nature machine intelligence

Article

https://doi.org/10.1038/s42256-023-00639-z

Regression Transformer enables concurrent sequence regression and generation for molecular language modelling

In the format provided by the authors and unedited

Supplementary Information

S1 Numerical encodings

S1.1 Description of Integer encodings.

As an alternative to the float-based numerical encodings (NE), we experimented with an encoding scheme relying solely on positive integers. Note that any regression problem can trivially be casted to a regression problem where all labels are positive integers. Under this consideration, we need to define NEs only for positive integers[1]; similar to positional encodings. We therefore propose to directly utilize the definition from Vaswani et al. [1] as NEs:

$$\begin{aligned} NE_{Int}(v, p, 2j) &= \sin\left[(v \cdot 10^p)/10000^{2j/d_e}\right] \\ NE_{Int}(v, p, 2j{+}1) &= \cos\left[(v \cdot 10^p)/10000^{2j/d_e}\right] \end{aligned} \tag{8}$$

where d_e is the embedding size. The advantage of this integer-based encoding is that every embedding dimension captures fluctuations of different frequencies; using trigonometric functions as continuous analogs to alternating bits. Practically, to use the Integer-NEs, the property values were casted to the range $[0, 1000]$ and rounded.

S1.2 Float vs. Integer-based numerical encodings

In Table S1 we report an ablation study on using integer-based numerical encodings instead of the float-based encodings used throughout the rest of the paper. For the integer-based encodings the QED property values were cast to $[0, 1000]$. In this setting,

Table S1: **Ablation study on alternative type on numerical encodings**. All models used SELFIES. The best embedding type (float, integer, none) for each training configuration and each metric is highlighted in bold.

Configuration				*Regression task*		*Generation task*	
Pretraining	Finetuning	NE	Perplexity (↓)	RMSE (↓)	PCC (↑)	0-Var (↓)	SCC (↑)
PLM	–	–	$1.61_{\pm 0.03}$	$0.0591_{\pm 0.00}$	$0.968_{\pm 0.00}$	$0.9\%_{\pm 0.2}$	$0.43_{\pm 0.01}$
PLM	–	Float	$\mathbf{1.59}_{\pm 0.03}$	$\mathbf{0.0547}_{\pm 0.01}$	$\mathbf{0.971}_{\pm 0.00}$	$\mathbf{0.3\%}_{\pm 0.1}$	$\mathbf{0.47}_{\pm 0.01}$
PLM	–	Integer	$1.63_{\pm 0.02}$	$0.0564_{\pm 0.00}$	$0.968_{\pm 0.00}$	$0.8\%_{\pm 0.3}$	$0.44_{\pm 0.01}$
PLM	Alternate ($\mathcal{L}_P$ & $\mathcal{L}_G$)	–	$2.57_{\pm 0.1}$	$0.034_{\pm 0.01}$	$0.988_{\pm 0.01}$	$\mathbf{0.2\%}_{\pm 0.1}$	$\mathbf{0.47}_{\pm 0.02}$
PLM	Alternate ($\mathcal{L}_P$ & $\mathcal{L}_G$)	Float	$\mathbf{2.10}_{\pm 0.1}$	$0.0498_{\pm 0.00}$	$0.982_{\pm 0.00}$	$0.3\%_{\pm 0.1}$	$\mathbf{0.47}_{\pm 0.03}$
PLM	Alternate ($\mathcal{L}_P$ & $\mathcal{L}_G$)	Integer	$2.63_{\pm 0.1}$	$\mathbf{0.0307}_{\pm 0.01}$	$\mathbf{0.990}_{\pm 0.01}$	$0.7\%_{\pm 0.2}$	$0.41_{\pm 0.01}$
PLM	Alternate w SC ($\mathcal{L}_P$ & $\mathcal{L}_{SC}$)	–	$\mathbf{2.41}_{\pm 0.1}$	$0.0483_{\pm 0.01}$	$0.978_{\pm 0.03}$	$0.3\%_{\pm 0.1}$	$0.49_{\pm 0.01}$
PLM	Alternate w SC ($\mathcal{L}_P$ & $\mathcal{L}_{SC}$)	Float	$2.67_{\pm 0.1}$	$\mathbf{0.0367}_{\pm 0.03}$	$\mathbf{0.987}_{\pm 0.03}$	$\mathbf{0.2\%}_{\pm 0.1}$	$\mathbf{0.52}_{\pm 0.02}$
PLM	Alternate w SC ($\mathcal{L}_P$ & $\mathcal{L}_{SC}$)	Integer	$2.71_{\pm 0.1}$	$0.0412_{\pm 0.02}$	$0.986_{\pm 0.02}$	$0.8\%_{\pm 0.3}$	$0.44_{\pm 0.01}$

from the two types of proposed numerical encodings, the float-based encodings yielded slightly superior result to integer-based encodings. Here, integer-encodings are superior for regression but inferior for conditional generation. Due to that and their non-applicability to floating numbers, we decided to not further explore them.

S1.3 Summation vs. concatenation of numerical encodings.

We decided to follow the common approach of *summing* additional encodings with the learned embeddings [1, 12] but note that disentangling content and position embeddings can improve language models [85]. So, instead of summing the numerical encodings to the regular embeddings, we also experimented with concatenation (dimensionality of 32 for the NEs.). This produced slightly inferior but nearly identical results, see Table S2. We propose to use a summation for two reasons: First, it avoids additional hyperparameters and model weights. Secondly, it probably yields approximately orthogonal subspaces of token embedding and numerical encodings (due to the high dimensionality). This obviates the need to enforce orthogonality with a concatenation. While we conjectured that using NEs improves the performance in both tasks (property prediction and conditional generation), we emphasize that providing this prior might not be necessary given enough data. We hypothesize that refining our NEs might yield better results and in particular a faster convergence, but leave further refinement to future work, especially given the plethora of research about positional encodings [86, 87, 88].

[1] Strictly speaking only integers with a single, non-zero digit (i.e., covered by the base-10 exponentiation of the decimal system)

Table S2: **Ablation study on NEs.** Results on PLM training.

NE	Type	RMSE (↓)	PCC (↑)
–	–	$0.0591_{\pm 0.00}$	$0.968_{\pm 0.00}$
Float	Concat.	$0.0581_{\pm 0.00}$	$0.966_{\pm 0.01}$
Float	Sum	$\mathbf{0.0547}_{\pm 0.01}$	$\mathbf{0.971}_{\pm 0.00}$
Int	Concat.	$0.0666_{\pm 0.01}$	$0.963_{\pm 0.01}$
Int	Sum	$0.0564_{\pm 0.00}$	$0.968_{\pm 0.00}$

S2 Chemical sensibility

To assess the chemical sensibility of the generated molecules, we drew inspiration from Zhang et al. [89]. Relying on the definition of Ertl [45], we computed the functional groups (FG) present in the QED training and testing dataset as well as the generated molecules. In Figure S1 we show the frequencies of the 1000 most frequent groups in the testing data, sorted by their frequency in the testing data.

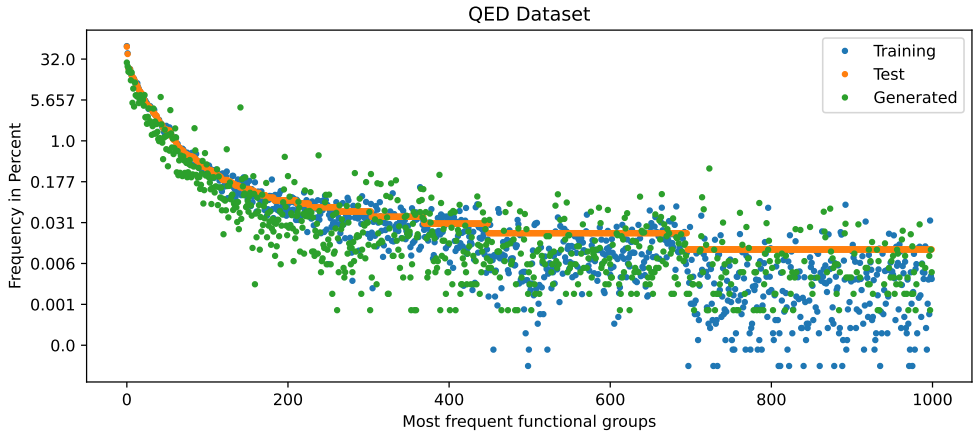

Figure S1: Frequency of functional groups in molecules used for training (blue), testing (orange) and generated molecules (green).

It can be seen that training molecules and the generated molecules largely replicate the frequencies of functional groups of the test data. To assess this more quantitatively, we measured the Wasserstein distance between the absolute counts of the FGs for the different data chunks. We observe that the distribution of FGs in the generated molecules is more similar to the testing data (d = 3.4) than the training data (d = 6.1).

S3 Conditional molecular generation (constrained property optimization benchmark)

On the constrained property optimization benchmark, we conducted ablation studies of the Regression Transformer for the use of float-based numerical encodings (NE) as well as the self-consistency loss function. The main metric of this task is the mean improvement in pLogP compared to the seed molecule. The results can be found in Table S3. The value of α refers to Equation 7: $\alpha = 0$ means that no self-consistency loss was used and $\alpha = 1$ implies that the self-consistency loss was used with a weight equal to the regular conditional text generation objective (cf. Equation 6). The results of the ablation study indicate that the RT consistently outperformed the JT-VAE and GCPN in the main metric (improvement) by a wide margin.

Table S3: **Further results on constrained property optimization benchmark.** JT-VAE is from Jin et al. [42], GCPN from You et al. [49], BT from Fan et al. [50] and MoFlow from Zang and Wang [51]. NE refers to the use of Numerical Encodings. Best model shown in bold, second best underlined. Standard deviations shown.

	Training configuration		*Generation task*			*Regression*
Model	Numerical Encoding	Self-consistency (α)	**Improvement**	Similarity δ	Success rate	Pearson's r (PCC)
JT-VAE	–	–	$1.91_{\pm 2.0}$	$0.28_{\pm 0.2}$	97.5%	*Unfeasible*
GCPN	–	–	$4.20_{\pm 1.3}$	$\underline{0.32}_{\pm 0.1}$	**100%**	*Unfeasible*
MoFlow	–	–	$\underline{8.61}_{\pm 5.4}$	$\mathbf{0.32}_{\pm 0.1}$	98.9%	*Unfeasible*
RT	✓	1	$\mathbf{8.67}_{\pm 2.5}$	$0.10_{\pm 0.1}$	**100%**	0.92
RT	✓	0	$7.96_{\pm 2.6}$	$0.11_{\pm 0.1}$	**100%**	0.90
RT	✗	1	$8.52_{\pm 2.5}$	$0.10_{\pm 0.1}$	**100%**	0.91
RT	✗	0	$8.35_{\pm 2.6}$	$0.10_{\pm 0.1}$	**100%**	**0.94**

(a) **No similarity threshold** ($\delta = 0.0$)

	Training configuration		*Generation task*			*Regression*
Model	Numerical Encoding	Self-consistency (α)	**Improvement**	Similarity δ	Success rate	Pearson's r (PCC)
JT-VAE	–	–	$1.68_{\pm 1.9}$	$0.33_{\pm 0.1}$	97.1%	*Unfeasible*
GCPN	–	–	$4.12_{\pm 1.2}$	$0.34_{\pm 0.1}$	**100%**	*Unfeasible*
MoFlow	–	–	$\mathbf{7.06}_{\pm 5.0}$	$\mathbf{0.43}_{\pm 0.2}$	96.8%	*Unfeasible*
RT	✓	1	$\underline{4.45}_{\pm 1.7}$	$0.35_{\pm 0.1}$	99.6%	0.92
RT	✓	0	$4.12_{\pm 1.7}$	$\underline{0.36}_{\pm 0.1}$	99.6%	0.90
RT	✗	1	$4.34_{\pm 1.6}$	$0.35_{\pm 0.1}$	$\underline{99.9\%}$	0.91
RT	✗	0	$4.40_{\pm 1.7}$	$0.35_{\pm 0.1}$	99.7%	**0.94**

(a) **Similarity threshold** $\delta = 0.2$

	Training configuration		*Generation task*			*Regression*
Model	Numerical Encoding	Self-consistency (α)	**Improvement**	Similarity δ	Success rate	Pearson's r (PCC)
JT-VAE	–	–	$0.84_{\pm 1.5}$	$0.51_{\pm 0.1}$	83.6%	*Unfeasible*
GCPN	–	–	$2.49_{\pm 1.3}$	$0.47_{\pm 0.1}$	**100%**	*Unfeasible*
BT	–	–	$\underline{4.21}$	N.A.	N.A.	*Unfeasible*
MoFlow	–	–	$\mathbf{4.71}_{\pm 4.6}$	$\mathbf{0.61}_{\pm 0.2}$	85.8	*Unfeasible*
RT	✓	1	$3.16_{\pm 1.5}$	$0.54_{\pm 0.1}$	$\underline{97.1\%}$	$\underline{0.92}$
RT	✓	0	$2.87_{\pm 1.5}$	$\underline{0.55}_{\pm 0.1}$	95.5%	0.90
RT	✗	1	$3.09_{\pm 1.5}$	$0.54_{\pm 0.1}$	97.2%	0.91
RT	✗	0	$3.04_{\pm 1.5}$	$0.54_{\pm 0.1}$	97.2%	**0.94**

(a) **Similarity threshold** $\delta = 0.4$

	Training configuration		*Generation task*			*Regression*
Model	Numerical Encoding	Self-consistency (α)	**Improvement**	Similarity δ	Success rate	Pearson's r (PCC)
JT-VAE	–	–	$0.21_{\pm 0.7}$	$\underline{0.69}_{\pm 0.1}$	46.4%	*Unfeasible*
GCPN	–	–	$0.79_{\pm 0.6}$	$0.68_{\pm 0.1}$	**100%**	*Unfeasible*
BT	–	–	**2.77**	N.A.	N.A.	*Unfeasible*
MoFlow	–	–	$2.10_{\pm 2.9}$	$\mathbf{0.79}_{\pm 0.1}$	58.3%	*Unfeasible*
RT	✓	1	$\underline{2.21}_{\pm 1.3}$	$\underline{0.69}_{\pm 0.1}$	$\underline{81.7\%}$	$\underline{0.92}$
RT	✓	0	$2.04_{\pm 1.3}$	$\underline{0.69}_{\pm 0.1}$	75.0%	0.90
RT	✗	1	$2.10_{\pm 1.3}$	$\underline{0.69}_{\pm 0.1}$	81.1%	0.91
RT	✗	0	$2.10_{\pm 1.3}$	$\underline{0.69}_{\pm 0.1}$	81.6%	**0.94**

(a) **Similarity threshold** $\delta = 0.6$

S4 Protein sequence language modeling

S4.1 Impact of loss functions (Protein interaction data)

Like for the QED dataset, for protein sequence modeling we also investigated the impact of the three training loss setups:

1. the vanilla PLM objective (Equation 3),
2. alternating objective with the PLM-based text loss ($\mathcal{L}_G$; Equation 6, equivalent to setting $\alpha = 0$ in Equation 7),
3. alternating objectives with the self-consistency term ($\mathcal{L}_{SC}$; Equation 7; $\alpha = 1$).

The results in Table S4 show that the proposed training scheme with alternating optimization of property tokens and text tokens was highly effective for both, the regression and the generation task. In addition, like on the QED dataset, the self-consistency

Table S4: **Ablation study on training schemes for Boman dataset.** Legend like Table 3.

		Regression task		*Generation task*	
Model	Loss	RMSE (↓)	Pearson's r (↑)	0-Var (↓)	Spearman ρ (↑)
k-NN	–	0.53	0.932	*Task unfeasible*	
RT	PLM	$0.69_{\pm 0.03}$	$0.944_{\pm 0.02}$	$0.3_{\pm 0.4}$	$0.76_{\pm 0.03}$
RT	$\mathcal{L}_G$	$\mathbf{0.17}_{\pm 0.04}$	$\mathbf{0.994}_{\pm 0.01}$	$\mathbf{0.2}_{\pm 0.1}$	$0.82_{\pm 0.01}$
RT	$\mathcal{L}_{SC}$	$0.20_{\pm 0.04}$	$0.991_{\pm 0.01}$	$\mathbf{0.2}_{\pm 0.1}$	$\mathbf{0.84}_{\pm 0.00}$

loss led to better results in conditional generation, but at the expense of slightly reduced accuracy in regression. As stated in the main text, this is most likely caused by the self-evaluations of the decoded sequences. These sequences might differ significantly from the training sequences but are still used with the property value of the original sequences. Since the Boman index can be computed directly from the sequence, this hypothesis could, in principle, be confirmed by correcting the property value during the self-evaluation call. However, limited value would come from such approach because real datasets work with more complex properties.

Apart from that, Figure S2 reveals a general trend in the conditional generation with the Regression Transformer: More freedom in the generative process (i.e., a higher fraction of masked amino acid residues) leads to better results in terms of Spearman ρ to the property primers (cf. Figure S2). This comes, however, at the cost of reduced similarity to the seed sequence.

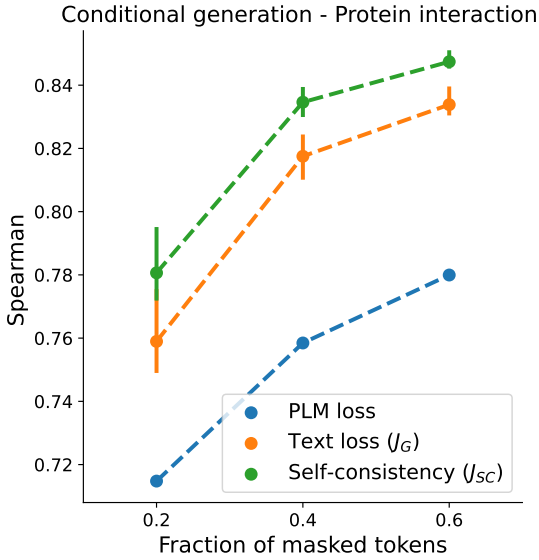

Figure S2: **Correlation between property primer and property of generated protein sequences** The model's ability to generate protein sequences with a desired protein interaction index. The self-consistency loss yielded the best results and, generally, a higher fraction of masked tokens led to generated peptides that adhere better to the primed property value. Note that the Boman/protein interaction index can be assessed *in-silico* from the sequence alone. Data presented as means, error bars denote 95% confidence intervals. For the text-loss and self-consistency models, three models were evaluated respectively.

S5 Chemical reaction modeling

This subsection lists additional results related to the reaction yield modeling. Table S5 reports an ablation study on the impact of p_{mask} (i.e., the probability to mask a specific token) for the reconstruction of additives in Buchwald-Hartwig aminations. Table S6 and Table S7 report an ablation study that assess whether co-encoding reaction yield enables the model to better reconstruct precursors.

p_{mask}	Top-3 acc.	Tanimoto sim.
1.0	$1.36\%_{\pm 0.5}$	$0.158_{\pm 0.002}$
0.5	$11.47\%_{\pm 1.0}$	$0.316_{\pm 0.002}$
0.25	$46.74\%_{\pm 3.5}$	$0.645_{\pm 0.003}$

Table S5: Performance in generating additives for Buchwald-Hartwig reactions [57] as a function of p_{mask}, i.e., the fraction of tokens in the additive that are masked. Generation was primed with remaining precursors and yield.

Precursor type	Top-3 accuracy		Tanimoto similarity		Unique n
	Prec. + Yield	Precursors	Prec. + Yield	Precursors	
Aryl halide	$98.23\%_{\pm 0.5}$	$98.21\%_{\pm 0.4}$	$0.991_{\pm 0.003}$	$0.991_{\pm 0.002}$	15
Ligand	$50.38\%_{\pm 1.6}$	$50.43\%_{\pm 1.7}$	$0.677_{\pm 0.010}$	$0.678_{\pm 0.010}$	4
Base	$100.0\%_{\pm 0.0}$	$100.0\%_{\pm 0.6}$	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	3
Additive	$1.36\%_{\pm 0.5}$	$1.25\%_{\pm 0.8}$	$0.158_{\pm 0.018}$	$0.158_{\pm 0.019}$	22

Table S6: Generating precursors for Buchwald-Hartwig reactions [57] based on remaining precursors or remaining precursors and yield. Full precursors were generated ($p_{mask} = 1$). Unique n denotes the number of unique samples per entity in the training dataset.

Precursor type	Top-3 accuracy		Tanimoto similarity		Unique n
	Prec. + Yield	Precursors	Prec. + Yield	Precursors	
Electrophile	$44.19\%_{\pm 17.6}$	$31.39\%_{\pm 15.3}$	$0.732_{\pm 0.160}$	$0.591_{\pm 0.141}$	7
Nucleophile	$100.0\%_{\pm 0.0}$	$100.0\%_{\pm 0.0}$	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	4
Ligand	$67.43\%_{\pm 20.0}$	$67.59\%_{\pm 19.8}$	$0.689_{\pm 0.152}$	$0.690_{\pm 0.152}$	5
Base	$90.53\%_{\pm 1.2}$	$90.50\%_{\pm 1.4}$	$0.811_{\pm 0.006}$	$0.811_{\pm 0.001}$	8
Solvent	$56.74\%_{\pm 1.1}$	$56.52\%_{\pm 1.0}$	$0.661_{\pm 0.009}$	$0.660_{\pm 0.007}$	4

Table S7: Generating precursors for Suzuki-cross-couplings reactions [58] based on remaining precursors or remaining precursors and yield. Full precursors were generated ($p_{mask} = 1$). Unique n denotes the number of unique samples per entity in the training dataset.

S6 Case studies

S6.1 Case study on scaffold decoration

Scaffold decoration is an exercise in medicinal chemistry that aims to discover novel compounds by modifying substituents (e.g., R-groups) while preserving a central, core structure (the scaffold).. We simulated this task on the QED dataset by determining the Murcko scaffold [90] with `RDKit` and masking only the non-scaffold tokens (in contrast to the regular evaluation where randomly 40% of the tokens were masked). This task was only performed with the SMILES models since scaffolds cannot be determined trivially in SELFIES. In general, this task is more challenging because the molecule is more constrained. On average, less tokens are being masked and in most cases the full range of drug-likeness cannot be captured, given the scaffold. This explains the higher percentage of molecules where the primer did not influence the generations (cf. Table S8).

Table S8: **Scaffold decoration performance for SMILES model.** No numerical encodings were used. No standard deviations are available for the scaffold results since the masking is deterministic.

Text loss	Task	0-Var (↓)	Spearman's ρ (↑)
$\mathcal{L}_G$	Masking non-scaffold	8.55%	0.136
$\mathcal{L}_{SC}$	Masking non-scaffold	9.76%	0.105
$\mathcal{L}_G$	Masking randomly	$0.80\%_{\pm 0.19}$	$0.108_{\pm 0.01}$
$\mathcal{L}_{SC}$	Masking randomly	$1.14\%_{\pm 0.19}$	$0.085_{\pm 0.02}$

But note, that this includes cases where the molecule is itself a scaffold and thus no tokens are masked (we do not control for that explicitly). The generations for one exemplary molecule are shown in Figure S3. In this example, it is interesting to see that the model decorated the scaffold with specific atoms on the rightmost six-ring. These atoms, iodine, chlorine and bromine which were rightfully provided from low to high QED primers seem to be indicative of different levels of drug-likeness. One drawback, however, is that the RT cannot fill no or multiple tokens in the position of one `[MASK]` location. For example, in the case of the

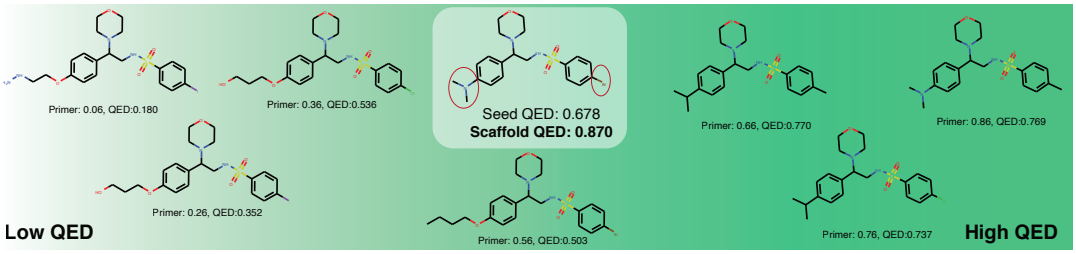

Figure S3: **Molecules sampled in a scaffold decoration task.** Only non-scaffold tokens (encircled in red) were masked.

last primer (0.86), the provided scaffold already had a QED of 0.87 and thus not adding any new atoms would have been the best choice here.

S6.2 Case study on attention visualization: Interpreting attention heads

We visualized the attention scores using BertViz [91]. Here, we aimed to compare the inference patterns across the two tasks, property prediction and conditional generation. The results for the first 4 (out of 32) layers are shown in Figure S4. In general, many attention patterns commonly described in natural language models are also present in the Regression Transformer. For example the bag-of-words pattern (i.e., evenly distributed attention, e.g., all heads of first layer) or the next-token (e.g., layer 4, head 4 and 5) or previous-token patterns (e.g., layer 2, head 2) are clearly visible. While the named patterns are consistently present in both tasks, probably because they are useful irrespective of the particular task, some distinctive patterns for either of the tasks can be found. For example, in the conditional generation task (Figure S4, right) many triangles with their right angle in the upper right are present. In these positions the property tokens are present and thus these patterns indicate that the representation of all other tokens, especially also the masked ones, are heavily influenced by the property value. Instead, in the property prediction task (Figure S4, left), many triangles with their right angle in the lower right are present. This implies a heavy attention on the `[END]` token which marks the end of the sequence and is a useful indicator for the QED score because it is critically influenced by the size/weight of the molecule. One particularly interesting attention head is head 3 in layer 2. In the property prediction task its role is to make the masked property tokens aware of the sequence length. In the conditional generation task, its role is to make all tokens aware of the property values.

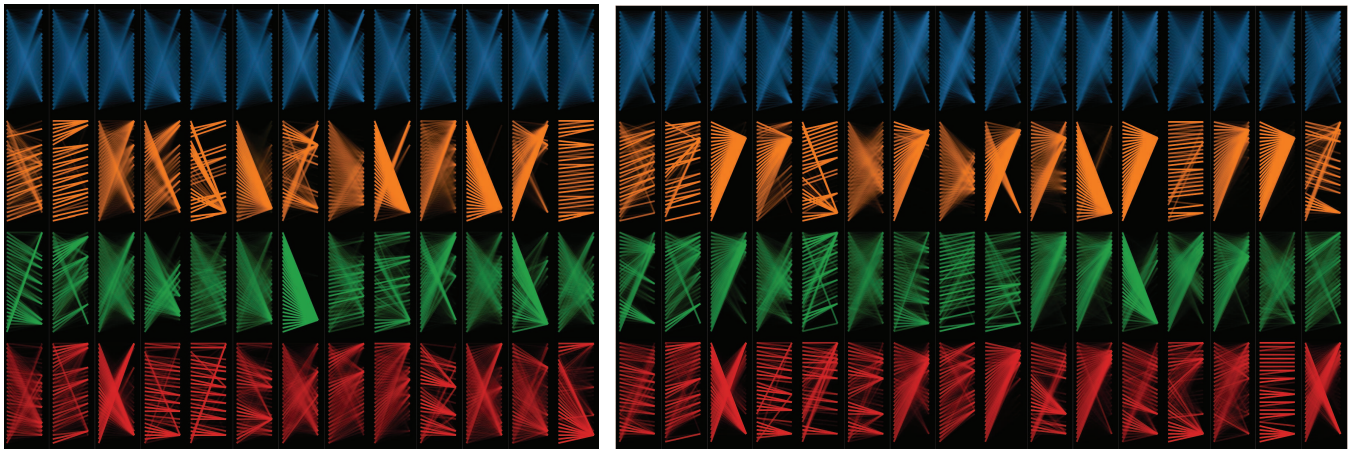

Figure S4: **Comparing attention scores across both tasks with BertViz [91].** Attention scores for all heads of the first four layers. Rows depict layers, column depict attention heads. Within each cell, the tokens are ordered from top to bottom. *Left:* Property prediction task. *Right:* Conditional generation task. Plot performed with SELFIES model with float encodings, trained on the self-consistency loss.

S7 Additional results

S7.1 Failure mode in fluourescence dataset from TAPE

This dataset is particularly interesting due to its bimodal mode: one mode corresponding to bright proteins, the other to dark proteins (cf. Figure S5). An interesting failure mode was observed when initially training on the Fluorescence dataset. Figure S5 shows that the dark mode has one sharp spike, exactly at a log fluorescence value of 1.301. Almost 10% of all training samples and almost 50% of the proteins in the dark mode have this exact value. The Regression Transformer is trained on a classification loss and so, the loss during training for such samples will be distributed across the five tokens `1_0`, `.`, `3_0`, `0_-1` and `1_-2`. In many cases, the model collapsed to always predicting 3.301 where the first token (`3_1`) was correct for all samples in the bright mode and the remaining tokens (`3_0`, `0_-1` and `1_-2` were correct for most samples in the dark model. This happened because no weighting of the individual numerical tokens was applied. As a non-algorithmic remedy, we added Gaussian noise to those training samples.

S7.2 Detailed performance - Classification vs. Regression

Figure S5 reveals the improved performance of the RT compared to the finetuned TAPE Transformer: Less samples were predicted in the wrong mode. However, the RT had difficulties with a fine grained regression, in particular in the bright mode. This becomes particularly apparent when inspecting the detailed results, grouped by bright and dark test proteins respectively

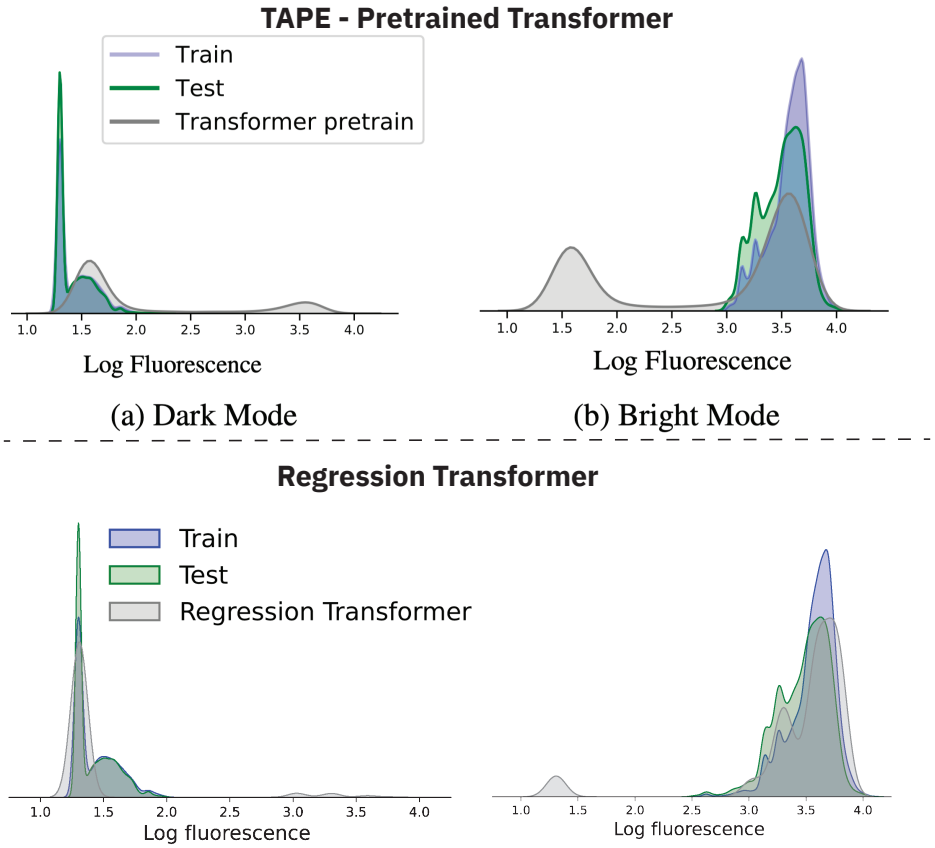

Figure S5: **Bimodal mode of fluorescence data.** The upper part of the plot has been copied from Rao et al. [43] (Figure 3). It shows the bimodal mode of the training data and the test predictions from the TAPE Transformer. At the bottom, we show our remake of the above plot by replacing the predictions from the pretrained TAPE Transformer with the predictions from the Regression Transformer.

(cf. Table S9). While the RT achieved the best results in the overall Spearman ρ, the recommended metric by [43], it does not

Table S9: **Detailed fluorescence prediction results.** MSE abbreviates mean squared error. TAPE and UniRep performances taken from Rao et al. [43]. For the RT all standard deviations on ρ and MSE were < 0.05 and < 0.1 respectively.

		Full test set		Bright proteins		Dark proteins	
Model	Source	MSE	ρ	MSE	ρ	MSE	ρ
One-Hot	TAPE	2.69	0.14	0.08	0.03	3.95	0.00
k-NN	**Ours**	2.31	0.59	**0.05**	0.30	3.37	0.04
Pretr. LSTM	TAPE	**0.19**	0.67	0.12	0.62	**0.22**	0.04
Pretr. Transf.	TAPE	0.22	0.68	0.09	0.60	0.29	**0.05**
UniRep	UniRep	0.20	0.67	0.13	**0.63**	0.24	0.04
RT	**Ours**	0.34	**0.72**	0.19	0.45	0.40	0.04

dominate any of the mode-specific metrics. This is a noteworthy finding because it reflects the tendency of the RT to strive for a multi-class classification rather than performing a full regression. It is also interesting to see that the baseline models (k-NN and TAPE One-Hot) achieved the best results in MSE of bright proteins.