Supplementary Information: Large Scale Chemical Language Representations Capture Structure and Properties

A Model and Methods

A.1 MoLFormer Model and Pre-Training Details

In this section we include additional details and insights of MoLFormer pretraining.

A.1.1 Optimizer

For optimization we used the Fused Lamb optimizer from[71] as implemented via APEX due to Lamb's superior performance in several aspects of training. For example, learning rate warm ups were found to be unnecessary and training was found to be robust when large batch sizes were used. All other optimizers were unable to maintain their stability without large amounts of modification any time a training configuration needed to be changed.

A.1.2 Linear Attention

Preliminary experiments showed an acceptable balance between computation speed and minimal performance deficit when compared to the FAVOR[31] feature map. Generalized Features are a simplification of the feature map in FAVOR[31]. The feature map size we settled on is 32.

A.1.3 Rotary versus Absolute position embeddings

We show in Figure 5 that MoLFormer with linear attention and rotary embeddings has a better validation loss than its absolute position counterpart. This observation lead us to adopt MoLFormer with linear attention and rotary embedding throughout the paper.

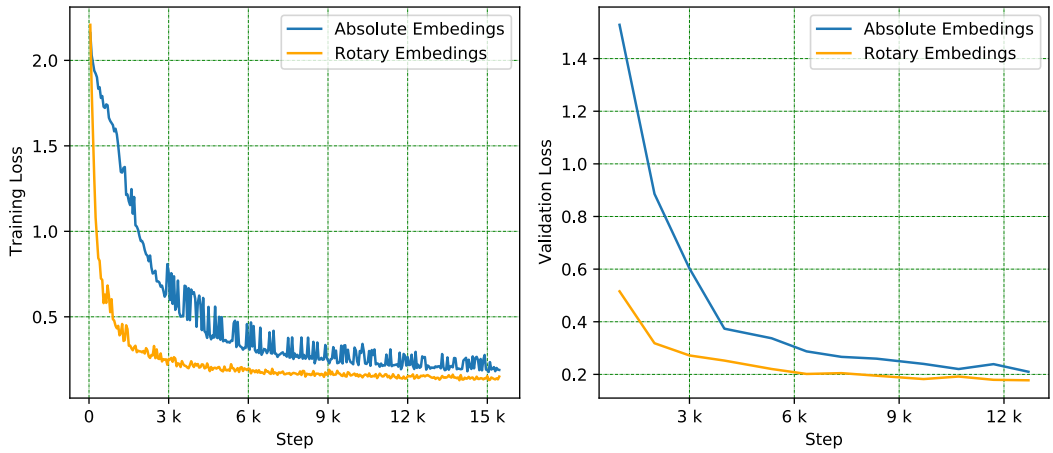
A.2 Parallelization and Computing Environment

All experiments were performed on a GPU cluster where each node contains either 8 NVIDIA Tesla V100 (32GB) or 8 Ampere A100 (40GB) GPUs connected via NVLink. The V100 nodes are equipped with dual 28-core (Intel Xeon Gold 6258R) CPUs, the A100 nodes are equipped with dual 64-core (AMD EPYC 7742) CPUs, and all nodes are connected by 2 non-blocking EDR InfiniBand (100Gbps) network adapters as well as 2 100Gbps Ethernet adapters. All nodes are installed with RHEL 8.3, CUDA 10.2, and cuDNN 7.5.

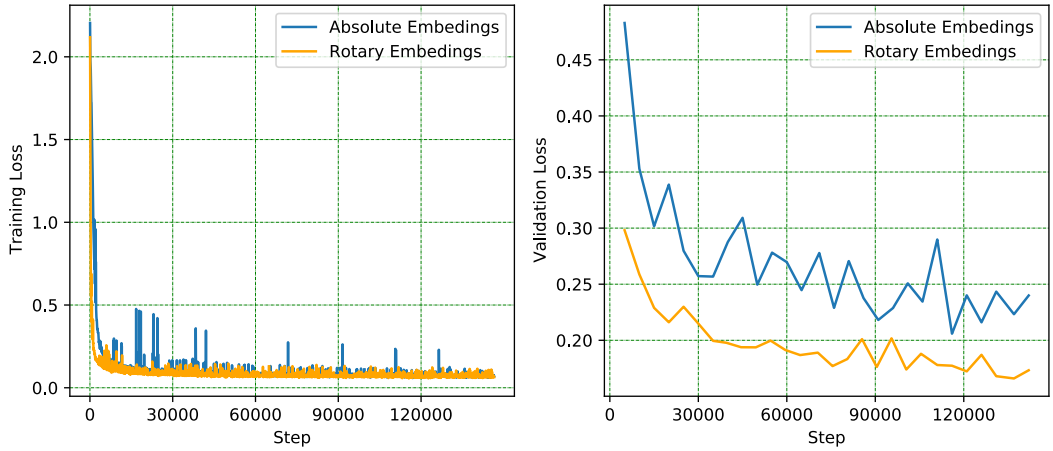
Due to the size of the datasets utilized in pre-training, our training environment relies on the Distributed Data Parallel functions provided by Pytorch and Pytorch Lightning utilizing the NCCL backend. By utilizing RDMA to enable GPU-direct technology we were able to efficiently scale to multi-node multi-GPU training. Additionally, we utilized HuggingFace Datasets to localize the data onto the machines where pre-training took place to improve performance during pre-training. Our pre-training task consists of training on the full dataset to 4 epochs. Training a single epoch of just PubChem on a single NVIDIA V100 GPU would take approximately 60 hours. Utilizing Distributed Data Parallel, pre-training on the full PubChem dataset alone took approx. 22 hours on 16 NVIDIA V100 GPUs this averages to about 5.5 hours per epoch. The speed up achieved by parallelizing training to 16 GPUs gave us a factor of 10.9. Pre-training for 4 epochs on the combined PubChem+ZINC datasets took approx 208 hours on a 16 NVIDIA V100 GPUs which averages to about 52 hours of compute for a single epoch. All fine-tunning tasks were able to be performed on single GPUs (either V100 or A100) and completed in approx. 12 hours.

A.3 Memory Efficient Training with Adaptive Bucketing By Sequence Length

We observed that the distribution of molecule lengths in our dataset centered around molecules that were less than 45 characters long after tokenization. This fact coupled with large batch sizes increased the likelihood that padding tokens would overwhelm each batch and result in large amounts of computational waste. To address this problem we decided to break each minibatch into multiple buckets. This process is done on a batch by batch basis, i.e. on the fly, which means full dataset preprocessing does not take place. It should be noted that gathering of statistics for the full dataset did take place before training and buckets were defined by sequence interval length gathered from that process. The first bucket would contain SMILES strings of length 1 to 42, the next bucket would be of size 43 to 66, the third bucket would be of size 67 to 122 and finally the last bucket would be of size 123 to 202. Due to the length distribution of our dataset buckets 1 and 2 would always be present in all training steps while bucket 3 would be present for the majority of minibatches. Molecules that fell into bucket 4 appeared in most minibatches but would usually only represent a very small percentage of molecules found within the minibatch. With this information we decided to not utilize bucket 4 in the training procedure until it reached a threshold of 50 molecules thus preventing us from training on a bucket that consistently contained a very small amount of molecules which we believe aided training. Bucketing combined with gradient accumulation across buckets gave us stable training, maintained training randomization and reduced computational time needed compared to the traditional method of keeping GPU memory full at all times to maximize computation without consideration of the wasted computation that arises because of the large amount of padding tokens. To be



(a) PubChem Pre-training



(b) PubChem+ZINC Pre-training

Figure 5. Training and validation losses of our Linear attention MOLFORMER with *rotary* (relative) and absolute position embeddings on a) PubChem and b) PubChem+ZINC (>1 billion data points). We see that both rotary and absolute MOLFORMER have graceful training curves. Our Rotary Linear attention MOLFORMER leads to lower training and validation losses than MOLFORMER with absolute position embeddings. This observation lead us to focus on MOLFORMER with rotary position embeddings.

more concrete, without bucketing on 1 V100 GPU and on PubChem only a single epoch would take approx. 1200 hours while with bucketing the same epoch only took around 60 hours giving adaptive bucketing a speedup of 20×. A similar concept, namely micro-batching[72] exists but we became aware of it after implementing our adaptive bucketing technique. We have not yet baselined the differences between our domain specific bucketing implementation towards molecular data against the generic micro-batching[72] . Using Linear attention and bucketing allowed us to reduce the number of GPUs needed for quadratic attention and no bucketing from roughly 1000 to 16.

A.4 Pre-training Scaleout

We give in Figure 6 the estimated training times of MOLFORMER as a function of the used GPUs in the parallelization.

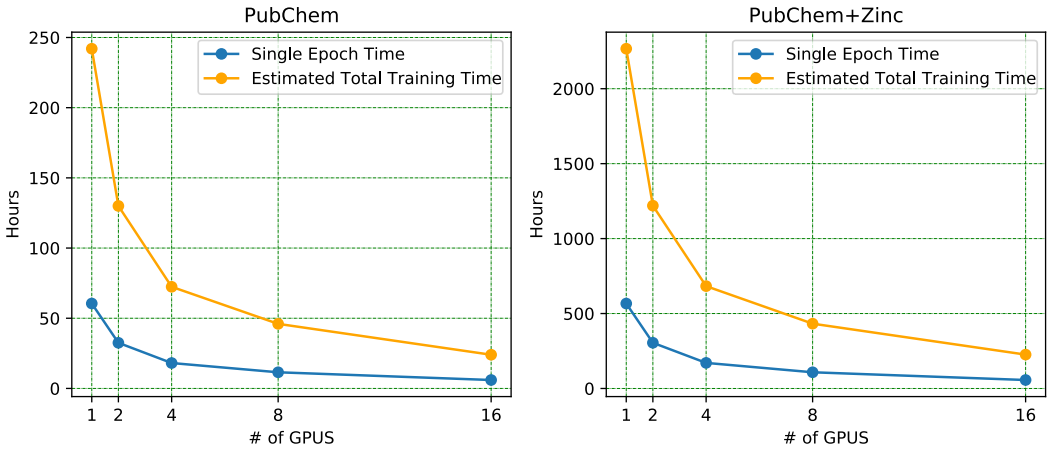


Figure 6. Estimated training times for our Linear attention MOLFORMER with *rotary* embeddings on a) PubChem and b) PubChem+ZINC datasets taken after 250 iterations. We see that training time decreases slightly sub-linearly as GPUs are added. Training time also scales approximately linearly as more data is added.

B Fine-tuning MOLFORMER for Property Prediction

During the fine-tuning process, where the MOLFORMER weights are not frozen, we experimented with different hyperparameters. In our experiments, we found that batch sizes of both 64 and 128 work best for the downstreams tasks. Also, among various learning rates, 3e-5 seems to be the best fit for all the measures on QM9 dataset. We found that the beta values used by the optimzier were import and were set to beta1 equaling .9 and beta2 equaling .99. We found the model to be very sensitive to the betas2 value during fine-tuning. The discriminator used was of a fixed size of 2 layers with a hidden size of 768 for all fine-tuning experiments.

For the frozen strategy where the embeddings from the MOLFORMER are fixed, we use a fully connected model to predict the properties. A hyperparameter sweep was performed for the frozen strategy using grid search and we randomly picked 25 different variations for each task. The best model with the lowest validation loss was picked for further analysis. The different values of the frozen strategy hyperparameters are summarized in the table below.

Table 8. Different Values of the hyperparameters for the Frozen Models

Hyperparameter	Values
Learning Rate	0.001, 0.0001, 0.0005, 0.00005
Batch Size	64, 128, 256
Hidden Dimension	64, 128, 256, 512, 1024
Number of Layers	2, 3, 4

C Dataset, Vocabulary, and Property Units

All our downstream evaluations are performed on tasks from the MoleculeNet dataset[27]. All the tasks mentioned in Table 2 use random splits as suggested in [27], while those in Table 1 use scaffold splits as suggested in[25]. A brief description of the downstream datasets are in Tables 9 and 10. We refer the reader to [27] for more details on the specific tasks.

We have observed that various related work in the past have used different units for quantitative analysis on QM9 dataset without explicitly stating the units, making it difficult to compare the relative performance of different methods. We have listed the units of the measures used in this paper in Table 13. We also give in Tables 11 and 12 the statistics of sequence length and vocabulary for the datasets considered in this work. While the vocabulary of each dataset varies the architecture of our models is identical for all experiments contained in this paper. The vocabulary for our models is defined by the union of the vocabularies of both PubChem and Zinc dataset.

	# of compounds	Description
BBBP	2039	Blood brain barrier penetration dataset
Tox21	7831	Toxicity measurements on 12 different targets
Clintox	1478	Clinical trial toxicity of drugs
HIV	41127	Ability of small molecules to inhibit HIV replication
BACE	1513	Binding results for a set of inhibitors for β – secretase 1
SIDER	1427	Drug side effect on different organ classes

Table 9. Description of classification datasets used for downstream evaluations

	# of compounds	Description
QM9	133885	12 quantum mechanical calculations of small organic molecules with upto nine heavy atoms
QM8	21786	12 excited state properties of small molecules
ESOL	1128	Water solubility dataset
FreeSolv	642	Hydration free energy of small molecules in water
Lipophilicity	4200	Octanol/water distribution coefficient of molecules

Table 10. Minimum, maximum and mean and standard deviation of sequence length for the datasets considered in this work. The vocabulary size after tokenization is also given in the last column.

Pre-trained Data	Min	Max	Mean	Std	Vocab Size
QM9	1	22	14.76	2.02	30
Zinc	4	152	43.08	8.70	113
10 PubChem + 10 Zinc	2	2031	42.52	10.06	1044
PubChem	1	2211	43.24	22.21	2349
100 PubChem + 10 Zinc	2	2211	42.86	16.93	2355
PubChem + Zinc	1	2211	44.76	14.55	2362

Table 11. Description of regression datasets used for downstream evaluations

Pre-trained Data	Most Frequent Tokens
QM9	`C, 1, 0, 2, (`
PubChem	`C, =, (, ), 0`
Zinc	`C, c, (, ), 1`
PubChem + Zinc	`C, c, (, ), =`

Table 12. Top five most frequent tokens for each data source.

D Additional Results and Ablations on QM9 Benchmark

In this section we show additional results and ablations on the pre-training and fine-tuning of MOLFORMER on the QM9 benchmark.

Fine-tuned MOLFORMER-XL versus Baselines We show in Table 14 a comparison on QM9 benchmark between MOLFORMER-XL (fine-tuned on QM9) and other competitive methods that are graph or geometry based (using 3D information) as well as against ChemBERTa a smiles based baseline. We see that MOLFORMER-XL outperforms those baselines in terms of MAE although it does not have access to the 3D information.

Impact of MOLFORMER pre-training dataset on Downstream task We present in Table 15 an ablation on the impact of the pre-training dataset on the performance of fine-tuning MOLFORMER in the downstream propriety prediction tasks on QM9. We see that as the pre-training dataset sizes becomes large, MOLFORMER achieves better performance (i.e lower MAE). Note that MOLFORMER-XL refers to MOLFORMER pre-trained on PubChem+ ZINC.

Measure	Unit
α	Bohr^3
C_v	cal/(mol*K)
G	Hartree
gap	Hartree
H	Hartree
ε_{homo}	Hartree
ε_{lumo}	Hartree
μ	Debye
$\langle R^2 \rangle$	Bohr^2
U_0	Hartree
U	Hartree
ZPVE	Hartree

Table 13. Units of QM9 target measures

	Graph-Based			Geometry-Based			SMILES-Based	
Measure	A-FP	123-gnn	GC	CM	DTNN	MPNN	MOLFORMER-XL	ChemBERTa
α	0.492	**0.27**	1.37	0.85	0.95	0.89	**0.3327**	0.8510
C_v	0.252	**0.0944**	0.65	0.39	0.27	0.42	**0.1447**	0.4234
G	0.893	**0.0469**	3.41	2.27	2.43	2.02	**0.3362**	4.1295
gap	0.00528	**0.0048**	0.01126	0.0086	0.0112	0.0066	**0.0038**	0.0052
H	0.893	**0.0419**	3.41	2.27	2.43	2.02	**0.2522**	4.0853
ε_{homo}	0.00358	**0.00337**	0.00716	0.00506	0.0038	0.00541	**0.0029**	0.0044
ε_{lumo}	0.00415	**0.00351**	0.00921	0.00645	0.0051	0.00623	**0.0027**	0.0041
μ	0.451	0.476	0.583	0.519	**0.244**	**0.358**	0.3616	0.4659
$\langle R^2 \rangle$	26.839	22.90	35.97	46.00	**17.00**	28.5	**17.0620**	86.150
U_0	0.898	**0.0427**	3.41	2.27	2.43	2.05	**0.3211**	3.9811
U	0.893	**0.111**	3.41	2.27	2.43	2.00	**0.2522**	4.3768
ZPVE	0.00207	**0.00019**	0.00299	0.00207	0.0017	0.00216	**0.0003**	0.0023
Avg MAE	2.6355	**1.9995**	4.3536	4.7384	2.3504	3.1898	**1.5894**	8.7067
Avg std MAE	0.0854	**0.0658**	0.1683	0.1281	0.1008	0.1108	**0.0567**	0.1413

Table 14. MOLFORMER performance on QM9 test set. Our best MOLFORMER variant is pre-trained on PubChem+ZINC dataset and fine-tuned for each measure. Baseline performance values are taken from[27,37,39]. **Blue** and **Orange** indicates best and second-best performing model, respectively. MOLFORMER trained with rotary embeddings on PubChem+ZINC achieves the best Avg MAE and Avg. std. MAE across all tasks.

Impact Fine-tuning versus Frozen Embedding/ Rotary versus Absolute on Downstream tasks We show in Table 16 the impact of fine-tuning versus using frozen MOLFORMER embeddings from pre-training phase on the performance on the QM9 benchmark in both rotary and absolute embeddings. We see that fine-tuning and rotary achieve the best performance.

Mean and Standard Deviation for Prediction Results Across Different Data Folds in the fine-tuning phase We also report from Table 16 the mean and standard deviations of MAE for 5 different folds of the data split into 80% training and 10% validation and 10% test. Most of the related work does not perform cross validation and just report the results on a single split. We note that the standard deviations are quite low for most of the predictions and the mean errors are in line with the main paper for all folds of all tasks which suggests the MOLFORMER representations are robust.

Pre-training Data →	QM9 Only			PubChem Only			PubChem+ZINC		
Measure ↓	Frozen × Rotary	Fine-tuned × Rotary	Fine-tuned ✓ Rotary	Frozen × Rotary	Fine-tuned × Rotary	Fine-tuned ✓ Rotary	Frozen × Rotary	Fine-tuned × Rotary	Fine-tuned ✓ Rotary
α	1.6258	**0.5078**	0.6001	1.5470	**0.5280**	0.8452	0.5312	0.3713	**0.3327**
C_v	1.0176	**0.1589**	0.1906	0.9984	**0.1506**	0.2701	0.2303	0.1584	**0.1447**
G	3.2528	0.9985	**0.7479**	2.0089	**0.8626**	1.5920	**0.3066**	0.6861	0.3362
gap	0.0187	**0.0057**	0.0061	0.0182	**0.0050**	0.0109	**0.0036**	0.0039	0.0038
H	1.9221	1.1579	**1.0250**	2.3627	1.3342	**0.7088**	0.3675	.07369	**0.2522**
ε_{homo}	0.0115	**0.0042**	0.0046	0.0147	**0.0038**	0.0082	0.0062	**0.0028**	0.0029
ε_{lumo}	0.0157	**0.0041**	0.0056	0.0148	**0.0036**	0.0080	0.0058	**0.0025**	0.0027
μ	0.8394	**0.4380**	0.4630	0.8509	**0.4284**	0.6166	0.6463	0.3921	**0.3616**
$\langle R^2 \rangle$	86.9461	**24.0785**	25.9482	87.2816	**30.2904**	34.0425	27.5962	18.8286	**17.0620**
U_0	3.0626	**1.1462**	1.3168	2.0613	**0.8969**	1.5503	0.4500	0.4244	**0.3211**
U	1.8555	**1.0454**	1.6158	1.9638	**1.1122**	1.1351	0.4480	0.7370	**0.2522**
ZPVE	0.0020	**0.0011**	0.0012	0.0020	**0.0008**	0.0012	0.0004	**0.0002**	0.0003
Avg MAE	8.3808	**2.4621**	2.6604	8.260	**2.968**	3.3990	2.5497	1.8620	**1.5894**
Avg std MAE	0.2390	0.0843	0.0937	0.2447	0.0801	0.1355	0.0978	0.0611	**0.0567**
# Wins for fixed data	0	**10**	2	0	**11**	1	2	3	**7**

Table 15. Comparison of different MOLFORMER variants on the QM9 test set. Models on the left half of the table are pre-trained using QM9 only and the model in the middle is trained on PubChem only, whereas the models on right half are pre-trained on the PubChem+ZINC dataset. The variants with (✓) and without (×) rotary embeddings are compared. Our best candidate variant (for Table 14) is picked based on the average MAE score.

	PubChem+Zinc			
Measure ↓	Frozen × Rotary	Frozen ✓ Rotary	Fine-tuned × Rotary	Fine-tuned ✓ Rotary
α	0.6468 ± 0.0169	1.2956 ± 0.0244	0.4246 ± 0.0486	0.3455 ± 0.0066
C_v	0.2825 ± 0.0078	0.7959 ± 0.0141	0.2080 ± 0.0340	0.1589 ± 0.0102
G	1.2865 ± 0.1192	2.1535 ± 0.0503	0.7696 ± 0.0719	0.3609 ± 0.0276
gap	0.0089 ± 0.0000	0.0165 ± 0.0002	0.0038 ± 0.0001	0.0040 ± 0.0001
H	1.0730 ± 0.3188	1.8490 ± 0.1469	0.8525 ± 0.0855	0.3318 ± 0.0512
ε_{homo}	0.0066 ± 0.0000	0.0103 ± 0.0001	0.0030 ± 0.0001	0.0029 ± 0.0001
ε_{lumo}	0.0063 ± 0.0000	0.0128 ± 0.0001	0.0039 ± 0.0002	0.0029 ± 0.0001
μ	0.6872 ± 0.0049	0.8089 ± 0.0205	0.3986 ± 0.0060	0.3709 ± 0.0065
$\langle R^2 \rangle$	29.3779 ± 0.2944	72.8752 ± 1.1180	19.9005 ± 0.3305	17.8121 ± 0.8596
U_0	1.2877 ± 0.2373	2.1030 ± 0.0967	0.8492 ± 0.1102	0.3795 ± 0.0820
U	1.3238 ± 0.1954	1.8647 ± 0.0601	0.8631 ± 0.0999	0.3677 ± 0.0382
ZPVE	0.0005 ± 0.0000	0.0018 ± 0.0001	0.0003 ± 0.0001	0.0003 ± 0.0001

Table 16. Mean and standard deviation of Mean Absolute Error (MAE) on the various QM9 tasks using a pre-trained model that is trained on both PubChem and Zinc

E MOLFORMER Attention Visualization and Structure Discovery

In this section we present a visual comparison to show the attention representations of two molecules from the QM9 test dataset (`gdb_62509` and `gdb_1105`) generated by the linear and full attention model variants. Both these models of MOLFORMER-XL have rotary positional embedding in place. We picked `gdb_62509` and `gdb_1105` based on the best cosine similarity from the medium bucket category, and are the same as the ones used in Table 7. For full quantitative comparison on this metric, refer to Table 7. The attention head weights of each layer are averaged and all layers are presented in figures 8 and 9. Please note that the colorbar on the subplots are of different scale for different variants. Several interesting observations can be made from these representations. For both full-attention and its linear counterpart most of the attentions are directed toward the closing parentheses in higher layers (layer 9 and up). Therefore, it is reasonable to avoid other layers for identifying meaningful features with respect to structure. We also notice that intermediate layers 8 and 9, from the linear-attention with rotary variant, capture the 3D structure of the molecules better. This also reinforces observation made in quantitative analysis reflected in

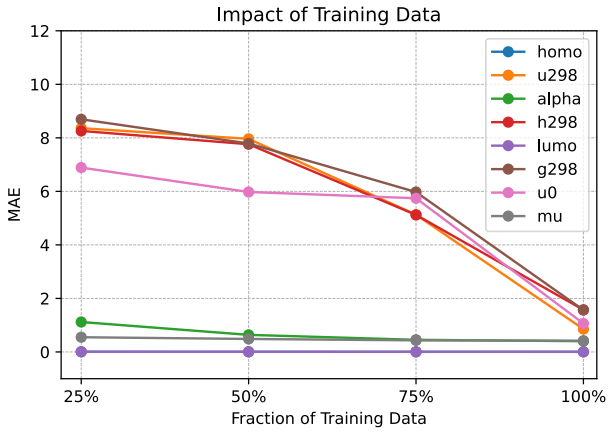
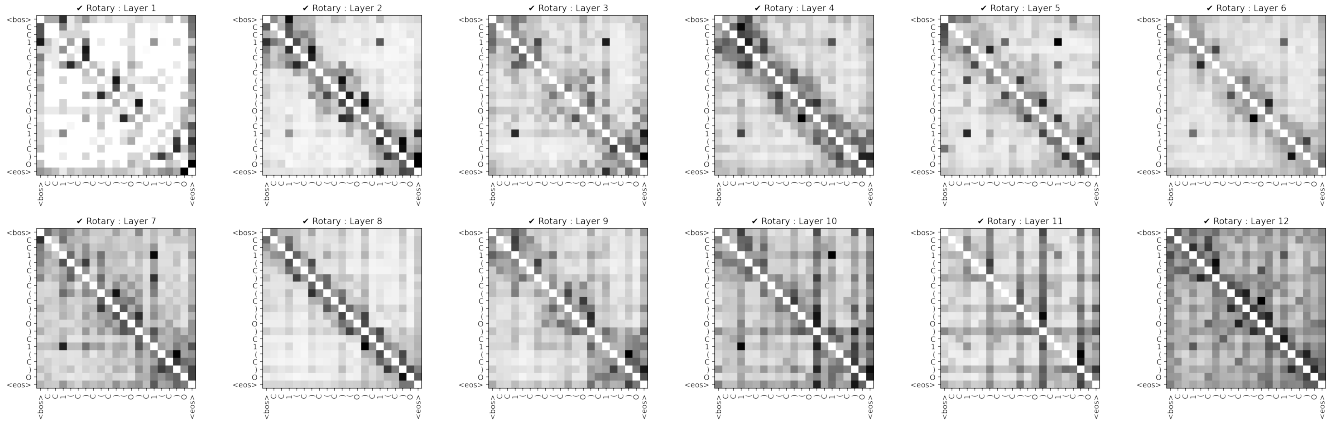
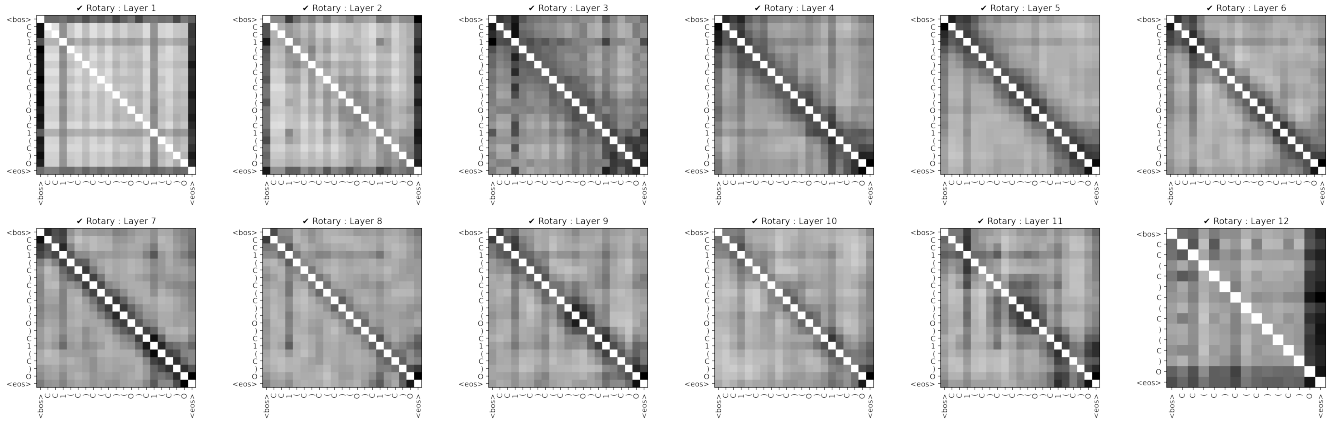


Figure 7. Mean absolute errors with varying training set size. Fine-tuning of MOLFORMER with rotary embeddings for prediction of various properties on QM9 Molecules for different training set sizes.

Table 7. While the model variations have similar in downstream task performance in our evaluations the differences in their attention weights and in their ability of discovering structural information from SMILES representation are insightful and intriguing. Specifically with regard to how the linear-attention embedding captures structural information of molecules. Also, it is worth observing that the MOLFORMER-XL variants here are not fine-tuned for the QM-9 dataset and yet the attentions manage to represent the structure.

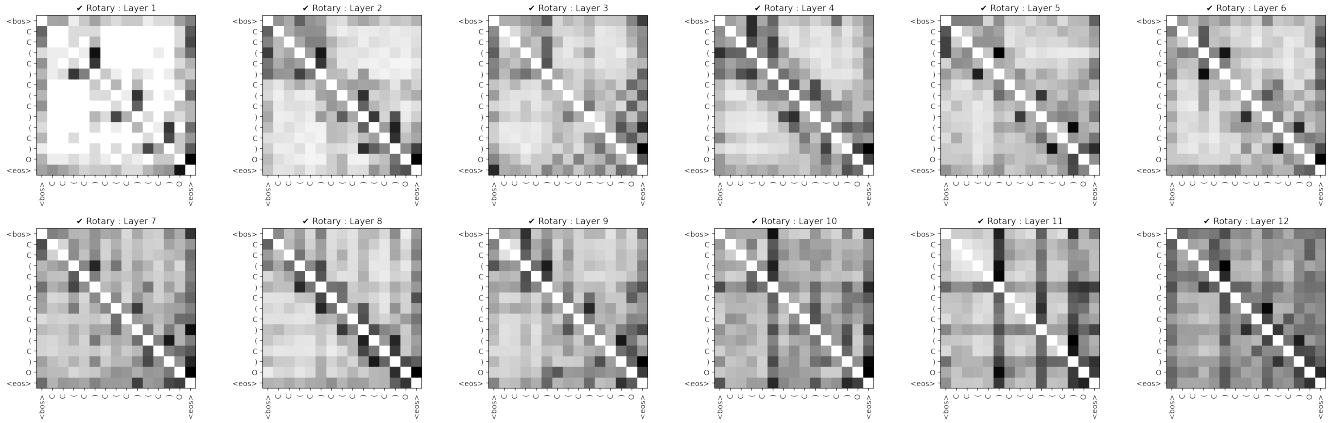


(a) Attention Map for full-attention & rotary/relative embedding variant of MOLFORMER

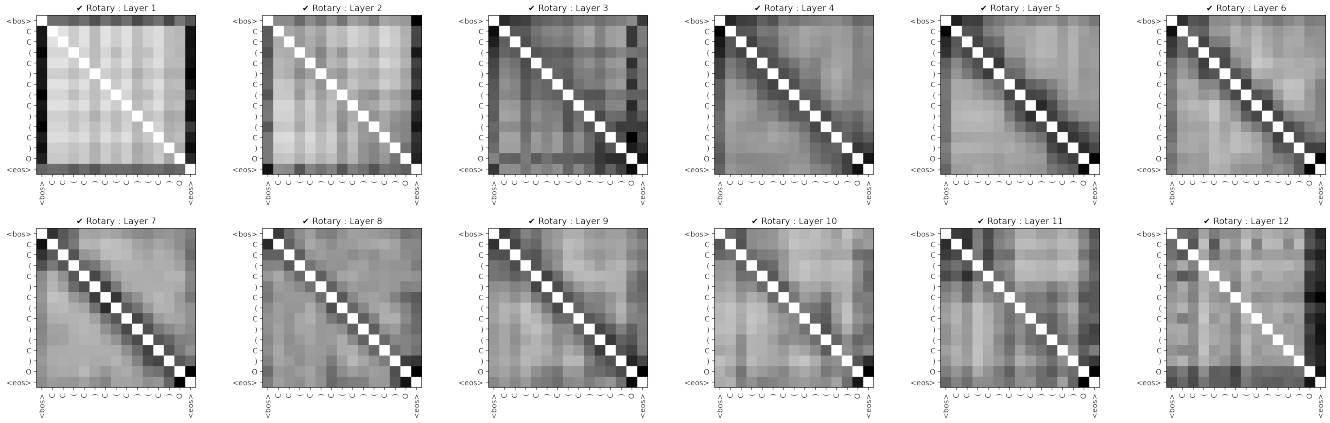


(b) Attention Map for linear-attention & rotary/relative embedding variant of MOLFORMER

Figure 8. Attention map shown for the smile sequence, `gdb_62509`, `'CC1(C)C(C)(O)C1(C)O'` for all the layers of MOLFORMER. For each layer, the attentions are average-pooled across all the heads.



(a) Attention map for full-attention & rotary/relative embedding variant of MOLFORMER



(b) Attention Map for linear-attention & rotary/relative embedding variant of MOLFORMER

Figure 9. Attention map shown for the smile sequence, `gdb_1105`, `'CC(C)C(C)(C)O'` for all the layers of MOLFORMER. For each layer, the attentions are average-pooled across all the heads.

F Assets and License

The following table summarizes the libraries utilized for our experiments and their accompanying license terms.

Asset	License	Link
Fast-Transformers	MIT License	https://github.com/idiap/fast-transformers/blob/master/LICENSE
Pytorch	BSD Style License	https://github.com/pytorch/pytorch/blob/master/LICENSE
RDKit	BSD 3-Clause "New" or "Revised" License	https://github.com/rdkit/rdkit/blob/master/license.txt
Pytorch Lightning	Apache 2.0 License	https://github.com/PyTorchLightning/pytorch-lightning/blob/master/LICENSE
HuggingFace Datasets	Apache 2.0 License	https://github.com/huggingface/datasets/blob/master/LICENSE
Nvidia APEX	BSD 3-Clause "New" or "Revised" License	https://github.com/NVIDIA/apex/blob/master/LICENSE