STrXL: Approximating Permutation Invariance/Equivariance to Model Arbitrary Cardinality Sets

Kendra Givens, David Ludwig II, Joshua L. Phillips
Department of Computer Science
Middle Tennessee State University
Murfreesboro, TN 37132
Klg6z@mtmail.mtsu.edu, Dwl2x@mtmail.mtsu.edu, Joshua.Phillips@mtsu.edu

Abstract

Current deep-learning techniques for processing sets are limited to a fixed cardinality, causing a steep increase in computational complexity when the set is large. To address this, we have taken techniques used to model long-term dependencies from natural language processing and combined them with the permutation equivariant architecture, Set Transformer (STr). The result is Set Transformer XL (STrXL), a novel deep learning model capable of extending to sets of arbitrary cardinality given fixed computing resources. STrXL's extension capability lies in its recurrent architecture. Rather than processing the entire set at once, STrXL processes only a portion of the set at a time and uses a memory mechanism to provide additional input from the past. STrXL is particularly applicable to processing sets of high-throughput sequencing (HTS) samples of DNA sequences as their set sizes can range into hundreds of thousands. When tasked with classifying HTS prairie soil samples and MNIST digits, results show that STrXL exhibits an expected memory size-accuracy trade-off that scales proportionally with the complexity of downstream tasks, but, unlike STr, is capable of generalizing to sets of arbitrary cardinality.

Deep learning excels at tasks utilizing ordered data such as text generation, image classification, and voice recognition. For example, many languages have a temporal structure, and words depend on their predecessors; while images have a positional structure, each pixel makes sense only in the context of its neighbors. In contrast, tasks such as multiple instance learning Lee et al. [2019], shape identification , and future prediction of molecule states Zhang et al. [2021] all take unordered sets as input.

The inherent lack of spatial relationships in sets renders most deep-learning models inadequate. Their inadequacy stems not from the fact that their order-dependent architectures are entirely incapable of processing sets, but because ensuring absolute robustness to permutation would require training on all permutations of the elements. The computational complexity of $O(n!)$ for a set of size n would be an impractical ask on modern hardware, making the need for a naturally permutation invariant model more apparent.

Zaheer et al. [2018] responded to this necessity by proposing Deep Sets, an efficient, naturally permutation invariant deep learning architecture. Rather than training on all permutations, the model passes each element through a neural network individually and then aggregates the outputs through permutation equivariant pooling operations. Although it is a universal set function approximator, its independent processing hinders higher-order element-wise interactions from being established.

STr builds on the work of Deep Set's pooling operations, but introduces the Transformer block Vaswani et al. [2017] with removed positional encodings to learn the complexity between set element interactions Lee et al. [2019]. Repeated stacks of these permutation equivariant blocks allow a hierarchical representation of what set elements are most influential to emerge. The time complexity of the Transformer's Multi-Head Self Attention (MHA) is $O(n^2)$, but can be reduced to $O(nm)$ with the use of Induced Set Attention Blocks (ISABs). In contrast, Mini-Batch Consistent Models Andreis et al. [2021] represent a newer form of permutation equivariant models that use slot attention to process sets of arbitrary cardinalities. However, their method of slot encoding prevents contextualized item embeddings and restricts the embedding to the full set. Due to this limitation, we have based our architecture on STr's foundation.

While STr is effective at modeling complex relationships in sets, it is still subject to a common problem in deep learning: modeling long-term dependencies. For HTS samples, whose sets range into the hundreds of thousands, even with ISABs, STrs become computationally intractable. For insight on how to better model the long-term dependencies required in HTS samples and other larger sets, we have looked to Natural Language Processing (NLP), a field that has arguably wrestled the most with this problem in long text corpora. Many NLP solutions to this rely on sparsity to reduce the computational cost Jaszczur et al. [2021], Zaheer et al. [2021], similar to STr's ISAB. However, it is not possible to extend to truly arbitrary lengths with a sparse model as it still requires the full input, so we are led instead to the possibility of using Transformer XL's (TrXL) Dai et al. [2019] memory-based, sliding window technique instead.

One important real-world application of such models is the analysis of HTS samples, but to our knowledge, deep learning models for processing HTS samples have yet

to be explored in the literature. Current algorithmic techniques for modeling HTS samples first perform heavy preprocessing Hao et al. [2022], and the few deep learning models that do process HTS samples only do so one sequence at a time Ji et al. [2021], Alharbi and Rashid [2022], Wang et al. [2023]. A deep learning model capable of processing HTS samples would allow us to directly capture interactions between the raw sequences, advancing our understanding of how microorganisms influence one another when it comes to tasks such as gene expression, disease-causing mutations, and microorganism impacts when compared to current techniques.

Our contributions are as follows: (1) a novel architecture capable of extending to sets of arbitrary cardinality, (2) a novel learned compression mechanism for more efficient memory utilization, and (3) the first attempt to process raw HTS samples using deep learning.

Background

Set Transformer

STr was developed to allow for more complex mappings of sets by capturing interactions between respective elements Lee et al. [2019]. STr preserves the underlying encoder-decoder structure of the vanilla Transformer Vaswani et al. [2017] but modifies it to be permutation equivariant. The encoder consists of multi-head attention blocks (MAB) as defined by Vaswani, but without positional encodings. The MAB is formally defined as:

$$\begin{aligned} \text{MAB}(X, Y) &= \text{LN}(H + \text{rFF}(H)), \\ \text{where H} &= \text{LN}(X + \text{MHA}(X, Y, Y)) \end{aligned} \tag{1}$$

Here $X \in \mathbb{R}^{n \times d}$ represents the set of n d-dimensional vectors, LN is layer normalization proposed by Ba et al. [2016], MHA is multi-head attention a described in Vaswani et al. [2017], and rFF is a row-wise feed-forward layer. Since STr uses self-attention for the sets, they additionally formally define a Set Attention Block (SAB) which is an MAB with the same inputs:

$$\text{SAB}(X) = \text{MAB}(X, X) \tag{2}$$

Rather than using a fixed pooling mechanism such as mean, max, or sum, STr's decoder portion is modified to introduce a learned pooling operation, Pooling by MHA (PMA) as shown below in equation 3. PMA aggregates the features with k learnable seed vectors which can then be used for classification, regression, or set prediction.

$$\text{PMA}_k(Z) = \text{MAB}(S, \text{rFF}(Z)) \in \mathbb{R}^{k \times d} \tag{3}$$

Here $Z \in \mathbb{R}^{n \times d}$ is the set of n d-dimensional feature vectors to be pooled and $S \in \mathbb{R}^{k \times d}$ is the set of k learnable seed vectors.

Induced attention Lee et al. [2019] is a modification introduced by STr to reduce computational/space complexity and allow for the handling of larger set cardinalities. Rather than performing MHA between every element of the set, induced attention projects the input set onto m learned inducing points to create an intermediate representation and then re-projects that representation back to its original dimensionality by attending to the original set. This transformation allows for a global encoding of important features of the set while still allowing every element to interact with every other element and reduces the computational complexity of MHA from $O(n^2)$ to $O(nm)$. However, neither the vanilla STr nor its variant with induced attention can scale to truly arbitrary cardinalities as they both depend on taking the full set as input.

Transformer XL

TrXL was introduced to better model long-term dependencies in NLP without increasing computational complexity Dai et al. [2019]. The model is described in Algorithm 1. TrXL breaks the input text into segments and consecutively passes each through a series of Transformer decoder blocks. Each decoder block consists of Relative MHA, two layer-norms, and a feed-forward neural network. To prevent the context fragmentation problem that appears when treating each segment independently Al-Rfou et al. [2018], TrXL introduces a segment-level recurrent memory into the architecture.

Each layer caches its past activations to allow the current segment to attend not only to itself, but also to past segments. To prevent the current input from influencing the previous layers, the gradients for each layer are stopped from propagating backward through the memory. Once the memory is full, the oldest hidden states are discarded.

Algorithm 1 Transformer XL

$\mathbf{mems}_0 \leftarrow 0$
$\mathbf{segments} \leftarrow \text{Segment}(\mathbf{x})$
for segment t = 1, 2, ..., T **do**
  $\mathbf{h}_t^{(1)} \leftarrow \text{Embed}(\mathbf{segments}_t)$
  for layer l = 1, 2, ..., L **do**
    $\mathbf{m}_t \leftarrow \text{concat}(\text{StopGrad}(\mathbf{mems}^{(l)}), \mathbf{h}_t^{(l)})$
    $\mathbf{attn} \leftarrow \text{MHA}^{(l)}(\mathbf{h}_t^{(l)}, \mathbf{m}_t, \mathbf{m}_t)$
    $\mathbf{norm} \leftarrow \text{LN}(\mathbf{attn}^{(l)} + \mathbf{h}_t^{(l)})$
    $\mathbf{mems}_{t+1}^{(l)} \leftarrow \text{concat}(\mathbf{mems}_t^{(l)}, \mathbf{h}_t^{(l)})\,[-\text{mem\_len} :]$
    $\mathbf{h}_t^{l+1} \leftarrow \text{LN}(\text{rFF}^{(l)}(\mathbf{norm}^{(l)}) + \mathbf{norm}^{(l)})$
  end for
end for

Absolute positional encodings no longer work with a recurrent transformer because the current and past segments would have the same position. To solve this, TrXL introduces relative positional encodings which provide information on how far away a token is relative to another rather than where it is globally. However, these positional encodings, prevent TrXL from being capable of permutation invariant processing.

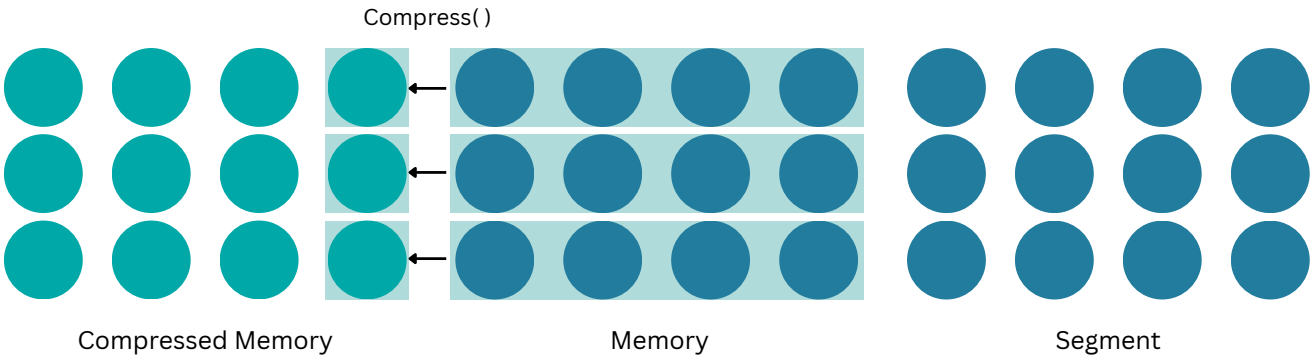

Figure 1: Illustration of memory in CTr. Segments are transferred into memory. When the memory is full, rather than discarding the oldest, CTr compresses them via a pooling operation and stores them in a long-term memory cache.

Compressive Transformer

Compressive Transformer (CTr) builds on TrXL's architecture, but modifies the memory mechanism to compress the excess memory rather than discarding it entirely Rae et al. [2019]. After the current segment is processed, it moves into short-term, uncompressed memory (see Figure 1). Once the short-term memory is full, the older memories are compressed into long-term memory through compression functions such as max/mean pooling, 1D convolution, dilated convolution, and most-used. This method allows the current segment to have access to a much larger temporal range than in TrXL.

Additionally, CTr introduces two auxiliary losses: auto-encoding loss and attention reconstruction loss. The auto-encoding loss tries to recreate the original uncompressed memories from the compressed memories. However, the auto-encoding loss does not perform as well as the auto-reconstruction loss. The auto-reconstruction loss attempts to reconstruct the MHA between the uncompressed memory and current segment from the compressed memory and current segment. Because the second loss is a lossy objective, it is hypothesized that this allows the model to discard what is not needed, whereas the first forces the model to remember even potentially useless information. In both cases, the gradients are only used for optimizing the compression function and do not propagate into the main network.

However, similar to TrXL, there is a dependence on the order preventing it from being a permutation invariant architecture.

DNABERT

DNABERT is a method that uses stacks of transformer blocks to produce contextualized embeddings of DNA sequences Ji et al. [2021]. Each sequence is split into overlapping kmers and associated with a class token. The model is pretrained on two objectives: 15 percent of the sequence is masked out and the model is forced to recreate the missing part, and the consecutive sequence prediction task. The class token representation is thereby conditioned and used as the resulting embedding for a complete DNA sequence. The original model is pretrained on sequence reads of 500 bases in length from the human genome and fine-tuning of the pretrained models may be performed on a variety of tasks related to human health and genetics.

DNABERT was integrated into a recently released online framework for DNA sequence analysis, DeepBIO Wang et al. [2023] and was demonstrated to be the current state-of-the-art deep learning model for DNA sequence analysis. However, it is currently limited to processing single DNA sequences and is not trained on appropriate microorganism DNA segments. Both of these issues currently limit DNABERT's applicability to HTS samples.

Methods

MNIST Point Clouds Dataset

MNIST point cloud classification is a common benchmark to validate a set model's performance. The task is created by turning MNIST handwritten digits into point cloud representations Li Deng [2012] as seen in Figure 2. The task is to correctly classify the number.

We generate our sets of MNIST point clouds by uniformly selecting random pixels sampled from the original handwritten digit. For each pixel selected, we generate a continuous 2d point from a Gaussian distribution. The result is a point cloud that represents the original 2d handwritten digit.

We use this set to validate the performance of STrXL on a simple benchmark classification task.

HTS Samples Dataset

Our HTS dataset contains 205 prairie soil samples collected in Nachusa, Illinois. Each sample contains 150-length sequences targeting the 16s ribosomal RNA (rRNA) region of the bacteria present. These samples are raw reads off the sequencer without any downstream processing or decontamination.

We set up a sample identification task where each sample becomes an individual class. Given a sample, the model must correctly identify the correct class out of 205 possible classes. We chose a classification task as it does not require

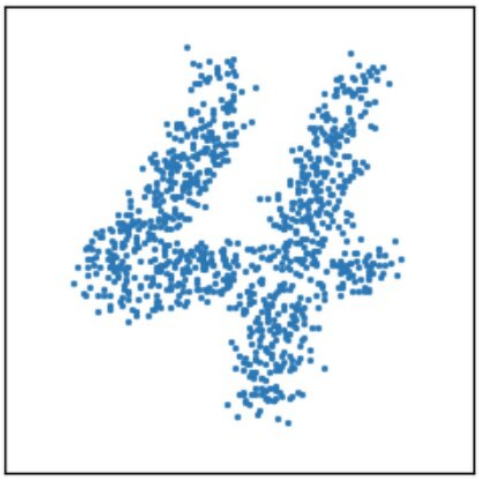

Figure 2: Illustration of MNIST point cloud of a 4.

any additional biological metadata to validate our model's performance on HTS samples.

Since we are performing sample classification, we cannot leave out entire samples for validation/testing so we set aside 20% of the sequences in each sample for validation.

DNA Sequence Embeddings

To make the HTS samples compatible with a deep learning framework, each sequence must first be embedded into a vector representation. To create these embeddings, we use the DNABERT architecture as it creates high-quality contextualized embeddings Ji et al. [2021]. As their model was trained on human genome data, we pretrain it on our dataset. As our dataset consists of fixed-length sequences, we no longer require padding tokens or varying the length of the sequence during interference/training. We also replace the absolute positional encodings with relative encodings as relative is more analogous to sequence alignment. For example, when comparing two sequences, if one were to be shifted over by one nucleotide, absolute encodings would likely result in a very different embedding, while relatively encoded embeddings would only be changed slightly. Lastly, we use pre-layer normalization to stabilize learning Xiong et al. [2020].

For the specific architecture, we use 64-dimensional embeddings with 8 pre-layer norm Transformer blocks each containing 8 MHA heads. The training procedure and learning rate schedule are the same as the original DNABERT paper.

STrXL

The set of embedded sequences is split into segments. Each segment is passed through a series of Transformer blocks which we have modified to use permutation equivariant MABs rather than position dependent relative MABs. The activations in each layer $\mathbf{h}_t^{(l)}$ are cached to be used when processing the next time-step $\mathbf{h}_{t+1}^{(l)}$. We modify each MAB to use a pre rather than post layer norm Xiong et al. [2020] and take in both the current segment $\mathbf{h}_t^{(l)}$ and the concatenation of the memory with $\mathbf{h}_t^{(l)}$ to produce $\mathbf{m}_t$ (see 4). The segment is then sent through PMA to aggregate its features. Depending on the downstream task, either all the segments can be pooled together or just the last one. The architecture for STrXL is shown in Figure 3.

$$\begin{aligned} MAB(X,Y) &= \text{rFF}(\text{LN}_3(H+X)) \\ \text{where } H &= \text{MHA}(\text{LN}_1(X), \text{LN}_2(Y), \text{LN}_2(Y)) \end{aligned} \tag{4}$$

Here X represents the current segment being passed in as the queries, while Y is the concatenation of the segment and memory to be used as the keys and values.

Compressive Modification

We use the compressive memory modification to compare to our TrXL baseline. As with STrXL, we replace their Relative MHA with permutation equivariant MABs. We swap their compression functions with PMA to preserve permutation equivariance in the case of convolution and to encode more information through learned vectors rather than fixed in the case of max/mean/most-used pooling.

Training & Evaluation

MNIST Point Cloud

STrXL uses 3 different memory sizes: 250, 500, and 750 with a segment size of 250 embedded points. We train 5 independent models for 30 epochs using these 3 memory sizes. Additionally, we train our control STr. Both STrXL and STr are trained using set sizes of 1000, with embedding point dimensions of 32, a batch size of 32 with an 80/20 train/validation split of the 60,000 digits in MNIST, and 2 Transformer blocks with 4 attention heads each.

HTS Samples

For this task, STrXL is again trained with 3 different memory sizes: 250, 500, 750. The memory sizes are all trained on the same segment size of 250 embedded sequences.

CTr was trained with a memory size of 250 and a compressed memory size of 250 making it comparable to STrXL's 500 memory. Like STrXL, it uses a segment size of 250. Initially, CTr was trained with the auxiliary attention reconstruction loss, however in 4/5 runs (Data not shown) unstable learning behavior took place. A learning rate scheduler is the typical fix for this Vaswani et al. [2017] in transformers, however, because our other models were trained with a fixed rate, we decided to remove the auxiliary loss and only keep the compressive memory operation.

We trained 5 independent models for each memory size and architecture described above as well as our control STr. Each model was trained for 60,000 steps using the Adam optimizer with a learning rate of $1e-3$, training batch size of 20, validation batch size of 5, set sizes of 1000 with 64 dimensional embedded sequences, and 8 pre-layer norm Transformer blocks with 8 attention heads each.

For both MNIST point cloud classification and HTS sample identification, we pool only the last segment. Additionally, we choose their respective hyperparameters to ensure

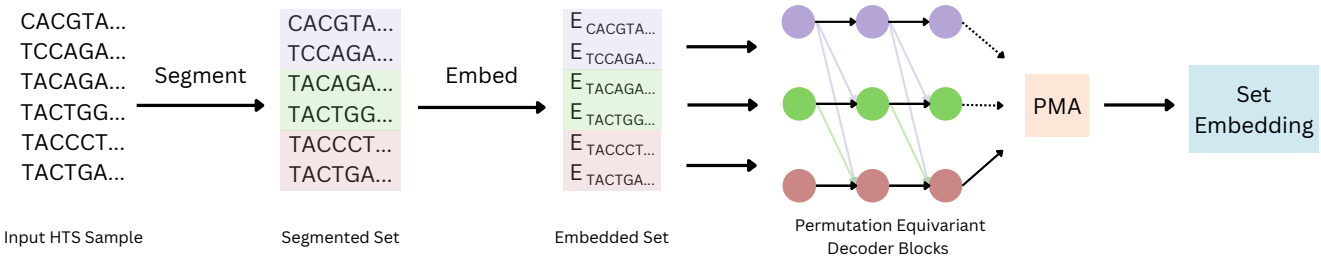

Figure 3: Illustration of STrXL for HTS samples. In this example, the set is first split into segments containing 2 sequences each. Each segment is then embedded and passed through 3 Transformer blocks. The outputs of each segment are then pooled producing a set embedding.

stable learning behavior and keep them constant for each model to allow fair comparisons.

To evaluate our models on both tasks, we calculate the mean and 95% confidence interval (CI) using the last point of validation accuracy in all 5 runs for each model.

Results

MNIST Point Clouds Classification

In this task, the model must correctly classify the digit represented by a cloud of points. The results for each architecture are shown in Table 1:

Model	Mean	95% CI
StrXL (250)	0.963	0.007
StrXL (500)	**0.966**	**0.004**
StrXL (750)	0.966	0.004
Str	0.959	0.007

Table 1: The resulting mean accuracy with 95% confidence interval per model for the MNIST point cloud classification task.

For this particular problem, although StrXL (500) and StrXL (750) achieved the same accuracy, we bold StrXL (500) as the best performing due to its additional benefit of being more efficient through a smaller segment size. Additionally, because all the models accuracy is similar, we hypothesize that the MNIST point cloud classification task is simple enough that the expected memory size-accuracy trade-off does not have an effect.

HTS Sample Identification

In this task, the model must correctly identify the correct HTS sample its input set originated from. The results for classifying HTS samples are shown in Table 2:

With a memory size of 750, STrXL outperforms STr. Since both models have access to the full set, the expected results for each should have been close to the same value. However, we hypothesize that because the set is randomly shuffled, each segment drawn from it is representative of an underlying distribution of the full set. The implication is that

Model	Mean	95% CI
StrXL (250)	0.560	0.100
StrXL (500)	0.704	0.088
StrXL (750)	**0.818**	**0.054**
Compressive (500)	0.656	0.076
Str	0.752	0.075

Table 2: The resulting mean accuracy with 95% confidence interval per model for the HTS sample identification task.

theoretically, a permutation invariant model could learn with no recurrence because each segment can stand alone as its representative distribution. This could not be said in NLP where each segment is not a smaller distribution, but part of a larger whole due to its dependence on order. We think this compacted set representation is used to gate the future segments, allowing STrXL to more efficiently determine what activations to propagate forward. Since STr is comparing everything to everything through each Transformer block, it is harder to filter the most useful information out.

The memory size-accuracy trade-off is much more clear in this task than in MNIST point cloud classification. We hypothesize that the trade-off scales proportionally to the complexity of the downstream task. In this case, HTS samples encode a lot more information than digits of point clouds so the accuracy between varying memory sizes begins to differentiate.

Discussion

We have presented STrXL, a novel architecture capable of extending to sets of arbitrary cardinality while maintaining fixed computing resources. As expected, the accuracy is proportional to the size of memory used but the trade-off scales proportionality to the complexity of the downstream task. This means that in domains where the task is simpler, STrXL can provide the same accuracy with fewer computing resources. Alternatively, for a complex task, the memory size could be increased allowing for a tunable accuracy with respect to an individual's computing resources.

We have also introduced a novel learned compression

mechanism to encode more information on how to compress the set. With more exploration into the model, we think that this could eventually relieve the need for an auxiliary loss in CTr since the model will have already learned a reasonable compression procedure.

Additionally, to our knowledge, we are the first to process raw HTS samples using deep-learning. This allows many biological tasks to now have the full complexity of higher-order relationships and interactions within them.

One limitation of STrXL is that it can only ever be a permutation invariant approximation as breaking the set up does introduce an order-dependence on which segment is processed first. However, we believe any order introduced is negligible as STrXL performs almost to the standard of STr and, in the case of a 750 memory size, is better.

Another limitation is that the context fragmentation problem Al-Rfou et al. [2018] never fully disappears even with the recurrence, however, the segment-level distribution representation and recurrence do minimize it significantly.

Further work will include stabilizing the training of CTr with learning rate schedules, exploring ways to improve the learned compression, testing in-depth biology tasks such as identifying microbes present, learning how microbes impact the community, finding mutations within a sequence, comparing samples from different seasons, classifying the amount of Nitrogen in the soil, characterizing infections growth rate, etc.

Acknowledgements

We are grateful to Wesley Swingley (Northern Illinois University), Meghan Midgley (Morton Arboretum), and Nicholas Barber (San Diego State University) for providing the DNA sequences used in this research.

Kendra Givens and Joshua L. Phillips were supported by the National Science Foundation under REU award 1757493. David Ludwig II and Joshua L. Phillips were supported by the National Science Foundation under DEB award 1933925.

References

Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. Character-Level Language Modeling with Deeper Self-Attention, December 2018. URL http://arxiv.org/abs/1808.04444. arXiv:1808.04444 [cs, stat].

Wardah S. Alharbi and Mamoon Rashid. A review of deep learning applications in human genomics using next-generation sequencing data. *Human Genomics*, 16(1): 26, July 2022. ISSN 1479-7364. doi: 10.1186/s40246-022-00396-x. URL https://doi.org/10.1186/s40246-022-00396-x.

Bruno Andreis, Jeffrey Willette, Juho Lee, and Sung Ju Hwang. Mini-Batch Consistent Slot Set Encoder for Scalable Set Encoding, October 2021. URL http://arxiv.org/abs/2103.01615. arXiv:2103.01615 [cs].

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization, July 2016. URL http://arxiv.org/abs/1607.06450. arXiv:1607.06450 [cs, stat].

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context, June 2019. URL http://arxiv.org/abs/1901.02860. arXiv: 1901.02860.

Zhiqiang Hao, Xiaojuan Liang, and Guanglin Li. Quality Control and Preprocessing of Sequencing Reads. *BIO-PROTOCOL*, 12(13), 2022. ISSN 2331-8325. doi: 10.21769/BioProtoc.4454. URL https://bio-protocol.org/e4454.

Sebastian Jaszczur, Aakanksha Chowdhery, Afroz Mohiuddin, Łukasz Kaiser, Wojciech Gajewski, Henryk Michalewski, and Jonni Kanerva. Sparse is Enough in Scaling Transformers, November 2021. URL http://arxiv.org/abs/2111.12763. arXiv:2111.12763 [cs].

Yanrong Ji, Zhihan Zhou, Han Liu, and Ramana V Davuluri. DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome. *Bioinformatics*, 37(15):2112–2120, August 2021. ISSN 1367-4803. doi: 10.1093/bioinformatics/btab083. URL https://doi.org/10.1093/bioinformatics/btab083.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks, May 2019. URL http://arxiv.org/abs/1810.00825. arXiv: 1810.00825.

Li Deng. The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. *IEEE Signal Processing Magazine*, 29(6):141–142, November 2012. ISSN 1053-5888. doi: 10.1109/MSP.2012.2211477. URL http://ieeexplore.ieee.org/document/6296535/.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. Compressive Transformers for Long-Range Sequence Modelling, November 2019. URL http://arxiv.org/abs/1911.05507. arXiv:1911.05507 [cs, stat].

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, December 2017. URL http://arxiv.org/abs/1706.03762. arXiv:1706.03762 [cs].

Ruheng Wang, Yi Jiang, Junru Jin, Chenglin Yin, Haoqing Yu, Fengsheng Wang, Jiuxin Feng, Ran Su, Kenta Nakai, Quan Zou, and Leyi Wei. DeepBIO: an automated and interpretable deep-learning platform for high-throughput biological sequence prediction, functional annotation and visualization analysis. *Nucleic Acids Research*, 51(7):3017–3029, April 2023. ISSN 0305-1048, 1362-4962. doi: 10.1093/nar/

gkad055. URL https://academic.oup.com/nar/article/51/7/3017/7041952.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On Layer Normalization in the Transformer Architecture, June 2020. URL http://arxiv.org/abs/2002.04745. arXiv:2002.04745 [cs, stat].

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep Sets, April 2018. URL http://arxiv.org/abs/1703.06114. arXiv:1703.06114 [cs, stat].

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big Bird: Transformers for Longer Sequences, January 2021. URL http://arxiv.org/abs/2007.14062. arXiv:2007.14062 [cs, stat].

David W. Zhang, Gertjan J. Burghouts, and Cees G. M. Snoek. Set Prediction without Imposing Structure as Conditional Density Estimation, February 2021. URL http://arxiv.org/abs/2010.04109. arXiv:2010.04109 [cs, stat].