT-b (88M)	78.5	93.6
M2-ViT-b (45M)	79.5	94.5
Orchid-ViT-b (48M)	**80.2**	**94.9**

Model (size)	Top-1 (%)
CIFAR-10	
ViT (1.2M)	78.6
ViT + Monarch (607K)	79.0
Hyena-ViT (1.3M)	80.6
M2-ViT (741K)	80.8
Orchid-ViT (836K)	**84.3**

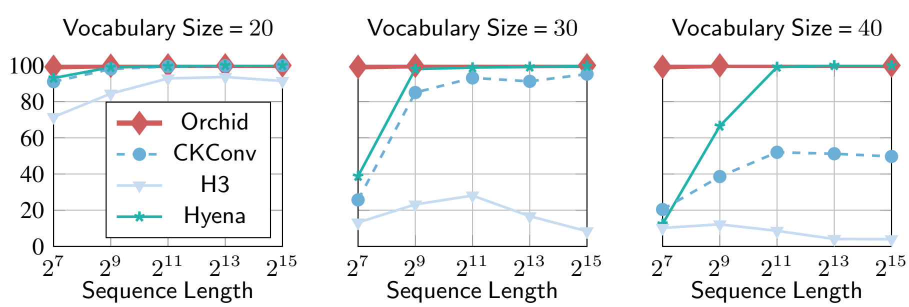

Figure 4.1: Performance of the associative recall task across different long implicit convolution models on various sequence lengths and vocabulary sizes (number of possible token values).

Table 4.3: This table shows the performance of in-context learning on the associative recall task with a vocabulary size of 20 and different sequence lengths. The results for the baseline models are drawn from Poli et al. (2023); Fu et al. (2023a). The symbol ✘ indicates that the Transformer model failed to complete the task within a week or the model d oes not fit in memory.

Model	128	512	2K	8K	32K	128K
Transformer	100	100	100	100	✘	✘
Monarch-Mixer	-	98.7	99.4	99.4	99.4	99.4
Hyena	93	99	99.6	100	100	-
Orchid	99.2	99.8	100	100	100	100

References

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Beidi Chen, Tri Dao, Kaizhao Liang, Jiaming Yang, Zhao Song, Atri Rudra, and Christopher Ré. Pixelated butterfly: Simple and efficient sparse training for neural network models. 2021a.

Beidi Chen, Tri Dao, Eric Winsor, Zhao Song, Atri Rudra, and Christopher Ré. Scatterbrain: Unifying sparse and low-rank attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021b.

Yu Cheng, Felix X Yu, Rogerio S Feris, Sanjiv Kumar, Alok Choudhary, and Shi-Fu Chang. An exploration of parameter redundancy in deep networks with circulant projections. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2857–2865, 2015.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Tri Dao, Beidi Chen, Nimit S Sohoni, Arjun Desai, Michael Poli, Jessica Grogan, Alexander Liu, Aniruddh Rao, Atri Rudra, and Christopher Ré. Monarch: Expressive structured matrices for efficient and accurate training. In *International Conference on Machine Learning*. PMLR, 2022.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

Daniel Y Fu, Simran Arora, Jessica Grogan, Isys Johnson, Sabri Eyuboglu, Armin W Thomas, Benjamin Spector, Michael Poli, Atri Rudra, and Christopher Ré. Monarch mixer: A simple sub-quadratic gemm-based architecture. *arXiv preprint arXiv:2310.12109*, 2023a.

Daniel Y Fu, Tri Dao, Khaled K Saab, Armin W Thomas, Atri Rudra, and Christopher Ré. Hungry hungry hippos: Towards language modeling with state space models. *International Conference on Learning Representations*, 2023b.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.

Harsh Mehta, Ankit Gupta, Ashok Cutkosky, and Behnam Neyshabur. Long range language modeling via gated state spaces. *arXiv preprint arXiv:2206.13947*, 2022.

Marcin Moczulski, Misha Denil, Jeremy Appleyard, and Nando de Freitas. Acdc: A structured efficient linear layer. *arXiv preprint arXiv:1511.05946*, 2015.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Alan V Oppenheim. *Discrete-time signal processing*. Pearson Education India, 1999.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Hyena hierarchy: Towards larger convolutional language models. *International Conference on Machine Learning*, 2023.

David W Romero, Anna Kuzina, Erik J Bekkers, Jakub Mikolaj Tomczak, and Mark Hoogendoorn. Ckconv: Continuous kernel convolution for sequential data. In *International Conference on Learning Representations*, 2021.

I Schlag, K Irie, and J Schmidhuber. Linear transformers are secretly fast weight programmers. icml 2021. *Preprint*.

N Thomas, T Smidt, S Kearnes, L Yang, L Li, K Kohlhoff, and P Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. arxiv 2018. *arXiv preprint arXiv:1802.08219*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. volume 30, 2017.

Junxiong Wang, Jing Nathan Yan, Albert Gu, and Alexander M Rush. Pretraining without attention. *arXiv preprint arXiv:2212.10544*, 2022.

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

Shuangfei Zhai, Walter Talbott, Nitish Srivastava, Chen Huang, Hanlin Goh, Ruixiang Zhang, and Josh Susskind. An attention free transformer. *arXiv preprint arXiv:2105.14103*, 2021.