REGRESSION TRANSFORMER: CONCURRENT CONDITIONAL GENERATION AND REGRESSION BY BLENDING NUMERICAL AND TEXTUAL TOKENS

Jannis Born[1,2], Matteo Manica[1]
[1] Accelerated Discovery, IBM Research Europe, 8804 Rüschlikon, Zurich, Switzerland
[1] Department of Biosystems Science & Engineering, ETH Zurich, 4058 Basel, Switzerland
{jab,tte}@zurich.ibm.com

ABSTRACT

We report the Regression Transformer (RT), a method that abstracts regression as a conditional sequence modeling problem. The RT casts continuous properties as sequences of numerical tokens and encodes them jointly with conventional tokens. This yields a dichotomous model that can seamlessly transition between solving regression tasks and conditional generation tasks; solely governed by the mask location. We propose several extensions to the XLNet objective and adopt an alternating training scheme to concurrently optimize property prediction and conditional text generation with on a self-consistency loss. Our experiments on both chemical and protein languages demonstrate that the performance of traditional regression models can be surpassed despite training with cross entropy loss. Importantly, priming the same model with continuous properties yields a highly competitive conditional generative models that outperforms specialized approaches in a constrained property optimization benchmark. In sum, the Regression Transformer opens the door for ”swiss army knife” models that excel at both regression and conditional generation. This finds application particularly in property-driven, local exploration of the chemical or protein space.

1 INTRODUCTION

Transformers (Vaswani et al., 2017) are now ubiquitous in natural language processing (NLP) and have also enjoyed large success in molecular (Schwaller et al., 2019; 2021) and protein language modeling (Rives et al., 2021; Jumper et al., 2021). The invention of Transformers was the latest step in a steady decline of inductive biases in applied ML, a trend that started with the rise of deep learning: CNNs outperformed traditional feature descriptors in object recognition (Krizhevsky et al., 2012), self-attention generalized dense layers to learn sample-dependent instead of static affine transformations (Luong et al., 2015) and Transformers exploited self-attention to supersede RNNs as the de-facto standard in NLP. The success of vision transformers has questioned the need for translation equivariance in image processing (Ramachandran et al., 2019) and now, even frozen Transformers pretrained on text achieve SOTA results in object detection and protein classification (Lu et al., 2021). Given that Transformers are today’s most generic model[1], it is not surprising that entire domains have been formulated as sequence modeling problem to leverage their power (e.g., offline RL Chen et al. (2021)).

1.1 THE REGRESSION TRANSFORMER

A provocative next step toward reducing inductive biases might be to refrain from explicitly modeling target variables as functions of input variables. Instead of following this discriminative modelling approach when tuning task-specific language heads in Transformers, learning the joint distribution over input and target variables could effectively further blur the lines between predictive and conditional generative models. The feasibility of such approach can be assessed via permutation language modeling (PLM) (Yang et al., 2019).

Here, we reformulate regression as a sequence modeling task of textual tokens. We propose the Regression Transformer (RT), a class of models that can be trained on combinations of numerical and textual tokens (Figure 1). This circumvents the canonical way of solving regression with Transformers; i.e., a set of dense layers trained with a regression loss on the `[CLS]` token (Devlin et al., 2019). Despite solely relying on tokenization of numbers and cross-entropy loss, the RT can successfully solve regression tasks. Notably, the same model can conditionally generate text sequences given continuous properties. This is achieved simply by

[1]graph neural networks with multihead attention as neighborhood aggregation on complete graphs

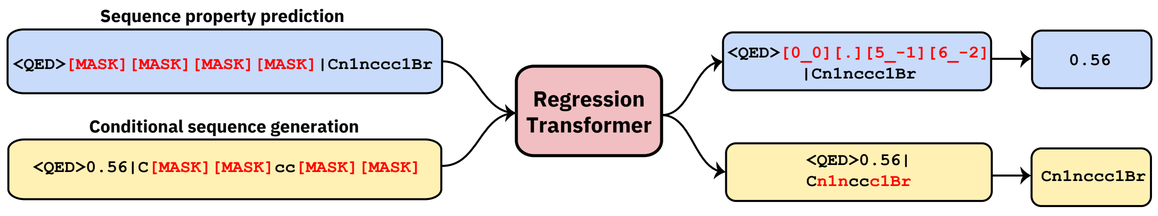

Sequence property prediction
<QED>[MASK][MASK][MASK][MASK]|Cn1nccc1Br
Conditional sequence generation
<QED>0.56|C[MASK][MASK]cc[MASK][MASK]
Regression Transformer
<QED>[0_0][.][5_-1][6_-2]|Cn1nccc1Br
0.56
<QED>0.56|Cn1nccc1Br
Cn1nccc1Br

Figure 1: **Overview of Regression Transformer (RT).** The RT is a XLNet-based model designed to handle combinations of text and numbers. Training the RT yields a dichotomous model that seamlessly switches between property prediction and conditional text generation. Dependent on the mask location, the same model either predicts numerical tokens given textual tokens, thus performing a regression task *(blue stream, top)*; or predicts textual tokens given both numerical and textual tokens, thus performing a property-driven conditional generation *(yellow stream, bottom)*. We demonstrate the strength of the RT in chemical and protein language modeling tasks.

moving the [MASK] location and thus does not require finetuning individual heads. To equip the RT with an inductive bias for handling continuous properties, numbers are first tokenized into a sequence of tokens preserving the decimal order. We then devise numerical encodings to inform the model about the semantic proximity of these tokens. To allow for concurrent optimization of regression and conditional generation, we propose an PLM-inspired training scheme that alternates between optimizing property prediction and text generation. For the latter, we derive a novel self-consistency loss that exploits the dichotomy of the RT by querying itself with the generated candidate sequence.

Related work and application domain. Models that concurrently excel at regression and conditional text generation are, besides theoretical considerations, of special interest for chemo- and bioinformatics. Text sequences are often labelled with continuous properties (e.g., drug efficacy or protein solubility) and design tasks are intertwined with biochemical properties. But despite the rise of deep generative models in molecular (Elton et al., 2019) and protein design (Wu et al., 2021), current approaches still develop property predictors and generative models independently. Transformer-based architectures have been used widely on chemical tasks but either focused on property prediction (Wang et al., 2019; Kim et al., 2021) or on conditional molecular design (Irwin et al., 2021; Mahmood et al., 2021), never on both. This semantic gap persists across architectural flavors (e.g., GANs (Méndez-Lucio et al., 2020), RL (Born et al., 2021) or VAEs (Gomez-Bombarelli et al., 2018)). To our knowledge, all existing approaches limit the communication between both modules to a reward/loss and thus fail to ”entangle” constrained structure generation with property prediction. By sharing weights between both tasks, the RT adheres to the intuitive expectation that a property-driven generative model should, in the first place, excel at recognizing this property. The RT aims to close this gap.

Experimental scope. In the remainder of this paper, we describe the Regression Transformer (RT). The RT is evaluated in predictive and generative tasks in chemical as well as protein language modeling. For chemical languages, we first validate the RT on a synthetic dataset of drug-likeness (Bickerton et al., 2012), and then test it on three property prediction datasets from the MoleculeNet benchmark (Wu et al., 2018) and one property-driven molecular generation benchmark (Jin et al., 2018). For protein sequence modeling, we apply the RT on a synthetic dataset of solubility (Boman, 2003) and two datasets from TAPE (Rao et al., 2019).

2 Methods

2.1 XLNet backbone

The RT is built upon an XLNet backbone (Yang et al., 2019) to retain the benefits of auto-regressive modeling in combination with a bidirectional context. The bidirectionality is critical because the RT is required to fill multiple tokens at arbitrary positions in a sequence while attending the full remaining sequence[2]. However, the independence assumption in bidirectional but non-autoregressive models like BERT becomes increasingly disruptive as more masked tokens are filled. In general, it is important to notice that the proposed framework can be applied to all transformer flavors, but it certainly benefits from an autoregressive generation with full sequence attention even for discontiguous mask locations, like XLNet or MPNet (Song et al., 2020).

[2]e.g. SMILES are non-local sequences such that masking functional groups usually implies masking disconnected tokens.

2.2 Tokenization

This section describes the processing of strings consisting of a mixture of numerical and textual symbols (cf. Appendix Figure A1, *top*). Representing every number as a single token is suboptimal due to a lack of generalization to new numbers and sparsity of the provided tokens. We therefore propose a scheme for converting text representing numbers into a sequences of tokens. First, the following regular expression splits a string denoting a numerical:

`\s*\s*?(\+|-)?(\d+)(\.)?(\d+)?\s*` (1)

Each of the resulting matches containing a number is converted to a token $t_{v,p}$ where $v \in \mathbb{N} \cap [0..9]$ is the value/digit and $p \in \mathbb{Z}$ is the decimal place (e.g., 12.3 is split into [`1_1`, `2_0`, `.`, `3_-1`]). We call these *numerical tokens*. This representation has the advantage that it allows easy decoding of the digit sequence but also distinguishes their decimal order by adhering to classic positional notation. Negative numbers are preceded with a special token. Regarding alphabetic tokens, we represent molecules either as SMILES (Weininger, 1988) or SELFIES (Krenn et al., 2020) strings and tokenize with the regular expression from Schwaller et al. (2018) and the internal SELFIES tokenizer, respectively. Protein sequences are tokenized per amino acid.

2.3 Numerical encodings (NE)

Due to the inherent structure of numbers, learning the embeddings of numerical tokens in a purely data-driven way might be ineffective. Moreover, since the RT is trained with cross-entropy loss, no notion of similarity between numerical tokens is conveyed. As a remedy, we propose numerical encodings (NE), a simple inductive bias about the semantic proximity of numerical tokens, similar to positional encodings (Vaswani et al., 2017). In practice, we sum the NEs with regular word embeddings and relative positional encodings from XLNet (see Appendix Figure A1 for a workflow). Our proposed numerical encodings are zero vectors for all but numerical tokens of the dictionary. We follow positional notation as above. Given a token $t_{v,p}$ (with digit value v and decimal place p), the numerical encoding at embedding dimension j is defined as:

$$NE_{Float}(v,p,j) = (-1)^j \cdot \frac{v \cdot 10^p}{j+1} \tag{2}$$

Thus, the amplitude of the NE scales with the numerical value of the token. The NEs are perfectly correlated among embedding dimensions but alternate between positive and negative values for even and odd dimensions and vanish for higher dimensions (see example in appendix Figure A2). Critically, the pairwise distances of the numerical encodings are symmetric and decay monotonically with the float value (see appendix Figure A2).

2.4 Training objectives

The input $\mathbf{x}$ for a RT is defined by a concatenation of k property tokens $[\mathbf{x}^p]_k$ and l textual tokens $[\mathbf{x}^t]_l$, such that: $\mathbf{x} = [\mathbf{x}^\mathbf{p}, \mathbf{x}^\mathbf{t}]_T = [x_1^p, ..., x_k^p, x_1^t, ..., x_l^t]_T$. $T{=}k{+}l$ is the full sequence length and $\mathbf{x}^p$ and $\mathbf{x}^t$ are property and textual tokens respectively.

Permutation language modeling (PLM) objective. The idea of PLM (Yang et al., 2019) is to fill masked tokens auto-regressively by sampling a factorization order $\mathbf{z}$ for a sequence $\mathbf{x}$ at runtime. Decomposing the likelihood $p_\theta(\mathbf{x})$ according to the facorization order yields, in expectation, a bidirectional auto-regressive model. Let $\mathbf{z} \in \mathcal{Z}_T$ denote one of the $T!$ permutations of our sequence $\mathbf{x}$. If z_i and $\mathbf{z}_{<i}$ are the i-th and first $i-1$ elements of $\mathbf{z}$, the PLM objective is:

$$\max_\theta \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_\mathcal{T}} \left[\sum_{i=1}^{T} \log p_\theta(x_{z_i} | \mathbf{x}_{\mathbf{z}_{<i}}) \right] \tag{3}$$

In practice, partial prediction is performed, i.e., only the last c tokens of the factorization order $\mathbf{z}$ are predicted. Following XLNet, $\mathbf{z}$ is split into a (masked) target subsequence $\mathbf{z}_{>c}$ and an unmasked input sequence $\mathbf{z}_{\leq c}$ s.t. the objective becomes

$$\mathcal{J}_{PLM} = \max_\theta \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_\mathcal{T}} \left[\log p_\theta(\mathbf{x}_{\mathbf{z}>c} | \mathbf{x}_{\mathbf{z}_{\leq c}}) \right] = \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_\mathcal{T}} \left[\sum_{i=c+1}^{T} \log p_\theta(x_{z_i} | \mathbf{x}_{\mathbf{z}_{<i}}) \right] \tag{4}$$

where c is a hyperparamter, usually sampled per batch such that roughly $1/c$ of the tokens are masked. We notice that (4) does not make any specific choices on $\mathbf{x}^p$ and $\mathbf{x}^t$. It thus constitutes our baseline objective.

While (4) is a generic objective, it is computationally exhaustive to optimize due to the permutations. Moreover it is not ideal for our needs because it does not distinguish between textual and property tokens. Instead, we are aiming to develop a single model that can either predict numerical tokens (when given text sequences) or text tokens (when given a combination of numerical and text tokens). To that end, we propose to train on two alternating objectives, one designed for property prediction and one for text generation.

Property prediction objective. Instead of randomizing which tokens are masked, this objective exclusively masks all the property tokens. Specifically, we constrain the factorization order $\mathbf{z}$ by setting the first l elements to $\mathbf{x^t}$ and fixing $c = l$. This guarantees that only property tokens are masked. Let $\mathcal{Z}_T^p$ denote the set of possible permutations. Under this constraint, then the objective becomes

$$\mathcal{J}_P = \max_\theta \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_T^p}\left[\log p_\theta(\mathbf{x}^p|\mathbf{x}^t)\right] = \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_T^p}\left[\sum_{i=c+1}^{T}\log p_\theta(x_{z_i}^p|\mathbf{x}_{\mathbf{z}_{\leq c}}^t, \mathbf{x}_{\mathbf{z}_{>c<i}}^p)\right] \tag{5}$$

where $\mathbf{x}_{\mathbf{z}_{>c<i}}^p$ denotes the c-th to the $i-1$-th element of the factorization order $\mathbf{z}$. We emphasize that this "tailored" property objective $\mathcal{J}_p$ is still optimized with a cross-entropy loss in practice. Note that this loss cannot convey any notion on the qualitative proximity of the prediction to the labels because the level of measurement of tokens in a language model are on a nominal level. Instead, a traditional regression loss operates on a ratio scale[3].

Conditional text generation objective. This objective facilitates the generation of textual tokens given a property primer and textual tokens. We constrain the factorization order $\mathbf{z}$ by setting the first k elements to $\mathbf{x}^p$ to and sampling the cutoff c, s.t. $c >= k$. This ensures that masking only occurs on textual tokens. Then, we denote the set of permutations by $\mathcal{Z}_T^t$ and the objective becomes

$$\mathcal{J}_G = \max_\theta \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_T^t}\left[\log p_\theta(\mathbf{x}_{\mathbf{z}_{>c}}^t|\mathbf{x}_{\mathbf{z}_{\leq k}}^p, \mathbf{x}_{\mathbf{z}_{>k<c}}^t)\right] = \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_T^t}\left[\sum_{i=c+1}^{T}\log p_\theta(x_{z_i}^t|\mathbf{x}_{\mathbf{z}_{\leq k}}^p, \mathbf{x}_{\mathbf{z}_{>k<i}}^t)\right] \tag{6}$$

Intuitively, this objective applies regular PLM while sparing the numerical tokens. It then aims to reconstruct the full text sequence (i.e., molecule) given the uncorrupted property tokens and partially corrupted textual tokens.

Self-consistency (SC) objective. Standalone, the CT generation objective (6) does not reward if the generated sequences adhere to the primed property. This is critical because similar to natural language, changes in single tokens (atoms or amino acids) can drastically change the property (meaning) of a sequence (sentence). As a remedy, we extended the text generation objective $\mathcal{J}_G$ by a self-consistency term that exploits the dichotomy of the Regression Transformer. The full objective is given by:

$$\mathcal{J}_{SC} = \mathcal{J}_G(\mathbf{x}) + \alpha \cdot \mathcal{J}_P(\hat{\mathbf{x}}) \tag{7}$$

where the second addend is the self-consistency term, weighted by a factor α. Intuitively, it is given by the difference between the property of the sample and the predicted property of the generated sample $\hat{\mathbf{x}}$. Here, $\hat{\mathbf{x}}$ is obtained by greedy decoding of the masked tokens and combining it with the non-corrupted tokens of $\mathbf{x}$. To be precise, $\hat{\mathbf{x}} = [\mathbf{x}^p, \hat{\mathbf{x}}^t]$ where $\hat{\mathbf{x}}^t = [m_1\bar{x}_1 + (1-m_i)x_1, ..., m_l\bar{x}_l + (1-m_l)x_l]$. Here, $\mathbf{m}$ is an indicator vector whether masking occurred at a given position and $\bar{\mathbf{x}} = \arg\max \sum_{i=c+1}^{T} \log p_\theta(x_{z_i}^t|\mathbf{x}_{\mathbf{z}_{<k}}^p, \mathbf{x}_{\mathbf{z}_{>k<i}}^t)$ is the result of greedy decoding. In such a formulation, the RT acts as an oracle during its own optimization, resembling an additional layer of self-supervision. While this scheme risks undesired side effects when the model performs poorly at property prediction, it introduces a notion of self-consistency and rewards the generation of molecules that are different from training samples as long as they adhere to the property.

Evaluation & performance metrics. Regression (or property prediction) is evaluated by RMSE and Pearson's $\mathbf{r}$ (PCC) using the floating value resulting from converting the sequence of predicted tokens into digits (the model never failed to predict a token sequence not corresponding to a valid numerical). For conditional generation, we randomly mask tokens of the sequence and then query the model with ten equidistant property primers spanning the range of property values. As a sanity check, we report **0-Var**, i.e., the percentage of molecules for which the generation was unaffected by the primer (the lower the better). The main metric is the average Spearman's ρ between the ten primers and the actual properties. Spearman is favorable over Pearson because it is only rank-sensitive. Due to constraints induced by the fragmented sequence, covering the entire property spectrum is usually impossible such that e.g., RMSE is inappropriate for this task (e.g., priming a highly toxic scaffold with low toxicity cannot yield a non-toxic molecule). Details on the training procedure are in appendix A6.

2.5 Datasets

2.5.1 Chemical language modeling

Synthetic QED dataset. Starting from $\sim$ 1.6M bioactive molecules from ChEMBL (Mendez et al., 2019), we created a synthetic dataset by computing the QED (Bickerton et al., 2012) score (q $\in$ [0, 1]) for all molecules

[3]E.g., predicting a sequence of numerical tokens corresponding to a property score of 0.91 for a sample with a true property of 0.11 will not generally result in a higher loss than predicting 0.21.

with `RDKit` and rounded to 3 decimal places. We used $\sim$ 1.4M molecules for training, 1k for validation and 10k for testing.

MoleculeNet datasets. We focused on 3 regression datasets from the MoleculeNet benchmark (Wu et al., 2018): *ESOL*, *FreeSolv* and *Lipophilicity*, where the task is to predict water solubility, hydration free energy and lipophilicity of a molecule, respectively. For each dataset, we performed 3 random splits (as recommended by Wu et al. (2018)) with 15% validation data. Because the datasets are small (< 5000 samples), we used SMILES augmentation (Bjerrum, 2017) with a factor of 16.

Property-optimization benchmark. This is a benchmark for property-driven, conditional molecular generation. The goal is to adapt a seed molecule such that a property is maximized while adhering to a fixed similarity constraint. We obtained the data from Jin et al. (2018) which ships with a fixed split of 215,381 training and 799 test molecules and their penalized LogP (pLogP) value (Kusner et al., 2017). pLogP is the octanol-water partition coefficient (logP) penalized by the synthetic accessibility score and the number of cycles with > 6 atoms. Hence, pLogP just like QED can be computed deterministically from the molecule. To maximize comparability we followed the candidate assembly process of Jin et al. (2018), described in appendix A6.1.3.

2.5.2 Protein sequence language modeling

Synthetic Boman dataset. As a large-scale, labelled dataset we focused on the Boman index, a measure of potential protein interaction for peptides. It is the average of the solubility values of the residues (Boman, 2003). We collected all 2,648,205 peptides with 15 to 45 AAs from UniProt (Consortium, 2020), computed their Boman index, and used 10k and 1k for testing and validation respectively.

TAPE datasets. We focused on two datasets from the TAPE benchmark (Rao et al., 2019): *Fluorescence* (Sarkisyan et al., 2016) and *Stability* (Rocklin et al., 2017). The goal is to predict, respectively, the fluorescence and intrinsic folding stability of a protein that is one to four mutations away from a training protein. Both datasets ship with fixed splits. The fluorescence (stability) dataset has 21,446 (53,416) training, 5,362 (2,512) validation and 27,217 (12,851) test samples.

3 Results on chemical language modeling

3.1 Learning drug-likeness (QED dataset)

To test the feasibility of concurrent property prediction and conditional generation, we start with optimizing the vanilla PLM objective (Equation 3) on the QED dataset (see Figure A1 for a workflow illustration). Evaluating these models on property prediction (i.e., masking only numerical tokens) does not closely mimic the training dynamics. Despite this, as well as the unconventional formulation of a regression task as sequence modeling, all models generated sequences of numerical tokens that allowed decoding floats, and even achieved a RMSE < 0.06 (cf. Table 1). Instead, for the generative task, the same models were queried 10 times for every

Table 1: **Performance after PLM training.** RMSE (↓) and PCC refer to predicting QED, perplexity (↓) to the PLM objective (Equation (4)) and Spearman ρ (↑) and 0-Var (↓) to the conditional generation task. For RMSE and PCC, the mean across masking one to three numerical tokens is shown and for perplexity, 0-Var and ρ the mean across five evaluations. Full table with standard deviations in appendix Table A1.

Configuration			*Regression task*		*Generation task*	
Data	NE	Perpl.	RMSE	PCC (↑)	0-Var (↓)	ρ (↑)
SMILES	–	**1.55**	0.0549	**0.972**	1.6%	0.096
SELFIES	–	1.61	0.0591	0.968	0.9%	0.427
SELFIES	✓	1.59	**0.0547**	0.971	**0.3%**	**0.467**

seed molecule, with property primers equidistantly spaced in $[0, 1]$ and 40% of masked textual tokens. The high Spearman ρ values show that the model learned successfully to complete the corrupted scaffolds to produce full molecules with a desired QED. Here, the SELFIES models exceeded the SMILES models by far, because SMILES, unlike SELFIES, can be syntactically invalid. Notably, the novelty score (i.e., percentage of conditionally generated molecules not present in training data) was $> 99\%$ for all models. This demonstrates that the RT can generate novel chemical matter that adheres to a continuous property of interest. The richness of the generated molecules based on a seed molecule is exemplary visualized in Figure 2 (*top*). The slightly superior results in perplexity and regression for SELFIES can be explained respectively by an easier syntax and are aligned with previous findings (Markert et al., 2020). For the rest of the experiments, we focus only on SELFIES. Notably, the numerical encodings (NE) slightly improved performance in all tasks.

Based on our proposed training scheme with alternating objectives, the models were further refined: For every model in Table 1, two models were trained, *without* ($\alpha = 0$) and *with* ($\alpha = 1$) the self-consistency term in the text loss, respectively. Notably, the performance in regression as well as conditional generation

Table 2: **Performance evaluation on refined objectives.** Legend like Table 1. All models used SELFIES. NE means numerical encodings and α refers to the loss function in Equation 7. Full table with standard deviations in appendix Table A2. MAE stands for mean absolute error.

Config		*Regression task*		*Generation task*	
NE	α	RMSE	PCC	0-Var	Spearman ρ
✗	0	**0.0341**	**0.988**	0.2%	0.47
✗	1	0.0483	0.978	0.3%	0.49
✓	0	0.0498	0.982	0.3%	0.47
✓	1	0.0367	0.987	**0.2%**	**0.52**

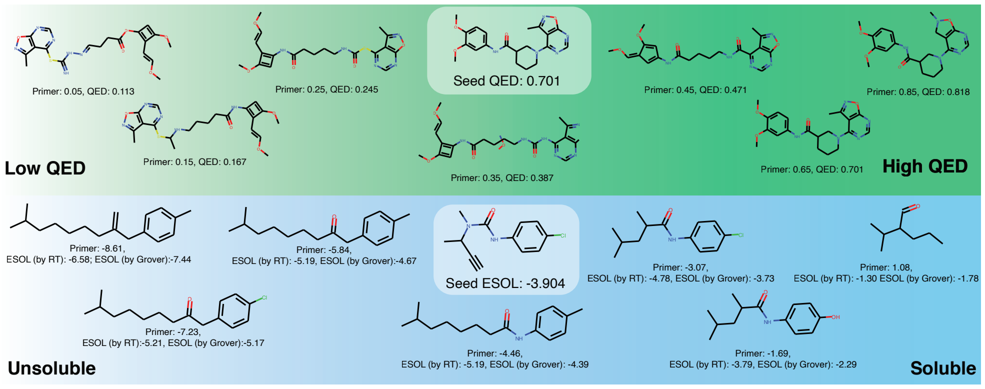
(a) Performance evaluation on refined objectives.

Model	MAE (↓)
k-NN (baseline)	0.054
SMILES-BERT (Kim et al., 2021)	0.020
RT - PLM obj.	0.035
RT - Alternating obj. ($\alpha = 0$)	**0.017**

(b) Performance comparison in predicting QED.

improved significantly, demonstrating the effectiveness of the refined objectives. Moreover, all configurations of the RT outperformed a baseline k-NN-regression model (RMSE of 0.083, PCC of 0.927 with $k = 30$). In addition, our best configuration even surpassed the SMILES-BERT model which achieved a MAE of 0.02 after large-scale pretraining and finetuning a regular regression head (see Table 2b).

Moreover, the self-consistency term further improved the model's ability to generate tailored ensembles of molecules and led to consistently higher correlation scores. However, this comes at the cost of inferior regression performance compared to the other models ($\alpha = 0$). Presumably, this is because the model weights in charge of the regression are confounded with the gradients from the self-evaluation (cf. Equation 7). The novelty scores for the molecules generated in this setting were even slightly higher than for the PLM training ($> 99.3\%$ for all models). A particularly challenging application for property-driven, local exploration of the chemical space is scaffold hopping; for an example on this see appendix A8.1. Again, for ablation studies on SMILES language and other types of NEs, see appendix A7.1.1. Our extended results show that the inductive bias of the NEs is not strictly necessary: In the absence of NEs, a large fraction of the learned embeddings of the numerical tokens directly and significantly encode the natural ordering of digits (see appendix A9.1).



Primer: 0.05, QED: 0.113
Primer: 0.25, QED: 0.245
Seed QED: 0.701
Primer: 0.45, QED: 0.471
Primer: 0.85, QED: 0.818
Primer: 0.15, QED: 0.167
Primer: 0.35, QED: 0.387
Primer: 0.65, QED: 0.701
Low QED
High QED
Primer: -8.61, ESOL (by RT): -6.58; ESOL (by Grover):-7.44
Primer: -5.84, ESOL (by RT): -5.19, ESOL (by Grover):-4.67
Seed ESOL: -3.904
Primer: -3.07, ESOL (by RT): -4.78, ESOL (by Grover): -3.73
Primer: 1.08, ESOL (by RT): -1.30 ESOL (by Grover): -1.78
Primer: -7.23, ESOL (by RT):-5.21, ESOL (by Grover):-5.17
Primer: -4.46, ESOL (by RT): -5.19, ESOL (by Grover): -4.39
Primer: -1.69, ESOL (by RT): -3.79, ESOL (by Grover): -2.29
Unsoluble
Soluble

Figure 2: **Property-driven, local optimization of molecular design with the Regression Transformer (RT).** For each row, the seed molecule is shown in the middle alongside its true property. Based on 10 property primers, 10 molecules were decoded but duplicates were discarded. Samples generated with the self-consistency model. *Top:* QED dataset. *Bottom:* ESOL dataset of aquatic solubility. The solubility of the novel molecules was predicted by the RT itself and is externally validated by Grover (Rong et al., 2020).

3.2 Regression benchmark (MoleculeNet)

Next, we finetuned our models from the QED dataset on the three MoleculeNet datasets (Wu et al., 2018). The regression performance on ESOL, FreeSolv and Lipophilicity is shown in Table 3 and compared to prior work. Due to countless research on the MoleculeNet benchmark, we only report performances of the original MoleculeNet release (Baselines: RF, XGBoost Advanced: MPNN) and the best comparable Transformer-based approaches: SMILES-BERT (Wang et al., 2019) and Mol-BERT (Fabian et al., 2020). For direct comparability to the RT, we also report the results of the XLNet backbone trained with a conventional regression loss. Note that, from all the models in Table 3, only the RT is trained with a classification loss. Interestingly, XGBoost, the strongest baseline by Wu et al. (2018), is outperformed by all our models on all tasks. Even the MPNN (Gilmer et al., 2017), a message-passing GNN, is slightly surpassed on FreeSolv and Lipophicility by some of our models. Remarkably, the RT is on par (Lipophilictiy dataset) or mildly inferior (i.e., within standard-deviation

Table 3: **RMSE (↓) in predicting MoleculeNet dataset properties.** Performance on three different datasets across predictive models. All models used random splits.

Model	Objective	NE	α	ESOL	FreeSolv	Lipo.
RF	Regression	–	–	$1.16_{\pm 0.15}$	$2.12_{\pm 0.68}$	$0.78_{\pm 0.02}$
XGBoost	Regression	–	–	$1.05_{\pm 0.10}$	$1.76_{\pm 0.21}$	$0.84_{\pm 0.03}$
MPNN	Regression	–	–	$0.55_{\pm 0.02}$	$1.20_{\pm 0.02}$	$0.76_{\pm 0.03}$
SMILES-BERT	Regression	–	–	$\mathbf{0.47}_{\pm 0.05}$	$\mathbf{0.81}_{\pm 0.09}$	–
Mol-BERT	Regression	–	–	$0.53_{\pm 0.04}$	$0.95_{\pm 0.33}$	$\mathbf{0.56}_{\pm 0.03}$
XLNet (ours)	Regression	–	–	$0.69_{\pm 0.01}$	$1.03_{\pm 0.25}$	$0.74_{\pm 0.02}$
RT (ours)	Classification	✗	0	$0.76_{\pm 0.05}$	$1.19_{\pm 0.29}$	$0.76_{\pm 0.03}$
RT (ours)	Classification	✗	1	$0.75_{\pm 0.04}$	$1.32_{\pm 0.39}$	$0.76_{\pm 0.03}$
RT (ours)	Classification	✓	0	$0.71_{\pm 0.04}$	$1.40_{\pm 0.47}$	$0.74_{\pm 0.05}$
RT (ours)	Classification	✓	1	$0.73_{\pm 0.04}$	$1.34_{\pm 0.29}$	$0.74_{\pm 0.03}$

range; ESOL & FreeSolv datasets) to a XLNet model trained with a regression loss. However, all our models are outperformed by BERT-based approaches that leveraged pretraining and tuned a regression head; most likely due to the massive unsupervised pretraining.

Table 4: **Conditional generation for MoleculeNet datasets.** Average performances across three splits for training with alternating objectives. ρ refers to Spearman ρ and was evaluated with Grover (Rong et al., 2020). Same legend like Table 3. Full table with standard deviations and self-evaluation with RT are in appendix Table A4.

Model	NE	α	ESOL		FreeSolv		Lipophilicity	
			0-Var	ρ	0-Var	ρ	0-Var	ρ
RT	✗	0	**4.4%**	0.44	7.9%	0.53	3.6%	0.29
RT	✗	1	5.9%	0.46	7.5%	0.56	**2.7%**	**0.35**
RT	✓	0	6.1%	0.46	8.9%	**0.57**	4.2%	0.29
RT	✓	1	6.1%	**0.47**	**6.5%**	**0.57**	**2.7%**	0.34
X-BERT			*Task unfeasible*					

But in stark contrast to all those approaches, only the RT can also be used to conditionally *generate* molecules similar to the training samples (cf. Table 4). Since the properties of generated molecules are intractable to evaluate *in-silico*, we could predict them, handily, using the RT. However, as this might be a biased estimator, we evaluated them using Grover (Rong et al., 2020), a self-supervised Graph Transformer. Hence, the Spearman reported in Table 4 is based on Grover predictions. Corroborative for our work was the high correlation of our property predictions (RT) with Grover's for molecules generated by the ESOL, FreeSolv and Lipo models (0.86, 0.84 and 0.75 respectively). Thus, the Spearman ρ scores obtained with RT predictions are consistent to Grover (cf. Table A4). In terms of molecular generation, the high Spearman scores are satisfying. The self-consistency loss clearly had a positive effect. For a qualitative evaluation, we depict the generations for one exemplary seed molecule of the solubility dataset in Figure 2 (*bottom*).

3.3 Conditional molecular generation benchmark

To assess whether the RT is a powerful conditional generative model, we benchmarked it on a property-driven molecular generation task, namely pLogP constrained optimization (Jin et al., 2018). Given a seed molecule and a similarity constraint δ (i.e., Tanimoto similarity), the goal is to generate molecules with higher pLogP values. We trained the RT using alternating objectives ($\alpha = 1$) and subsequently evaluated the model on the test molecules, masking 10%-60% of the molecular tokens with ten property primers (spaced in the upper half of pLogP values) until $K = 80$ molecules were obtained per seed (this follows the procedure suggested by Jin et al. (2018)). The results in Table 5 demonstrate that, for both similarity thresholds δ, the RT obtained the best results and significantly outperforms the JT-VAE and GCPN models by 614% and 103% in average improvement, respectively. Experiments on further values of δ ($\delta = 0.0$ and $\delta = 0.2$) confirm this trend (see A7.3). While the success rate of GCPN is higher than ours, we emphasize that both, JT-VAE apply optimization at *inference time*. The JT-VAE applies gradient ascent in the latent space and the GCPN performs a stepwise reward optimization. Instead, the RT does not only not require any optimization at this stage, but it was also never trained explicitly to produce molecules with *high* pLogP. This finding demonstrates that the RT is able to compete with specialized conditional generative models in goal-directed molecular generation. At the same time, the RT also predicted the pLogP value with a Pearson's correlation of 0.92, a task that cannot be addressed with normal conditional generative models. The results in Table 5 were obtained with the RT including a self-consistency loss, but for ablation studies on the RT and further results on $\delta = 0.2$ and $\delta = 0$, see appendix A7.3.

Table 5: **Constrained property optimization benchmark.** JT-VAE is from Jin et al. (2018) and GCPN from You et al. (2018). α denotes the minimal similarity to the seed.

	Similarity threshold $\delta = 0.4$				**Similarity threshold $\delta = 0.6$**			
	Generation task			*Regression*	*Generation task*			*Regression*
Model	Improvem.	Similarity δ	Success	PCC	Improvem.	Similarity δ	Success	PCC
JT-VAE	$0.84_{\pm 1.5}$	$0.51_{\pm 0.1}$	83.6%	*Unfeasible*	$0.21_{\pm 0.7}$	**0.69**$_{\pm 0.0}$	46.4%	*Unfeasible*
GCPN	$2.49_{\pm 1.3}$	$0.47_{\pm 0.1}$	**100%**	*Unfeasible*	$0.79_{\pm 0.6}$	$0.68_{\pm 0.1}$	**100%**	*Unfeasible*
RT (Ours)	**3.16**$_{\pm 1.5}$	**0.54**$_{\pm 0.1}$	97.1%	**0.92**$_{\pm 0.0}$	**2.21**$_{\pm 1.3}$	**0.69**$_{\pm 0.1}$	81.8%	**0.92**$_{\pm 0.0}$

4 Protein sequence language modeling

To assess the generality of the RT beyond chemical languages, we performed several experiments on protein language modeling. Unless indicated otherwise, all following results refer to the alternating training scheme with self-consistency objective and numerical encodings.

Protein interaction dataset (Boman index) The initial experiments on protein sequence modeling, conducted on the synthetic Boman dataset are shown in Table 6. The RT obtained nearly perfect results in predicting Boman's index (Spearman $\rho > 0.994$). It outperformed a baseline k-NN model with Levenshtein

Table 6: **Protein language modeling results.** The regression task was evaluated by Spearman's ρ (↑). The evaluation metrics for protein generation are identical to those for molecular generation (see above). TAPE datasets/performances taken from Rao et al. (2019).

		Boman dataset			Stability dataset			Fluorescence
		Regression	*Generation*		*Regression*	*Generation*		*Regression*
Model	Source	ρ (↑)	0-Var (↓)	ρ (↑)	ρ (↑)	0-Var (↓)	ρ (↑)	ρ (↑)
k-NN (baseline)	Ours	0.93			0.21			0.59
One-Hot	TAPE	–			0.19			0.14
Pretr. LSTM	TAPE	–	*Task unfeasible*		0.69	*Task unfeasible*		0.67
Pretr. Transformer	TAPE	–			**0.73**			0.68
Alley et al. (2019)	UniRep	–			**0.73**			0.67
RT	Ours	**0.99**	$0.2\%_{\pm 0.0}$	$0.84_{\pm 0.0}$	$0.71_{\pm 0.02}$	$19\%_{\pm 4.5}$	$0.44_{\pm 0.01}$	**0.72**$_{\pm 0.04}$

distance (Levenshtein, 1966) as similarity measure, which is a meaningful baseline because the Boman index solely depends on the frequencies of amino acids. At the same time, the RT very successfully generated peptides with a desired Boman index, given a partially corrupted AA sequence (cf. Spearman ρ of 0.84, see Table 6). An ablation study on the three loss functions (Equations 3, 6 and 7) confirmed the superiority of the self-consistency objective (see appendix A7.4.1).

TAPE datasets (Fluorescence & Stability) Next, comparing the performance of the RT on two protein regression datasets from TAPE yields very competitive results (cf. Table 6). This is remarkable given that the TAPE models were pretrained large-scale on unlabelled protein sequences and finetuned for these downstream tasks with a regression loss. For example, the RT outperforms all reported SOTA methods in Spearman ρ on the Fluorescence task; which has a distribution with two modes, for bright and dark proteins respectively. Inspecting the predictions in more depth showed that the RT excels at recognizing the mode of a protein but struggles with intra-mode precision (see appendix A9.3). This is not surprising given the training on a classification loss. Overall, the comparable predictive performance of the RT demonstrates that the benefits of self-supervised pretraining can be maintained, and even extended to numerically labelled datasets; which yields, *en passant*, a conditional generative model for property-driven local exploration of the protein sequence space. Evidence on this can be found in Table 6: Whereas all TAPE models as well as the UniRep method are incapable of addressing this generation task, the RT was able to modify the test proteins such that their (predicted) stability correlated strongly with the primed property ($\rho = 0.44$).

5 Conclusion

Here, we have presented the Regression Transformer (RT) and demonstrated that regression can be casted as conditional sequence learning task. We started our experiments with demonstrating the general feasibility of our idea on synthetic datasets in both chemical and protein language modeling. Afterwards, focusing on existing property prediction benchmarks, we find that the RT learns continuous properties even from small datasets, surpasses conventional regression models (like XGBoost or k-NN) and can compete with (and sometimes even outperform) SOTA Transformers trained with a regression loss. Remarkably, this is achieved without ever providing ratio-scale information about the property which opens the door toward "swiss army knife" transformers. Indeed, our experiments on conditional text generation demonstrate the dichotomy of the Regression Transformer: The RT can conditionally generate novel molecules and proteins that seemingly adhere to the primed property values. This could be useful for property-driven, structure constrained molecular or protein design. In fact, our experiments on the constrained molecular generation benchmark have shown that the RT can surpass prolific previous approaches in this task. Finally, our work is line with the recent trend towards multitask Transformers (Sanh et al., 2021; Lu et al., 2021) and we hope it paves the way toward foundation models in material design.

Software and Data

The codebase to facilitate reproduction of all experiments is publicly available at: `https://github.com/IBM/regression-transformer`

References

Ethan C Alley, Grigory Khimulya, Surojit Biswas, Mohammed AlQuraishi, and George M Church. Unified rational protein engineering with sequence-based deep representation learning. *Nature methods*, 16(12): 1315–1322, 2019.

He Bai, Peng Shi, Jimmy Lin, Yuqing Xie, Luchen Tan, Kun Xiong, Wen Gao, and Ming Li. Segatron: Segment-aware transformer for language modeling and understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 12526–12534, 2021.

J Bajorath. Computational scaffold hopping: cornerstone for the future of drug design? *Future Medicinal Chemistry*, 9(7):629–631, 2017.

G Richard Bickerton, Gaia V Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L Hopkins. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2):90, 2012.

Esben Jannik Bjerrum. Smiles enumeration as data augmentation for neural network modeling of molecules. *arXiv preprint arXiv:1703.07076*, 2017.

HG Boman. Antibacterial peptides: basic facts and emerging concepts. *Journal of internal medicine*, 254(3): 197–215, 2003.

Jannis Born, Matteo Manica, Joris Cadow, Greta Markert, Nil Adell Mill, Modestas Filipavicius, Nikita Janakarajan, Antonio Cardinale, Teodoro Laino, and María Rodríguez Martínez. Data-driven molecular design for discovery and synthesis of novel ligands: a case study on SARS-CoV-2. *Machine Learning: Science and Technology*, 2(2):025024, 2021. ISSN 2632-2153. doi: 10.1088/2632-2153/abe808.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345 (To appear at NeurIPS 2021)*, 2021.

The UniProt Consortium. UniProt: the universal protein knowledgebase in 2021. *Nucleic Acids Research*, 49 (D1):D480–D489, 11 2020. ISSN 0305-1048. doi: 10.1093/nar/gkaa1100. URL `https://doi.org/10.1093/nar/gkaa1100`.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 4171–4186. ACL, June 2019.

Daniel C Elton, Zois Boukouvalas, Mark D Fuge, and Peter W Chung. Deep learning for molecular design—a review of the state of the art. *Molecular Systems Design &amp; Engineering*, 4(4):828–849, 2019.

Benedek Fabian, Thomas Edlich, Héléna Gaspar, Marwin Segler, Joshua Meyers, Marco Fiscato, and Mohamed Ahmed. Molecular representation learning with language models and domain-relevant auxiliary tasks. *arXiv preprint arXiv:2011.13230*, 2020.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Rafael Gomez-Bombarelli, Jennifer N Wei, David Duvenaud, Jose Miguel Hernandez-Lobato, et al. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2): 268–276, 2018.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: decoding-enhanced bert with disentangled attention. In *9th International Conference on Learning Representations, ICLR 2021*, 2021.

Ross Irwin, Spyridon Dimitriadis, Jiazhen He, and Esben Jannik Bjerrum. Chemformer: A pre-trained transformer for computational chemistry. *Machine Learning: Science and Technology*, 2021.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pp. 2323–2332. PMLR, 2018.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, pp. 1–11, 2021.

Hyunseob Kim, Jeongcheol Lee, Sunil Ahn, and Jongsuk Ruth Lee. A merged molecular representation learning for molecular properties prediction with a web-based service. *Scientific Reports*, 11(1):1–9, 2021.

Mario Krenn, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik. Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 1(4):045024, nov 2020. doi: 10.1088/2632-2153/aba947.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1945–1954. JMLR. org, 2017.

Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics doklady*, 10(8):707–710, 1966.

Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Pretrained transformers as universal computation engines. *arXiv preprint arXiv:2103.05247*, 2021.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421, 2015.

Omar Mahmood, Elman Mansimov, Richard Bonneau, and Kyunghyun Cho. Masked graph modeling for molecule generation. *Nature communications*, 12(1):1–12, 2021.

Greta Markert, Jannis Born, Matteo Manica, Gisbert Schneider, and Maria Rodriguez Martinez. Chemical representation learning for toxicity prediction. *PharML Workshop at ECML-PKDD*, 2020.

David Mendez, Anna Gaulton, A Patrícia Bento, Jon Chambers, Marleen De Veij, Eloy Félix, María Paula Magariños, Juan F Mosquera, Prudence Mutowo, Michał Nowotka, et al. Chembl: towards direct deposition of bioassay data. *Nucleic acids research*, 47(D1):D930–D940, 2019.

Oscar Méndez-Lucio, Benoit Baillif, Djork-Arné Clevert, David Rouquié, and Joerg Wichard. De novo generation of hit-like molecules from gene expression signatures using artificial intelligence. *Nature communications*, 11(1):1–10, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. *Advances in Neural Information Processing Systems*, 32, 2019.

Roshan Rao, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Xi Chen, John Canny, Pieter Abbeel, and Yun S Song. Evaluating protein transfer learning with tape. In *Advances in Neural Information Processing Systems*, pp. 9686–9698, 2019.

Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C Lawrence Zitnick, Jerry Ma, et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15), 2021.

Gabriel J Rocklin, Tamuka M Chidyausiku, Inna Goreshnik, Alex Ford, Scott Houliston, Alexander Lemak, Lauren Carter, Rashmi Ravichandran, Vikram K Mulligan, Aaron Chevalier, et al. Global analysis of protein folding using massively parallel design, synthesis, and testing. *Science*, 357(6347):168–175, 2017.

David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.

Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. Self-supervised graph transformer on large-scale molecular data. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33*, 2020.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*, 2021.

Karen S Sarkisyan, Dmitry A Bolotin, Margarita V Meer, Dinara R Usmanova, Alexander S Mishin, George V Sharonov, Dmitry N Ivankov, Nina G Bozhanova, Mikhail S Baranov, Onuralp Soylemez, et al. Local fitness landscape of the green fluorescent protein. *Nature*, 533(7603):397, 2016.

Philippe Schwaller, Theophile Gaudin, David Lanyi, Costas Bekas, and Teodoro Laino. "found in translation": predicting outcomes of complex organic chemistry reactions using neural sequence-to-sequence models. *Chemical science*, 9(28):6091–6098, 2018.

Philippe Schwaller, Teodoro Laino, Théophile Gaudin, Peter Bolgar, Christopher A Hunter, Costas Bekas, and Alpha A Lee. Molecular transformer: A model for uncertainty-calibrated chemical reaction prediction. *ACS central science*, 5(9):1572–1583, 2019.

Philippe Schwaller, Benjamin Hoover, Jean-Louis Reymond, Hendrik Strobelt, and Teodoro Laino. Extraction of organic chemistry grammar from unsupervised learning of chemical reactions. *Science Advances*, 7(15): eabe4166, 2021.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. In *Advances in Neural Information Processing Systems 33*, 2020.

Taffee T Tanimoto. Elementary mathematical theory of classification and prediction. *International Business Machines Corp.*, 1958.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, et al. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

Jesse Vig. A multiscale visualization of attention in the transformer model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 37–42, 2019.

Sheng Wang, Yuzhi Guo, Yuhong Wang, Hongmao Sun, and Junzhou Huang. Smiles-bert: large scale unsupervised pre-training for molecular property prediction. In *Proceedings of the 10th ACM international conference on bioinformatics, computational biology and health informatics*, pp. 429–436, 2019.

Yu-An Wang and Yun-Nung Chen. What do position embeddings learn? an empirical study of pre-trained language model positional encoding. In *EMNLP*, pp. 6840–6849, 2020.

David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

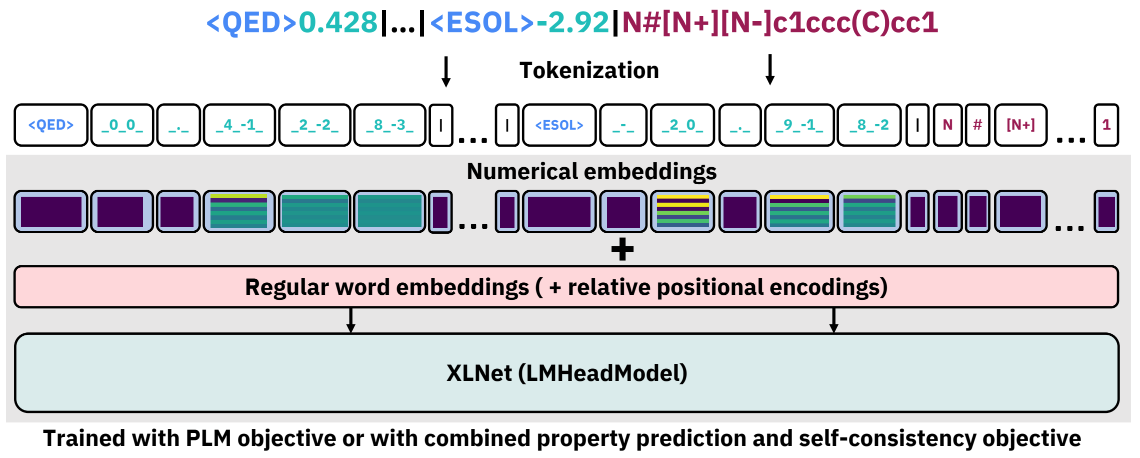
Zachary Wu, Kadina E Johnston, Frances H Arnold, and Kevin K Yang. Protein sequence design with deep generative models. *Current Opinion in Chemical Biology*, 65:18–27, 2021.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9 (2):513–530, 2018.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In *Advances in Neural Information Processing Systems*, pp. 6412–6422, 2018.

Appendix



<QED>0.428|...|<ESOL>-2.92|N#[N+][N-]c1ccc(C)cc1
Tokenization
<QED> _0_0_ _._ _4_-1_ _2_-2_ _8_-3_ | ... | <ESOL> _-_ _2_0_ _._ _9_-1_ _8_-2 | N # [N+] ... 1
Numerical embeddings
Regular word embeddings (+ relative positional encodings)
XLNet (LMHeadModel)
Trained with PLM objective or with combined property prediction and self-consistency objective

Figure A1: **Workflow of the Regression Transformer (RT) model.** Based on the XLNet backbone, the RT is a dichotomous model designed to handle combinations of text and numbers. *Top:* An input sequence consisting of a molecular string (red) and two property tags (blue), each associated to a floating value (green). Numbers are tokenized into a sequence of tokens that preserve the decimal order of each character. *Middle*: We propose numerical encodings that inform the model about the semantic proximity of these tokens and naturally integrate with relative positional encodings and classical learned embeddings. *Bottom:* The RT is trained with an alternating training scheme, derived from the PLM objective (Yang et al., 2019) and designed to concurrently optimize property prediction and conditional generation (*bottom*). The dots indicate that the RT naturally scales to multiple property tags.

A6 Training and evaluation procedure

A6.1 Chemical language modeling

A6.1.1 QED dataset.

We started training the models with the vanilla PLM objective (4) on the QED dataset until validation perplexity saturated ($\sim$ 4 days, single-GPU). Thereafter, the models were further refined on the same dataset by alternating every 50 steps between objectives (5) and (7). We perform ablation studies on the self-consistency loss, setting α in (7) to 0 and 1 respectively.

A6.1.2 MoleculeNet dataset.

For the MoleculeNet datasets, the models were warm-started using the QED initialization and trained only for 50k steps (batch size 4) with early stopping. Since the QED pretraining utilized numerical values in $[0, 1]$, we normalized the regression values of the MoleculeNet datasets to the same range and rounded them also to three decimal places. For all objectives, unless otherwise constrained, we set the masking hyperparamter $c = 5$ and restrict the span of consecutively masked tokens to a maximum of 5 tokens.

A6.1.3 Property-optimization benchmark

For this task, the models were also warm-started using the QED initialization and trained for 50k steps with early stopping on perplexity. To assemble the candidates for the optimization of one seed molecule, we tried to follow the process of Jin et al. (2018) as closely as possible. Jin et al. (2018) applied 80 gradient steps, then decoded 80 molecules and reported the molecule with the highest pLogP score that satisfies the similarity constraint δ. Instead, we form a pool of molecules by prompting 80 times with the same seed molecule but varying the fraction and the maximum span of masked tokens. From the pool of decodings we report the molecule with the highest pLogP, just like Jin et al. (2018) and You et al. (2018).

A6.2 Protein sequence language modeling

A6.2.1 Boman dataset

To model protein sequences, we started with training on the Boman dataset. We trained three groups of models, one for the vanilla PLM objective (Equation 3) and two for the alternating objectives. We again alternated every 50 steps between optimizing (Equation 5) and (Equation 7) and trained one set of models with and one set without the self-consistency loss, such that $\alpha = 1$ and $\alpha = 0$ respectively in Equation 7. Models were trained until validation perplexity saturated ($\sim$ 4days, single GPU). The numerical values of the Boman index, originally in the range $[-3.1, 6.1]$ were normalized to $[0, 1]$ and rounded to three decimal places.

A6.2.2 TAPE datasets

Following the ablation study on the loss functions (see Table A4) that revealed the best results for the self-consistency objective, we focused the finetuning exclusively on this configuration. For both datasets, three models were warm-started using the Boman initialization and trained until validation performance saturated ($\sim$ 100k steps). The numerical values were again scaled to $[0, 1]$. On the Fluorescence data, a small value of Gaussian noise was added to some training samples due to an interesting failure mode (see A9.2). For the evaluation of the conditional generation task, the models were given more flexibility: 60% of the tokens were masked (i.e., $c = 1.7$ in Equation 3) and the maximum span was 7 AA residues. We did not evaluate the RT on conditional generation for the Fluorescence dataset because of a massive pretraining-finetuning mismatch: While the Boman dataset used for pretraining consisted of 15 to 45 residues (mean/std: 36 ± 7), the fluorescence proteins were significantly larger (246 ± 0.2 residues). Instead, the proteins in the stability dataset were similar in size to the pretraining data (45 ± 3 residues).

A6.3 Hyperparameter & experimental infrastructure.

A6.3.1 Regression Transformer

We used the default XLNet configuration from the Huggingface interface unless otherwise specified. Specifically, we used 32 hidden layers in the Transformer encoder, with a dimensionality of 256 and 1024 in the feed-forward layer and 16 attention heads. Dropout was 0.2 and the SMILES/SELFIES vocabulary had 724 and 509 tokens respectively. During evaluation, greedy decoding was used for property prediction and beam search decoding for molecular generation. During the latter, we gave the model more flexibility by setting $c = 2.5$, s.t., $\sim 40\%$ of the tokens were masked (maximum span: 7 tokens).

We used `PyTorch` 1.3.1 (Paszke et al., 2019) and the XLNet backbone model in `Transformers` 3.1.0 (Wolf et al., 2019). All models were trained on `POWER8` processors and a single `NVIDIA Tesla P100`.

A6.3.2 kNN Baseline.

For chemical language tasks, the distance measure was (inverted) Tanimoto similarity (Tanimoto, 1958) of ECFP4 fingerprints Rogers & Hahn (2010). For the protein language models, the Levenshtein distance between the protein sequences was used (Levenshtein, 1966). For the k-nn baseline models, k was determined based on the best performance on the validation data. This led to $k = 25$ for the drug-likeness/QED task, $k = 21$ for the protein interaction (Boman index) task, $k = 50$ for the fluorescence and $k = 15$ for the stability task.

A7 Ablation studies

A7.1 Ablation study on numerical encoding

A7.1.1 Visualization of float encodings

A7.1.2 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[4]; similar to positional encodings. We therefore propose to directly utilize the definition

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

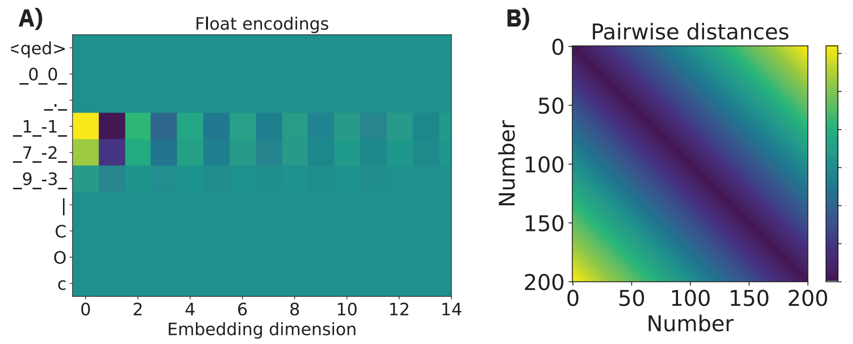


Figure A2: **Float-based numerical encodings.** *A)* Numerical encodings for an molecule with a QED of 0.179. *B)* Pairwise distances of numerical encodings for floats between 0 and 100 (the NEs of all tokens associated to a float are summed up).

from Vaswani et al. (2017) as NEs:

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

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 QED values were casted to the range $[0, 1000]$ and rounded.

A7.1.3 Float vs. Integer-based numerical encodings

Table A1 provides extended results compared to Table 1 in the main paper. It shows the standard deviations

Table A1: **Performance evaluation of PLM training.** FE means our main float-encodings whereas Int refers to the Integer encodings described above.

			Regression task task		*Generation task task*	
Data	NE	Perplexity (↓)	RMSE (↓)	PCC (↑)	0-Var (↓)	SCC (↑)
SMILES	–	**1.55**$_{\pm 0.02}$	0.0549$_{\pm 0.01}$	**0.972**$_{\pm 0.01}$	1.6%$_{\pm 0.2}$	0.096$_{\pm 0.02}$
SELFIES	–	1.61$_{\pm 0.03}$	0.0591$_{\pm 0.00}$	0.968$_{\pm 0.00}$	0.9%$_{\pm 0.2}$	0.427$_{\pm 0.01}$
SELFIES	FE	1.59$_{\pm 0.03}$	**0.0547**$_{\pm 0.01}$	0.971$_{\pm 0.00}$	**0.3%**$_{\pm 0.1}$	**0.467**$_{\pm 0.01}$
SELFIES	Int	1.63$_{\pm 0.02}$	0.0564$_{\pm 0.00}$	0.968$_{\pm 0.00}$	0.8%$_{\pm 0.3}$	0.440$_{\pm 0.01}$

across several runs of the Regression Transformer. In this setting, from the two types of proposed numerical encodings, the float-based encodings yielded slightly superior result to integer-based encodings. Similarly, Table A2 shows extended results of 2 in the main paper, including standard deviations and the ablation study on integer vs. float encodings. Here, integer-encodings (IE) are superior for regression but inferior for conditional

Table A2: **Performance evaluation on alternating objectives.** The decrease in perplexity compared to the vanilla PLM training is expected given the discrepancy between the refined, alternating objective and the PLM objective.

				Regression task task		*Generation task task*	
Data	NE	α	Perplexity	RMSE	Pearson's r	0-Var	Spearman's ρ
SMILES	–	0	2.15$_{\pm 0.1}$	0.0396	0.986	0.8%$_{\pm 0.2}$	0.11$_{\pm 0.02}$
SMILES	–	1	**1.73**$_{\pm 0.1}$	0.0507	0.982	1.1%$_{\pm 0.2}$	0.09$_{\pm 0.02}$
SELFIES	–	0	2.57$_{\pm 0.1}$	0.0341	0.988	0.2%$_{\pm 0.1}$	0.47$_{\pm 0.02}$
SELFIES	–	1	2.41$_{\pm 0.1}$	0.0483	0.978	0.3%$_{\pm 0.1}$	0.49$_{\pm 0.01}$
SELFIES	FE	0	2.10$_{\pm 0.1}$	0.0498	0.982	0.3%$_{\pm 0.1}$	0.468$_{\pm 0.03}$
SELFIES	FE	1	2.67$_{\pm 0.1}$	0.0367	0.987	**0.2%**$_{\pm 0.1}$	**0.52**$_{\pm 0.02}$
SELFIES	Int	0	2.63$_{\pm 0.1}$	**0.0307**	**0.990**	0.7%$_{\pm 0.2}$	0.41$_{\pm 0.01}$
SELFIES	Int	1	2.71$_{\pm 0.1}$	0.0412	0.986	0.8%$_{\pm 0.3}$	0.44$_{\pm 0.01}$

generation. Due to that and the non-applicability of IEs to floating numbers, we decided to not further explore them.

Summation vs. concatenation of numerical encodings. We decided to follow the common approach of *summing* additional encodings with the learned embeddings (Vaswani et al., 2017; Yang et al., 2019) but note that disentangling content and position embeddings can improve language models (He et al., 2021). 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 A3. We propose to use a summation for two reasons: First, it avoids additional hyperparameters and model weights.

Table A3: **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	$0.0564_{\pm 0.00}$	$0.968_{\pm 0.00}$

Secondly, it probably yields approximately orthogonal subspaces of token embedding and numerical encodings (due to the high dimensionality), obviating 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 (Dai et al., 2019; Bai et al., 2021; Wang & Chen, 2020).

A7.2 Conditional generation: External evaluation vs. self-evaluation

Generally, it is intractable to evaluate the performance in most property-driven molecular generation tasks because the property of interest cannot only be measured in the wet lab. In the main paper, we have reported the predicted ESOL, FreeSolv and Lipophilicity values respectively based on the GROVER approach (Rong et al., 2020), a graph Transformer with large-scale self-supervised pretraining. Table A4 shows that a self-evaluation with the Regression Transformer would have led to very similar results in all three conditional generation tasks. This is reassuring because the RT is, at least in the self-consistency setting ($alpha = 1$), a biased estimator since the model is used itself to optimize the conditional generation process. Based on this finding, we refrained from seeking external validation for the conditional protein generation tasks (TAPE datasets).

Table A4: **Conditional generation for MoleculeNet datasets.** Average performances across all splits for training with alternating objectives are given. "ρ with RT" refers to the self-evaluation whereas "ρ with Grover" refers to to predictions obtained with the model from (Rong et al., 2020).

Metric	$\alpha = 0$, no FE	$\alpha = \mathbf{1}$, no FE	$\alpha = \mathbf{0}$, with FE	$\alpha = \mathbf{1}$, with FE
0-Variance (↓)	$\mathbf{4.4}_{\pm 0.8}$	$5.9_{\pm 1.3}$	$6.1_{\pm 3.7}$	$6.1_{\pm 1.5}$
Spearman ρ based on RT predictions	$0.38_{\pm 0.1}$	$0.38_{\pm 0.0}$	$0.41_{\pm 0.1}$	$\mathbf{0.44}_{\pm 0.0}$
Spearman ρ with Grover predictions	$0.44_{\pm 0.0}$	$0.46_{\pm 0.0}$	$0.46_{\pm 0.1}$	$\mathbf{0.47}_{\pm 0.0}$

(a) **ESOL**

Metric	$\alpha = 0$, no FE	$\alpha = \mathbf{1}$, no FE	$\alpha = \mathbf{0}$, with FE	$\alpha = \mathbf{1}$, with FE
0-Variance (↓)	$7.9_{\pm 2.4}$	$7.5_{\pm 3.6}$	$8.9_{\pm 5.2}$	$\mathbf{6.5}_{\pm 2.6}$
Spearman ρ based on RT predictions	$0.51_{\pm 0.0}$	$\mathbf{0.52}_{\pm 0.1}$	$\mathbf{0.52}_{\pm 0.0}$	$0.44_{\pm 0.1}$
Spearman ρ based on Grover predictions	$0.53_{\pm 0.0}$	$0.56_{\pm 0.0}$	$\mathbf{0.57}_{\pm 0.0}$	$\mathbf{0.57}_{\pm 0.0}$

(b) **FreeSolv**

Metric	$\alpha = 0$, no FE	$\alpha = \mathbf{1}$, no FE	$\alpha = \mathbf{0}$, with FE	$\alpha = \mathbf{1}$, with FE
0-Variance (↓)	$3.6_{\pm 1.6}$	$2.7_{\pm 0.9}$	$4.2_{\pm 1.3}$	$\mathbf{2.7}_{\pm 0.7}$
Spearman ρ based on RT predictions	$0.22_{\pm 0.1}$	$\mathbf{0.29}_{\pm 0.0}$	$0.23_{\pm 0.0}$	$0.26_{\pm 0.0}$
Spearman ρ based on Grover predictions	$0.29_{\pm 0.1}$	$\mathbf{0.35}_{\pm 0.0}$	$0.29_{\pm 0.0}$	$0.34_{\pm 0.0}$

(c) **Lipophilicity**

A7.3 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 A3. 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 A3: **Further results on constrained property optimization benchmark.** JT-VAE is from Jin et al. (2018) and GCPN from You et al. (2018). NE refers to the use of Numerical Encodings.

	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	–	–	$4.20_{\pm 1.3}$	$\mathbf{0.32}_{\pm 0.1}$	**100%**	*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*
RT	✓	1	$\mathbf{4.45}_{\pm 1.7}$	$0.35_{\pm 0.1}$	99.6%	0.92
RT	✓	0	$4.12_{\pm 1.7}$	$\mathbf{0.36}_{\pm 0.1}$	99.6%	0.90
RT	✗	1	$4.34_{\pm 1.6}$	$0.35_{\pm 0.1}$	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*
RT	✓	1	$\mathbf{3.16}_{\pm 1.5}$	$0.54_{\pm 0.1}$	97.1%	0.92
RT	✓	0	$2.87_{\pm 1.5}$	$\mathbf{0.55}_{\pm 0.1}$	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	–	–	$0.21_{\pm 0.7}$	$\mathbf{0.69}_{\pm 0.1}$	46.4%	*Unfeasible*
GCPN	–	–	$0.79_{\pm 0.6}$	$0.68_{\pm 0.1}$	**100**%	*Unfeasible*
RT	✓	1	$\mathbf{2.21}_{\pm 1.3}$	$\mathbf{0.69}_{\pm 0.1}$	81.7%	0.92
RT	✓	0	$2.04_{\pm 1.3}$	$\mathbf{0.69}_{\pm 0.1}$	75.0%	0.90
RT	✗	1	$2.10_{\pm 1.3}$	$\mathbf{0.69}_{\pm 0.1}$	81.1%	0.91
RT	✗	0	$2.10_{\pm 1.3}$	$\mathbf{0.69}_{\pm 0.1}$	81.6%	**0.94**

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

A7.4 PROTEIN SEQUENCE LANGUAGE MODELING

A7.4.1 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 ($\mathcal{J}_G$; Equation 6, equivalent to setting $\alpha = 0$ in Equation 7),
3. alternating objectives with the self-consistency term ($\mathcal{J}_{SC}$; Equation 7; $\alpha = 1$).

The results in Table A4 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,

Table A4: **Ablation study on training schemes for Boman dataset.** Legend like Table 1.

		Regression task		*Generation task*	
Model	Loss	RMSE (↓)	Pearson's r (↑)	0-Var (↓)	Spearman ρ (↑)
k-NN	–	0.53	0.932	*Task unfeasible*	
RT	PLM	$0.69_{\pm 0.03}$	$0.944_{\pm 0.0}$	$0.3_{\pm 0.4}$	$0.76_{\pm 0.03}$
RT	$\mathcal{J}_G$	$\mathbf{0.17}_{\pm 0.04}$	$\mathbf{0.994}_{\pm 0.0}$	$\mathbf{0.2}_{\pm 0.1}$	$0.82_{\pm 0.01}$
RT	$\mathcal{J}_{SC}$	$0.20_{\pm 0.04}$	$0.991_{\pm 0.0}$	$\mathbf{0.2}_{\pm 0.1}$	$\mathbf{0.84}_{\pm 0.00}$

like on the QED dataset, the self-consistency 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 A3 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 A3). This comes, however, at the cost of reduced similarity to the seed sequence.

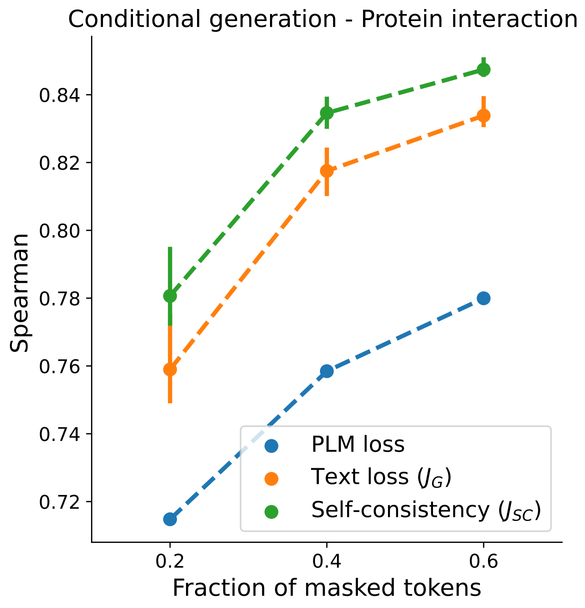


Figure A3: **Correlation between property primer and property of generated protein sequences** The model's ability to generate protein sequences with a desired protein interaction index. The self-consistency loss yielded the best results and, generally, a higher fraction of masked tokens led to generated 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.

A8 Case studies

A8.1 Case study on scaffold hopping

Scaffold hopping is a technique in medicinal chemistry with the goal to discover novel compounds by modifying the central core structure (i.e., removing substituents while retaining rings and their linker fragments) of known compounds (Bajorath, 2017). We simulated this task on the QED dataset by determining the scaffold 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 A5).

Table A5: **Scaffold hopping 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 (↑)
$\mathcal{J}_G$	Masking non-scaffold	8.55%	0.136
$\mathcal{J}_{SC}$	Masking non-scaffold	9.76%	0.105
$\mathcal{J}_G$	Masking randomly	$0.80\%_{\pm 0.19}$	$0.108_{\pm 0.01}$
$\mathcal{J}_{SC}$	Masking randomly	$1.14\%_{\pm 0.19}$	$0.085_{\pm 0.02}$

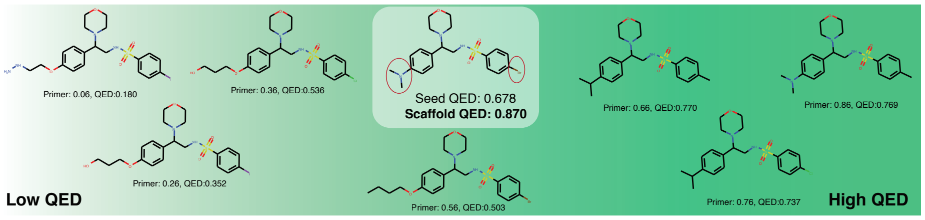


Figure A4: **Molecules sampled in a scaffold hopping task.** Only non-scaffold tokens (encircled in red) were masked.

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 A4. 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 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.

A8.2 Case study on attention visualization: Interpreting attention heads

We visualized the attention scores using BertViz (Vig, 2019). 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 A5. 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 A5, 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 A5, 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.

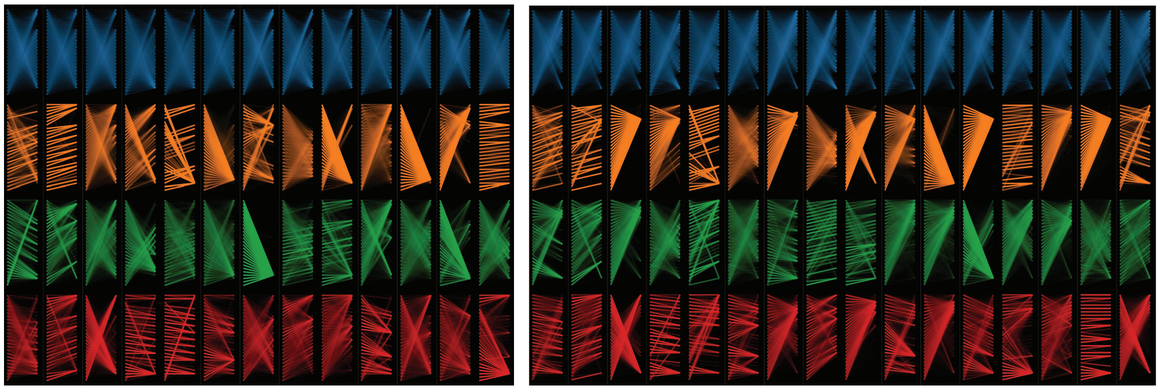


Figure A5: **Comparing attention scores across both tasks with BertViz (Vig, 2019).** Attention scores for all heads of the first four layers. Rows depict layers, column depict attention heads. Within each cell, the tokens are ordered from top to bottom. *Left:* Property prediction task. *Right:* Conditional generation task. Plot performed with SELFIES model with float encodings, trained on the self-consistency loss.

A9 Additional results

A9.1 Learning embeddings of numbers.

We sought to understand why the ablation studies on the numerical encodings (NE) on the QED dataset (Table 1 and 2) reveal only mild but not enormous superiority of models with NEs. Interestingly, in the absence of static NEs, the model learns the natural ordering of digits from the data (cf. Figure A6). A large number of embedding dimensions (47% and 36% for the decimal places -1 and -2 respectively) directly and significantly encoded the ordering of digits (i.e., $p < 0.05$ and $|PCC| > 0.62$ between the 10 embedding values and a strictly monotonic vector). For example, in Figure A6*A* the digit value is monotonically related to its embedding value. Notably, this ordering trend was much less present in the models using NEs ($\sim 16\%$). For reference, with random weights, 5% would be expected).

A9.2 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 A7). An interesting failure mode was observed when initially training on the Fluorescence dataset. Figure A7 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

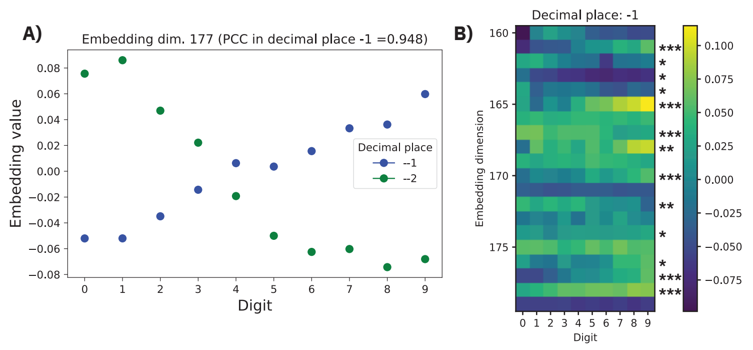


Figure A6: **Learned embeddings of numerical tokens.** *A)* For an exemplary dimension, embeddings for 20 tokens, corresponding to 10 digits and 2 decimal places are shown. *B)* Embeddings for 20 exemplary dimensions across 10. The stars indicate the significance level of the Pearson correlation. The analysis is based on the SELFIES model without static NEs (PLM objective).

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.

A9.3 Detailed performance - Classification vs. Regression

Figure A7 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,

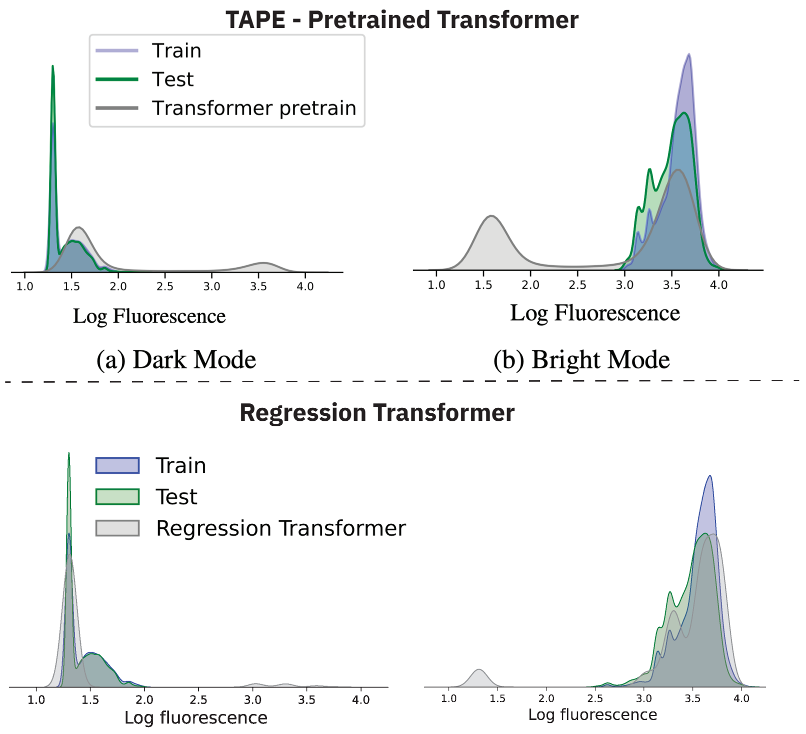


Figure A7: **Bimodal mode of fluorescence data.** The upper part of the plot has been copied from Rao et al. (2019) (Figure 3). It shows the bimodal mode of the training data and the test predictions from the TAPE Transformer. At the bottom, we show our remake of the above plot by replacing the predictions from the pretrained TAPE Transformer with the predictions from the Regression Transformer.

grouped by bright and dark test proteins respectively (cf. Table A6). While the RT achieved the best results

Table A6: **Detailed fluorescence prediction results.** MSE abbreviates mean squared error. TAPE and UniRep performances taken from Rao et al. (2019). 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

in the overall Spearman ρ, the recommended metric by Rao et al. (2019), it does not 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.

A9.4 Negative learning.

To actively facilitate the versatility of the generated molecules, we experimented with adaptations of Equation (6). In particular, instead of providing the true property $\mathbf{x}^p$, we sampled in 50% of the cases a property primer $\hat{\mathbf{x}}^p$ that was distant from the original property. For these samples the objective (6) was inverted, s.t., the goal was not to reconstruct the original molecule but to provide a maximally different molecule. Due to mixed results and the high novelty scores for all models, we did not include the experiments in the final version.