AMLDEPFL

AMLDEPFL

Regression Transformer

-

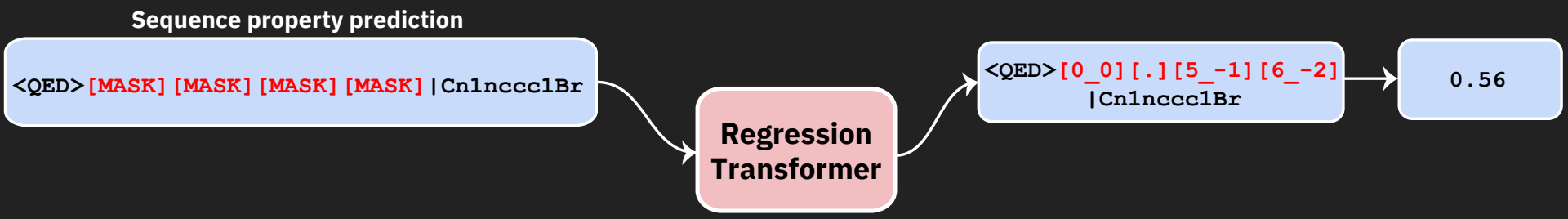
Concurrent Property Prediction and Conditional Molecular Generation by Blending Numerical and Textual Tokens

Applied Machine Learning Days (AMLD)
Track: *AI & the molecular world*

Jannis Born
30.03.2022

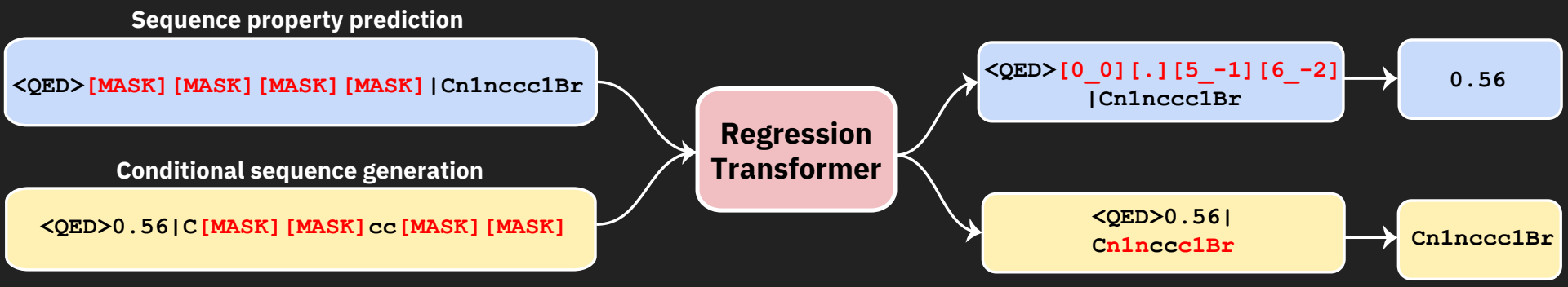
Regression Transformer (RT)

- Formulate regression as a conditional sequence modelling problem



Regression Transformer (RT)

- Formulate regression as a conditional sequence modelling problem



- This yields a dichotomous model that can seamlessly transition between property prediction and property-driven conditional text generation

Motivation I: Decline of inductive biases

- 2012: AlexNet – CNNs for object recognition (Krizhevsky et al., NeurIPS)
- 2015: Self-attention generalizes fully-connected layers (Luong et al., EMNLP)
- 2017: Transformers supersede RNNs in NLP (Vaswani et al, NeurIPS)
- 2019: Vision Transformers can match CNNs (Ramachandran et al, NeurIPS)

- 2021: Transformers are universal computation engines (Lu et al, AAAI)
- 2021: Abstract offline RL to sequence modelling (Chen et al., NeurIPS)

Motivation II: Generative Chemistry

Canonical approach

Labeled Data

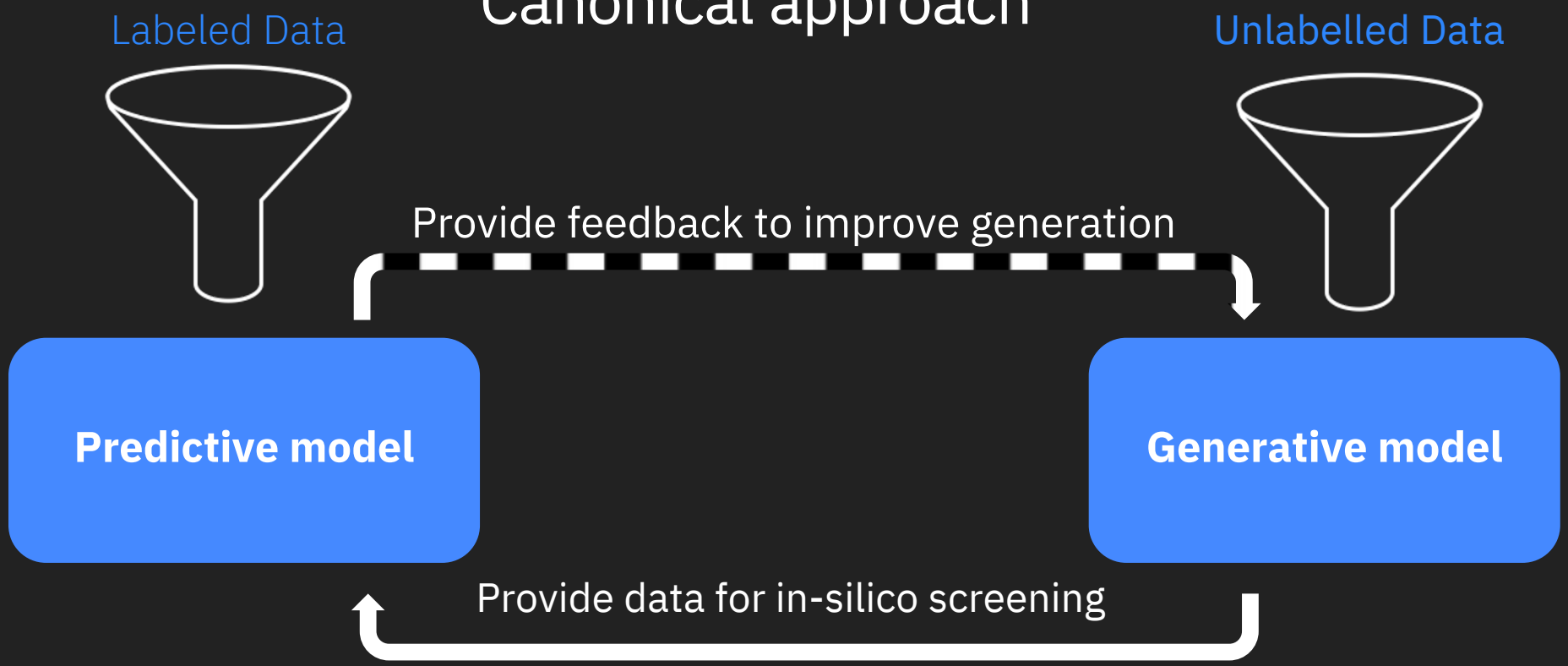
Predictive model

Unlabelled Data

Generative model

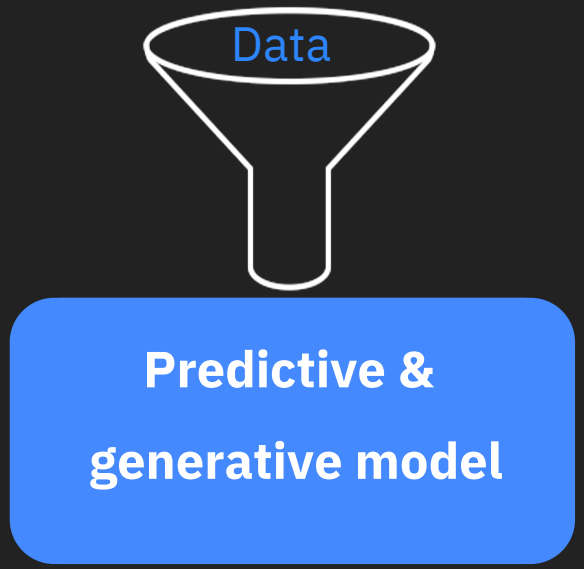
Motivation II: Generative Chemistry

Canonical approach



Motivation II: Entangle prediction & generation

Regression Transformer approach



Tokenization

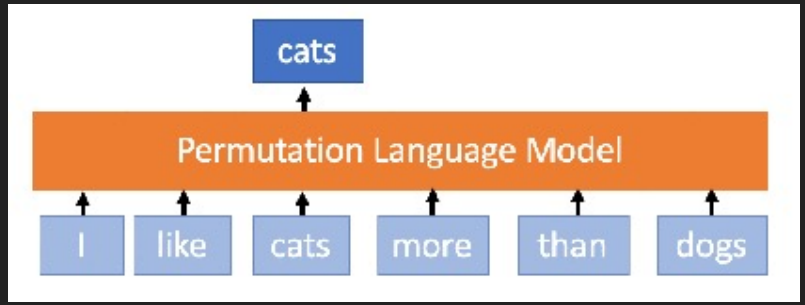
Numbers are tokenized into sequences of “numerical tokens”

<QED>0.51 | <Tox>1.4 | N#[N+][N-]c1ccc(C)cc1

<QED> 0_0 _._ _5_-1 _1_-2 | <Tox> 1_0 _._ 4_-1 | N …

RT backbone: XLNet model

- XLNet –A bidirectional, autoregressive Transformer (Yang et al, 2019, NeurIPS)
- Train with Permutation Language Modeling (PLM)
- PLM: Sample factorization order at runtime



1. Overcomes BERT's independence assumption in multiple token generation

   Example: *The largest city in the US is [MASK] [MASK]*

2. Unlike GPT-2, XLNet fully attends contextual information from both sides

   Example: *The city [MASK] [MASK] has the largest population in US*

Training objectives

- Vanilla PLM objective

$$\max_{\theta} \quad \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_T}\left[\sum_{t=1}^{T}\log p_\theta(x_{z_t} \mid \mathbf{x}_{\mathbf{z}_{<t}})\right].$$

<QED>0.51|N#[N+][N-]c1ccc(C)cc1

Randomly mask 20% of tokens

Molecular property prediction

- Dataset: Synthetic data of QED scores of molecules
- Evaluation queries: <QED><MASK><MASK><MASK><MASK>|N#[N+][N-]c1ccc(C)cc1

Configuration			*Regression task*	
Data	NE	Perpl.	RMSE	PCC (↑)
SMILES	–	**1.55**	0.0549	**0.972**
SELFIES	–	1.61	0.0591	0.968
SELFIES	✓	1.59	**0.0547**	0.971

- Comparison to conventional regression models

Model	MAE (↓)
k-NN (baseline)	0.054
SMILES-BERT (Kim et al., 2021)	0.020
RT - PLM objective	0.035

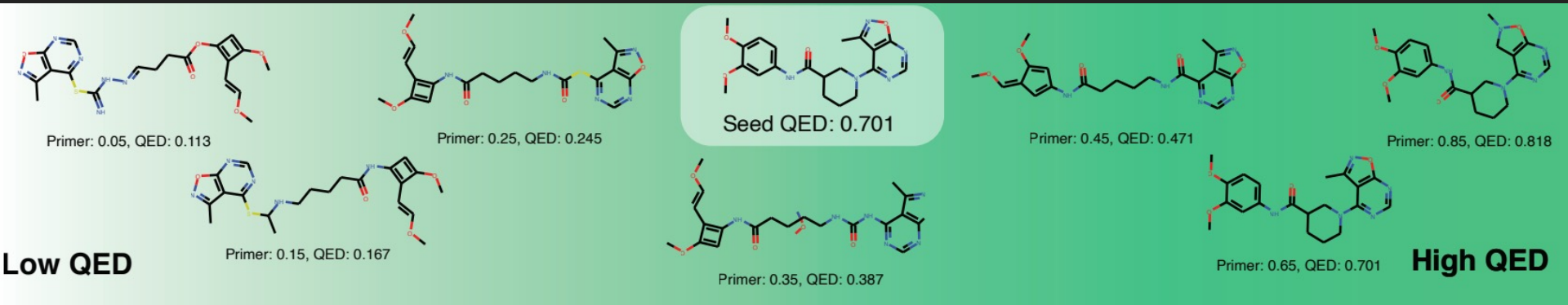
- No regression loss! This is achieved ***despite*** casting regression as a conditional sequence modelling problem & training with cross entropy loss.

Conditional molecular generation

- Task: Substructure-constrained, property-driven molecular generation
- Evaluation queries: `<QED>0.26|C1<MASK><MASK><MASK>CC1OCC(=O)NC<MASK><MASK>=C(NC..`

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**

- ρ is the Spearman correlation between the QED score of 10 property primers/prompts (0.1 – 0.9) and the QED of the obtained molecules



Refined, alternating training objectives

- Vanilla PLM objective (Yang et al, NeurIPS)

$$\mathcal{J}_{PLM} = \max_{\theta} \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_T}\left[\log p_\theta(\mathbf{x}_{\mathbf{z}>c}|\mathbf{x}_{\mathbf{z}_{\leq c}})\right]$$

`<QED>0.51|N#[N+][N-]c1ccc(C)cc1`

Randomly mask 20% of tokens

- Property prediction objective

$$\mathcal{J}_P = \max_{\theta} \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_T^p}\left[\log p_\theta(\mathbf{x}^p|\mathbf{x}^t)\right]$$

`<QED> 0.51 |N#[N+][N-]c1ccc(C)cc1`

N.M. | Mask | No mask

- Self-consistency objective for conditional generation

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

with

$$\mathcal{J}_G = \max_{\theta} \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_T^t}\left[\log p_\theta(\mathbf{x}^t_{\mathbf{z}_{>c}}|\mathbf{x}^p_{\mathbf{z}_{\leq k}}, \mathbf{x}^t_{\mathbf{z}_{>k<c}})\right]$$

`<QED>0.51|` `N#[N+][N-]c1ccc(C)cc1`

Not masked | Randomly masked

+

`<QED> 0.51 |N#[N+][N-]c1ccc(C)cc1`

NM | Mask | Generated before

Property prediction w/ alternating objectives

- Same dataset: Synthetic data of QED scores of molecules

Regression

Model	RMSE	PCC
RT – PLM objective	0.0547	0.971
RT – Refined objective	**0.0367**	**0.987**

- Comparison to conventional regression models

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

Conditional generation w/ alternating objectives

- Same dataset: Synthetic data of QED scores of molecules

	Regression		*Generation*	
Model	**RMSE**	**PCC**	**0-Var**	**Spearman**
RT – PLM objective	0.0547	0.971	0.3%	0.47
RT – Refined objective	**0.0367**	**0.987**	**0.2%**	**0.52**

- Comparison to conventional regression models

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

Molecular property prediction

- Real datasets: Solubility & lipophilicity (MoleculeNet benchmark)

Metric: RMSE (↓)

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

- → RT outperforms baseline methods in molecular property prediction
- → RT cannot beat Transformers with finetuned regression heads

Conditional molecular design

- But: The RT can *concurrently* generate molecules with desired property

Model	NE	α	ESOL 0-Var	ESOL ρ	FreeSolv 0-Var	FreeSolv ρ	Lipophilicity 0-Var	Lipophilicity 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*					

- E.g., solubility: Rank correlation between the 10 property primers and the (predicted) solubility of generated molecules is ~0.45

Conditional generation benchmark

- Task: Given a seed molecule, generate molecules with a higher logP score, while adhering to a similarity constraint (δ)

<u>Result:</u>
The RT outperforms competitive approaches in conditional molecular design

Table 6. **Constrained property optimization benchmark.** JT-VAE is from Jin et al. (2018) and GCPN from You et al. (2018).

	Generation task			*Regression*
Model	Improvem.	Similarity δ	Success	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 (Ours)	$\mathbf{3.16}_{\pm 1.5}$	$\mathbf{0.54}_{\pm 0.1}$	97.1%	$\mathbf{0.92}_{\pm 0.0}$

(a) Similarity threshold $\delta = 0.4$

	Generation task			*Regression*
Model	Improvem.	Similarity δ	Success	PCC
JT-VAE	$0.21_{\pm 0.7}$	$\mathbf{0.69}_{\pm 0.0}$	46.4%	*Unfeasible*
GCPN	$0.79_{\pm 0.6}$	$0.68_{\pm 0.1}$	**100%**	*Unfeasible*
RT (Ours)	$\mathbf{2.21}_{\pm 1.3}$	$\mathbf{0.69}_{\pm 0.1}$	81.8%	$\mathbf{0.92}_{\pm 0.0}$

(b) Similarity threshold $\delta = 0.6$

JT-VAE: Jin et al., ICLR (2018);
GCPN: You et al., NeurIPS (2018)

Protein language modeling

Datasets: TAPE benchmark

- RT can match state-of-the art protein language models in protein property prediction (TAPE: Rao et al., NeurIPS 2019; UniRep : Alley et al., Nature Methods 2019)

Table 7. **Protein regression tasks.** All values in Spearman's ρ (↑). TAPE datasets/performances taken from Rao et al. (2019).

Model	Source	Boman	Fluorescence	Stability
k-NN	Baseline	0.93	0.59	0.21
One-Hot	TAPE	–	0.14	0.19
Pretr. LSTM	TAPE	–	0.67	0.69
Pretr. Transformer	TAPE	–	0.68	**0.73**
Alley et al. (2019)	UniRep	–	0.67	**0.73**
RT	**Ours**	**0.99**	$\mathbf{0.72}_{\pm 0.04}$	$0.71_{\pm 0.02}$

- Same model can, to some extent, adapt existing proteins to fulfil a property of interest

Model	Boman dataset		Stability dataset	
	0-Var (↓)	Spearm. ρ	0-Var (↓)	Spearm. ρ
All TAPE	*Task unfeasible*		*Task unfeasible*	
UniRep				
RT	$0.2\%_{\pm 0.0}$	$0.84_{\pm 0.00}$	$19\%_{\pm 4.5}$	$0.44_{\pm 0.01}$

Demo: Generative Toolkit 4 Scientific Discovery

1. Predict solubility of a common herbicide
2. Generate similar molecules with improved solubility

jupyter regression-transformer-demo Last Checkpoint: 28 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

Demo: Regression Transformer in the Generative Toolkit for Scientific Discovery

In [*]:
```
!pip install gt4sd
```

In [1]:
```
import logging, sys
logging.disable(sys.maxsize)


from gt4sd.algorithms.conditional_generation.regression_transformer import (
    RegressionTransformer, RegressionTransformerMolecules
)
from rdkit import Chem
from selfies import encoder
```

Let us have a look at Buturon, a common herbicide

In [ ]:
```
smi = 'CC(C#C)N(C)C(=O)NCl=CC=C(Cl)C=Cl'
Chem.MolFromSmiles(smi)
```

Buturon has a water solubility score of -3.90

We can predict its ESOL (estimated solubility) value with the RegressionTransformer

In [ ]:
```
config = RegressionTransformerMolecules(search='greedy')
target = f"<esol>[MASK][MASK][MASK][MASK][MASK]|{encoder(smi)}"
esol_predictor = RegressionTransformer(configuration=config, target=target)
score = list(esol_predictor.sample(1))[0]
print(f'\nFor Buturuon, the predicted ESOL is {score}')
```

Ok, we can see that the prediction was decently close but not perfect

Now let us try to improve Buturuon to a molecule with higher solubility

Note, that we will use the same model to do so!

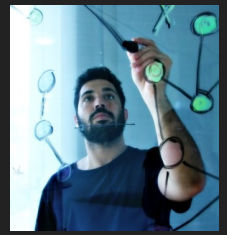
We simply set the hyperparameters for the search and mask undesired parts of the molecule

In [ ]:
```
config = RegressionTransformerMolecules(search='sample', temperature=2, tolerance=5)
target = "<esol>-3.534|[C][C][Branch1_3][Ring1][C][#C][N][Branch1_3][epsilon][C][C][Branch1_3][epsilon][MASK][MASK][MAS
```

Regression Transformer - Conclusion

1. The RT casts regression as conditional sequence modelling problem
2. In some cases, this can match SOTA performance in property prediction tasks despite using a cross-entropy loss
3. The same model can outperform specialized generative models in conditional molecular design benchmarks
4. This opens the door toward extending self-supervised pretraining to labelled datasets

Thanks for your attention



Joint work w/ Matteo Manica

- Read the full paper on arXiv:

Born, J., & Manica, M. (2022). Regression Transformer: Concurrent Conditional Generation and Regression by Blending Numerical and Textual Tokens. *arXiv preprint arXiv:2202.01338*.

- Further experiments on protein language modelling
- Ablation studies on numerical encodings & more

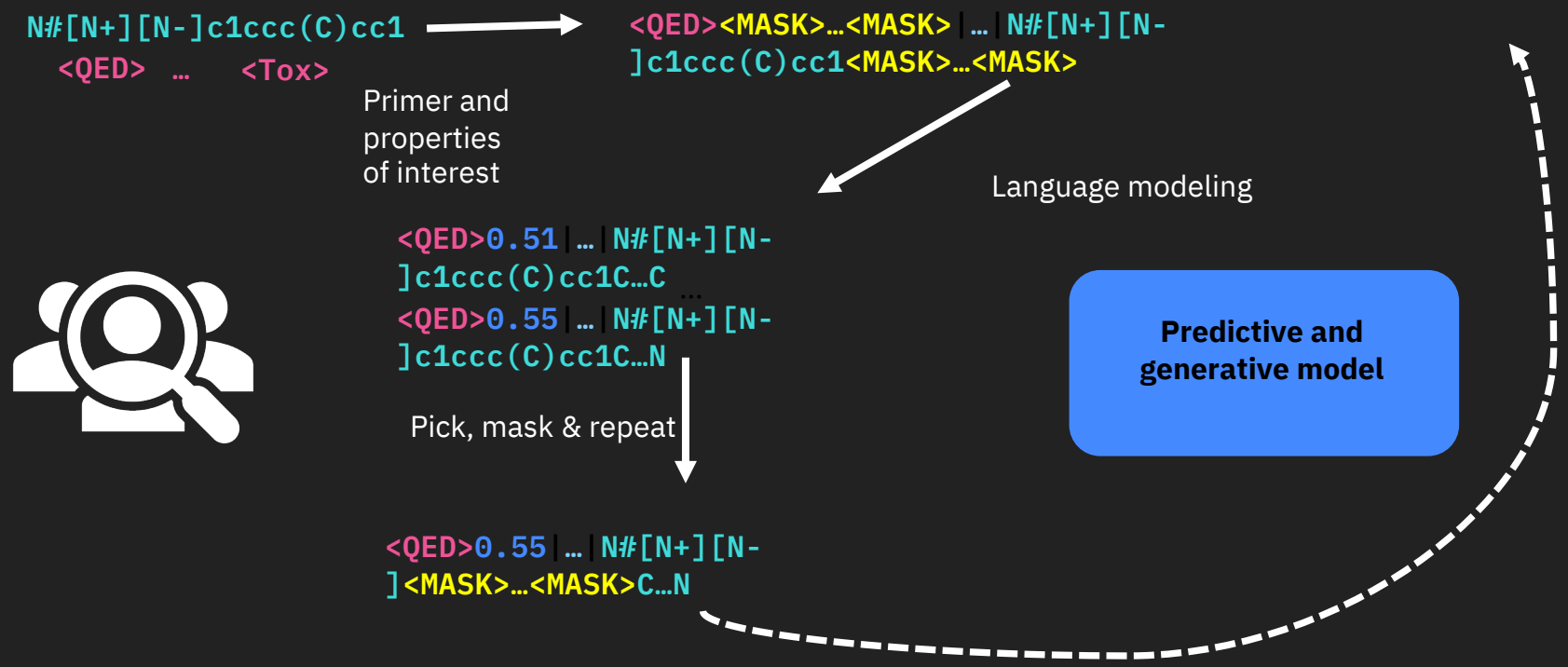
- Code public: https://github.com/IBM/regression-transformer

- Integrated into GT4SD: Generative Toolkit for Scientific Discovery: https://github.com/gt4sd/gt4sd-core

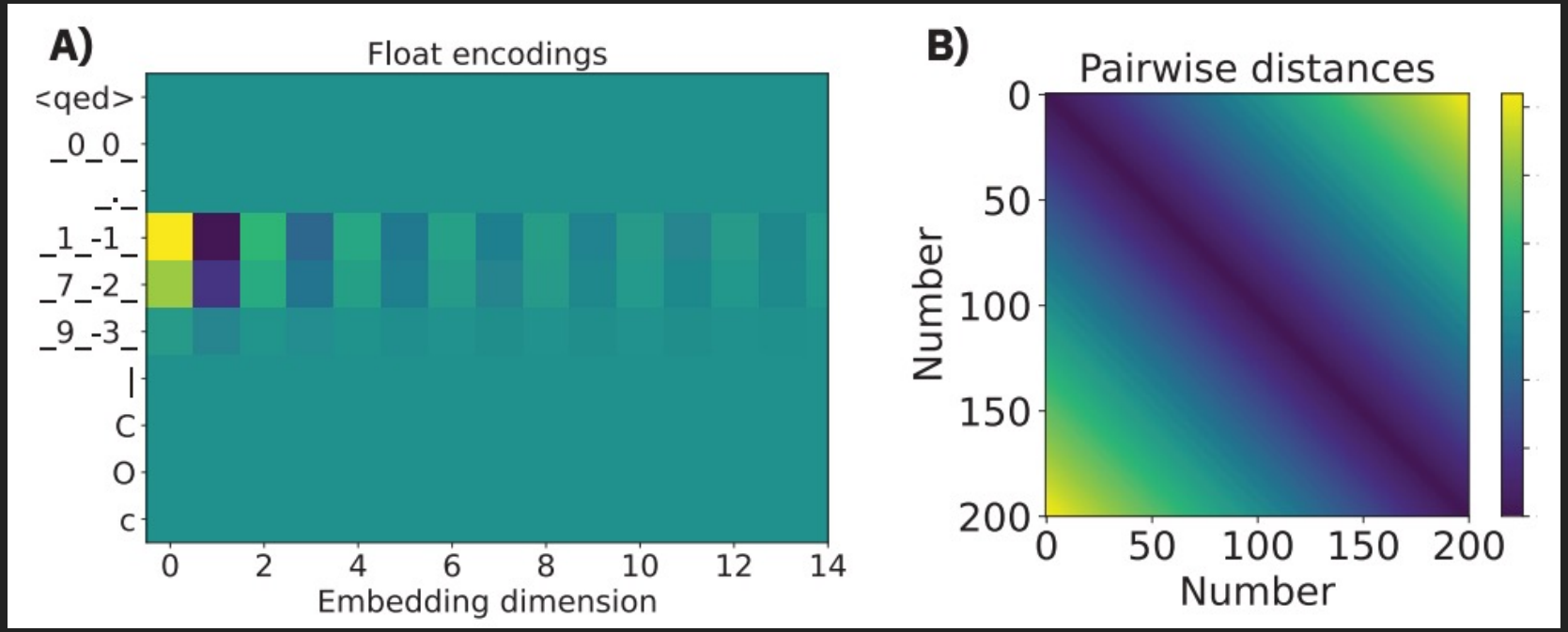
Code

Paper

User-model interaction



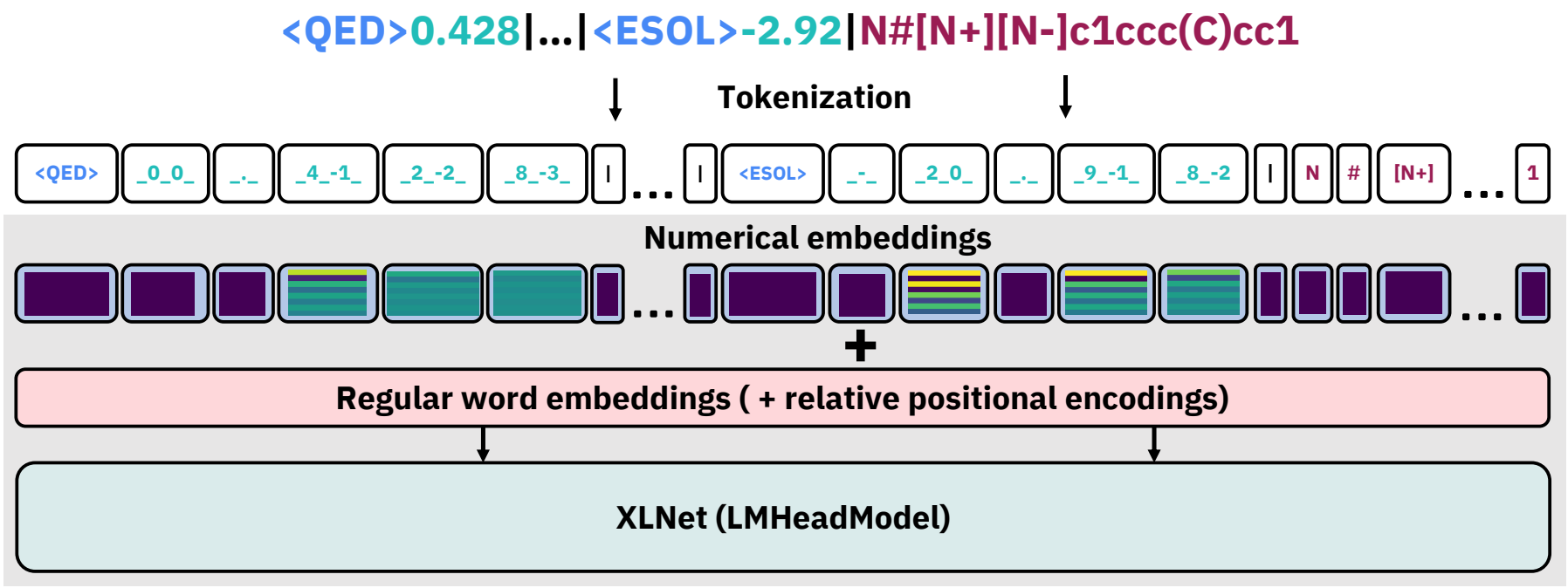
Inductive bias for numerical tokens



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

→ Summing with learned embeddings

Regression Transformer architecture



Trained with PLM objective or with combined property prediction and self-consistency objective

Sequence decoding in Transformers

Example 1: *The largest city in the US is [MASK] [MASK]*

Autoregressive model (e.g., GPT-2): $P(y_0|x_0 \dots x_6) \cdot P(y_1|\, x_0 \dots x_6, y_1)$
BERT: $P(y_0|x_0 \dots x_6) \cdot P(y_1|\, x_0 \dots x_6)$

→ BERT: Independence assumption is prohibitive

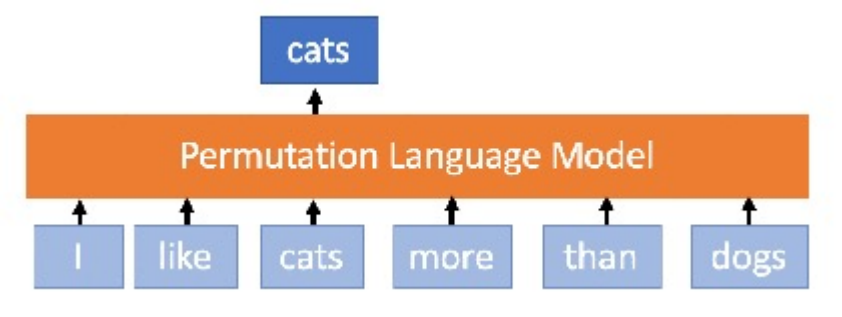
Example 2: *The city [MASK] [MASK] has the largest population in US*

Autoregressive model (e.g., GPT-2): $P(y_0|x_0, x_1) \cdot P(y_1|x_0, x_1, y_0)$
BERT: $P(y_0|x_0, \dots x_7) \cdot P(y_1|\, x_0, \dots x_7)$

→ Autoregressive model is blind to the future

RT backbone: XLNet model

- Solution: XLNet –A bidirectional, autoregressive Transformer (Yang et al, 2019, NeurIPS)
- Train with Permutation Language Modeling (PLM)
- PLM: Sample factorization order at runtime



1. Overcomes BERT's independence assumption in multiple token generation

2. Unlike GPT-2, XLNet fully attends contextual information from both sides

Protein design

- Dataset: Fluorescence & stability dataset are from TAPE benchmark

- Same model can, to a decent extent, adapt existing proteins to fulfil a property of interest

Protein property prediction

- Dataset: Fluorescence & stability dataset are from TAPE benchmark

Table 7. **Protein regression tasks.** All values in Spearman's ρ (↑). TAPE datasets/performances taken from Rao et al. (2019).

Model	Source	Boman	Fluorescence	Stability
k-NN	Baseline	0.93	0.59	0.21
One-Hot	TAPE	–	0.14	0.19
Pretr. LSTM	TAPE	–	0.67	0.69
Pretr. Transformer	TAPE	–	0.68	**0.73**
Alley et al. (2019)	UniRep	–	0.67	**0.73**
RT	**Ours**	**0.99**	$\mathbf{0.72}_{\pm 0.04}$	$0.71_{\pm 0.02}$

- RT can match state-of-the art protein language models in protein property prediction (TAPE: Rao et al., NeurIPS 2019; UniRep : Alley et al., Nature Methods 2019)

Protein design

- Dataset: Fluorescence & stability dataset are from TAPE benchmark

Model	Boman dataset		Stability dataset	
	0-Var (↓)	Spearm. ρ	0-Var (↓)	Spearm. ρ
All TAPE	*Task unfeasible*		*Task unfeasible*	
UniRep				
RT	$0.2\%_{\pm 0.0}$	$0.84_{\pm 0.00}$	$19\%_{\pm 4.5}$	$0.44_{\pm 0.01}$

- Same model can, to a decent extent, adapt existing proteins to fulfil a property of interest

Motivation III: Self-supervised pretraining

Transformers

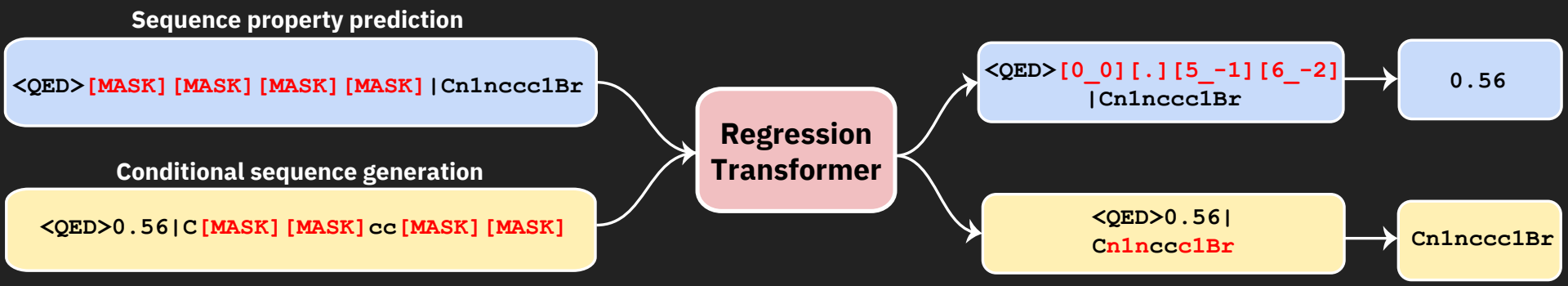
Processing text (word2vec) ✓

Self-supervised pretraining ✓

Encode continuous properties

How to extend self-supervised pretraining (BERT-style) to numerically labelled data?

Regression Transformer (RT)

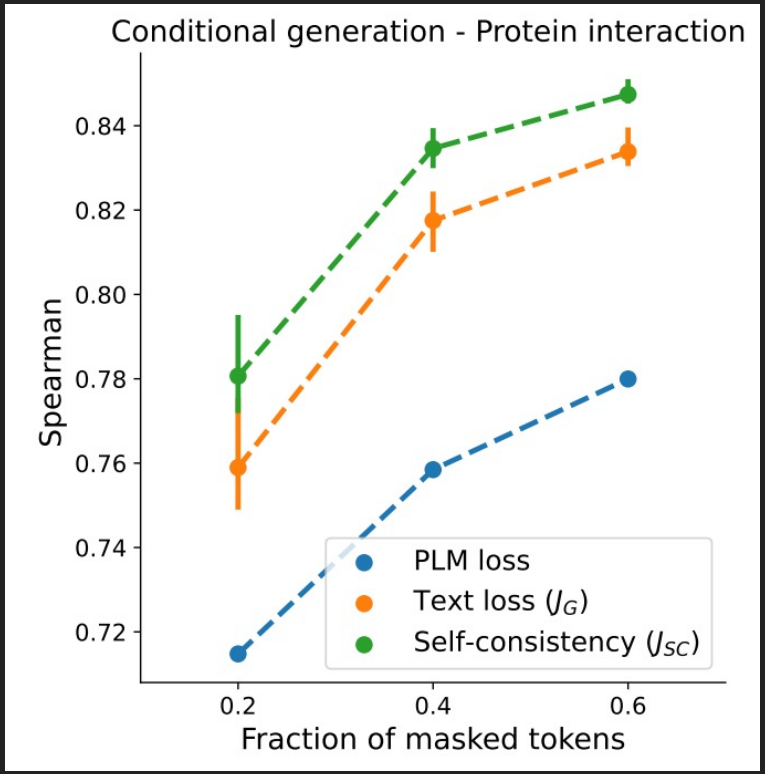


- Idea: Relax the inductive bias of discriminative modelling
- Let's learn joint distributions over input and target variables
- → Blur lines between predictive and conditional generative modeling

Protein design II

- More freedom (i.e., masked tokens) in the protein design task leads to better results

- But this comes at the cost of lower diversity



Property prediction results

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

Metric: RMSE (↓)

- The RT cannot match Transformers finetuned with a regression head, but....

Outline

1. Motivations for the Regression Transformer (RT)
2. How does the RT work?
3. Experiments on chemical languages
4. Experiments on protein languages

Motivation for Regression Transformer

1. Entangle molecular design & property prediction in generative chemistry
2. Decline of inductive biases in ML
3. Extend self-supervised pretraining to continuous properties

Color palette

Black R0 G0 B0 #000000	**Gray 100** R22 G22 B22 #161616	**Gray 90** R38 G38 B38 #262626	**Gray 80** R57 G57 B57 #393939	**Gray 70** R82 G82 B82 #525252	**Gray 60** R111 G111 G111 #6f6f6f	**Gray 50** R141 G141 B141 #8d8d8d	**Gray 40** R168 G168 B168 #a8a8a8	**Gray 30** R198 G198 B198 #c6c6c6	**Gray 20** R224 G224 B224 #e0e0e0	**Gray 10** R244 G244 B244 #f4f4f4
	Blue 100 R0 G17 B65 #001141	**Blue 90** R0 G29 B108 #001d6c	**Blue 80** R0 G45 B156 #002d9c	**Blue 70** R0 G67 B206 #0043ce	**Blue 60** R15 G98 B254 #0f62fe	**Cyan 50** R17 G146 B232 #1192e8	**Cyan 40** R51 G177 B255 #33b1ff	**Cyan 30** R130 G207 B255 #82cfff	**Cyan 20** R186 G230 B255 #bae6ff	**Cyan 10** R229 G246 B255 #e5f6ff
	Red 50 R250 G77 B86 #fa4d56	**Red 40** R255 G131 B137 #ff8389	**Red 30** R255 G179 B184 #ffb3b8	**Red 20** R255 G215 B217 #ffd7d9	**Red 10** R255 G241 B241 #fff1f1	**Purple 50** R165 G110 B255 #a56eff	**Purple 40** R190 G149 B255 #be95ff	**Purple 30** R212 G187 B255 #d4bbff	**Purple 20** R232 G218 B255 #e8daff	**Purple 10** R246 G242 B255 #f6f2ff
	Green 30 R111 G220 B140 #6fdc8c	**Green 20** R167 G240 B186 #a7f0ba	**Green 10** R222 G251 B230 #defbe6	**Yellow 20** R253 G220 B105 #fddc69	**Yellow 10** R252 G244 B214 #fcf4d6	**Teal 50** R0 G157 B154 #009d9a	**Teal 40** R8 G189 B186 #08bdba	**Teal 30** R61 G219 B217 #3ddbd9	**Teal 20** R158 G240 B240 #9ef0f0	**Teal 10** R217 G251 B251 #d9fbfb