Supplimentary Material: Controlled Molecule Generation via Self-Attention based Translation

Bonggun Shin
Deargen Inc.
Seoul, South Korea
bonggun.shin@deargen.me

Sungsoo Park
Deargen Inc.
Seoul, South Korea
sspark@deargen.me

JinYeong Bak
SungKyunKwan University
Suwon, South Korea
jy.bak@skku.edu

Joyce C. Ho
Emory University
Atlanta, GA
joyce.c.ho@emory.edu

1 Background

To efficiently present the idea of the proposed model, we briefly overview Transformer [4], the basic building block of the proposed model.

Input Embedding: For a given input sequence, $X = \{x_1, x_2, \cdots, x_i, \cdots, x_L\}, x_i \in \mathbb{R}^V$, where L is the length of the sequence and V is the number of vocabulary, we transform each token into a continuous vector, which is the sum of a token embedding vector and the positional embedding vector. These token embeddings are similar to word embeddings [2] except they are randomly initialized, therefore, each token, x_i is transformed into $v_i \in \mathbb{R}^d$, where d is the token embedding size. The token embeddings themselves are not sufficient to represent a sequence with a self-attention network, because a self-attention doesn't consider the sequence order when calculating the attention, unlike other attention mechanisms. Therefore, we add a fixed positional embedding, $p_i \in \mathbb{R}^d$, to v_i that makes the final input representation, $e_i = v_i + p_i, e_i \in \mathbb{R}^d$.

Self-Attention Layer: These transformed vectors, e_i, are the inputs of the encoder, consisting of multiple stacks of a self-attention layer and a feed-forward network. Each self-attention layer possesses three dense networks; a query network ($f_{\theta_Q}, \theta_Q \in \mathbb{R}^{d \times h}$), key network ($f_{\theta_K}, \theta_K \in \mathbb{R}^{d \times h}$), and value network ($f_{\theta_V}, \theta_V \in \mathbb{R}^{d \times h}$), where h is the hidden dimension. With these three networks, each input vector, e_i is projected into three utility vectors, a query vector (q_i), key vector (k_i), and value vector (v_i). Now, the output of a self-attention layer is computed as:

$$\begin{aligned} S &= \text{Attention}(Q, K, V) \\ &= \text{softmax}(\frac{QK^T}{\sqrt{h}})V \in \mathbb{R}^{L \times h} \end{aligned} \tag{1}$$

This self-attention computation (Equation 1) can be repeated H number of times with the same input, forming the multi-head attention.

Feed-Forward Layer: The outputs of this multi-head attention are concatenated and projected using another dense network, called an intermediate dense layer, which parameter is represented as $\theta_O \in \mathbb{R}^{H \cdot h \times d}$. Then, it forms the final output of one encoder block, $o_i \in \mathbb{R}^d$.

Encoder: The Transformer encoder is multiple stacks of the two layers; the self-attention layer and the feed-forward layer explained above. Note that the sequence length is preserved because the self-attention is applied to its own sequence, which preserves the input-output length. Therefore, the

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

final output of the Transformer encoder is $z_i \in \mathbb{R}^d$, which is compatible to be an input to another encoder.

Decoder: The Transformer decoder includes not only the same sub-layers as the Transformer encoder but one additional layer, the cross attention layer. It is similar to the self-attention layer in that it's controlled by three vectors, (q_i, k_i, v_i). The difference is that its attention weights are calculated between the last unit's output of the Transformer encoder and each unit of the decoder, while the self-attention layer calculates its weights between the same layers.

Loss Function: The output of the last unit of the Transformer decoder is passed through the two dense layers, one for producing logits for all tokens and another for producing vocabulary probabilities for all tokens, $\hat{Y} = \{\hat{y_1}, \hat{y_2}, \cdots, \hat{y_j}, \cdots, \hat{y_M}\}$, where M is the number of the output tokens. Given an output sequence, $Y = \{y_1, y_2, \cdots, y_j, \cdots, y_N\}$, and the predictions, $\hat{Y}$, the loss function is a cross entropy that can be formally defined as:

$$\begin{aligned} &\mathcal{L}_T(\boldsymbol{\theta}_T; X, p_X, p_Y) \\ &\quad = -\frac{1}{N}\frac{1}{M}\sum_{n\in N}\sum_{j\in M}\sum_{v\in V} y_{v,j,n} \cdot \log(\hat{y}_{v,j,n}) \end{aligned} \tag{2}$$

$\boldsymbol{\theta}_T$ denotes all parameters of the Transformer and N represents the number of training samples.

2 Implementation Details

For the reproducibility, we provide the implementation details including model configuration and hyper-parameters that we used.

2.1 Pre-Training of Constraint Networks

In this section, we describe the network configuration of the two constraint networks (PropNet and SimNet), and details of the pretraining process of them.

PropNet: We use 64-dimensional hidden vectors in the biLSTM layer, and 100 dimensions in the second last dense layer. We use Adam optimizer [1] with default parameters set by Tensorflow Keras[1]. The batch size is 4,096 and the number of epochs is 1,000. The best model was selected by evaluating 20% of the data, the validation set of PropNet described above. As a result, we selected the model at the 715th epoch, the mean square error (MSE) of the test set of which is 0.08554. Considering the values of QED and DRD2 range from 0 to 1, and the values of penalized logp typically range from -10 to 10, this MSE is small enough to be used as a property estimation.

SimNet: We use 64-dimensional hidden vectors in the biLSTM layer, and 100 dimensions in the second last dense layer. We use the same optimizer as PropNet. The batch size is 4,096 and the number of epochs is 1,000. The best model was selected by evaluating 20% of the data, test set set of SimNet described above. As a result, we selected the model at the 755th epoch, which recorded the prediction accuracy of the test set as 97.59%.

The weights of these two constraint networks are transferred to the corresponding part in the main CMG model. These constraint networks in CMG are frozen when training CMG and predicting a new molecule using it.

2.2 Single Objective Optimization(SOO)

We use 4 layers and 8 heads of self-attention and feed-forward layers for both the encoder and the decoder. The hidden vector size is 128 and the dimension of the intermediate dense layer is 256. We set the maximum sequence length to be 150 because the max length used in the previous self-attention based molecule representation model [3] was 100 and the default buffer size of a typical Transformer model is 50% of its maximum sequence length. For the two constraint networks, we use the same configuration as pre-training models of them so that they are compatible with each other when transferring the weights. We use Adam optimizer [1] with the learning rate=2.0,

[1] `https://www.tensorflow.org/api_docs/python/tf/keras`

$\beta_1 = 0.9$ and $\beta_2 = 0.997$. We train the proposed model (CMG) using 10M of the training set for 500 epochs with a batch size of 4,096. The dimension of the property vector is three, where the first one is PlogP, the second one is QED, and the last one is DRD2 values. We use the desired property vector of $\{X_{P\log P}, 0.0, 0.0\}$ with the sampled offset parameters of $\alpha = \{-1.0, -0.5, 0.0, 0.5, 1.0\}$, $\beta = \{0.1, 0.6\}$ and $\gamma = \{0.52, 0.8\}$.

2.2.1 Training Details

We train one model and use it for SOO and MOO. For the ablation study, we additionally train three other configurations and the following table shows their training times. Since SimNet processes two sequences, the whole model with it takes longer for training compared to the whole model with only PropNet. We train the proposed models using a single V-100 GPU.

Configurations		Training Time
PNet	SNet	
☐	☐	23 Hour
☑	☐	26 Hour
☐	☑	29 Hour
☑	☑	32 Hour

Table 1: **Training Time**: PNet is PropNet and SNet is SimNet.

2.3 Multi-Objective Optimization (MOO)

We use the same configuration and the same training sets used in the previous task, SOO. We select the best model based on the development set (985 samples). When predicting a new molecule, we sample 60 molecules per input for a fair comparison to the baselines. In this task, we use the desired property vector of $\{X_{P\log P}, 0.0, 0.0\}$ and the sampled offset parameters, $\alpha = \{1.0, 2.0, 3.0\}$, $\beta = \{0.91, 0.94, 0.97, 1.0\}$, and $\gamma = \{0.51, 0.6, 0.7, 0.8, 0.9\}$.

2.4 Case Study

: We use the same configuration and the same training sets used in the two previous tasks, SOO and MOO. We select the best model based on the development set of DRD2 task provided by VJTNN. In this task, we use the desired property vector of $\{X_{P\log P}, 0.0, 0.0\}$ and the sampled offset parameters, $\alpha = \{0.0\}$, $\beta = \{-0.1, -0.05, 0.0, 0.05, 0.1\}$, and $\gamma = \{0.6, 0.7, 0.8, 0.9\}$. We allow 20 generated samples for the proposed method and VJTNN, while we evaluate all states of MolDQN samples.

References

[1] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[2] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[3] Bonggun Shin, Sungsoo Park, Keunsoo Kang, and Joyce C. Ho. Self-attention based molecule representation for predicting drug-target interaction. In *Proceedings of the 4th Machine Learning for Healthcare Conference*, volume 106 of *Proceedings of Machine Learning Research*, pages 230–248. PMLR, 09–10 Aug 2019.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.