Pic2SMILES: Learning Chemical Representations from Molecular Diagrams

Hanif Leoputera Lim
University of California, Los Angeles
hanifleoputeralim@gmail.com

Wilson Jusuf
University of California, Los Angeles
wilsonjusuf1998@gmail.com

Varun Narayanan
University of California, Los Angeles
vchakravarthy21@ucla.edu

Abstract

Our goal is to predict the widely-used and standardized SMILES representation of a chemical molecule given its molecular diagram. We experimented with different convolutional, and transformer models on two different tasks: Molecule Semantic Segmentation and SMILES Prediction.

*Using a synthetic data set of 1 million molecular images, we managed to obtain **75% overall testing accuracy** and **79% testing accuracy on small molecules** using a transformer model in the SMILES prediction task, and a **77% pixel accuracy** in the molecular segmentation task using a pretrained HRNet. We find that the model does well with (more-common) smaller molecules. With this, we are confident that the network could extract meaningful molecular information to assist in classroom and lab settings for students and researchers.*

1. Introduction

The Simplified Molecular-Input Line-Entry System (SMILES) is a line notation commonly used in many molecule editors in research. Atom are represented by their symbols e.g. C, and bonds can be represented using `(., -, =, #, $, :, /, \)`. More complex structures such as rings can be represented by symbol repetition as well. [4]

Translating molecular diagrams into their SMILES representation is a challenging problem, particularly for large and complicated molecules. This is due to a couple reasons. One is that SMILES representation is non-unique; One could start interpreting from different elements in the diagram (obtaining different SMILES strings) and still get the same molecule. Another issue, which is most problematic, is that there exists different nomenclatures to name a molecule (this is also another reason why there are many legitimate SMILES representation for the same molecules).

Yet, the ability to convert into SMILES representation remains attractive especially in drug discovery. This is because most molecular databases store information in SMILES representation. Most publications containing data related to molecules do not provide the molecular structures in a computer readable format. Instead, they provide computer generated images of the molecules in those documents. Since only images of structures are published, people would resort to the tedious manual redrawing of structures in chemical rendering software to convert it into computer readable formats, ready-to-use in downstream computation and analysis tasks.

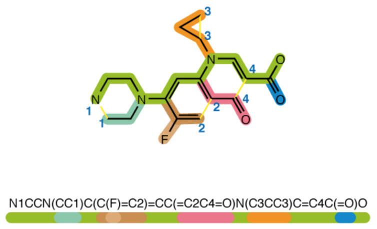

Figure 1. The molecular diagram of Ciprofloxacin ($C_{17}H_{18}FN_3O_3$), and its SMILES. Notice the correspondence of the SMILES string to the drawing.

1.1. Past work

To the best of our knowledge, there are currently 2 solutions commonly used for this problem. First is rule-based OCR. This is currently the best method in most situation, however its performance degrades sharply when input quality is low and when there exists irrelevant information on the background. Moreover, rule-based systems are necessarily highly interdependent and complex, and requires extensive domain expertise to maintain. The second method is by using LSTM. LSTM networks have shown significant promise in encoding chemical diagrams due to their ability to model higher-level structure and temporal order. [7] Yet, the limited temporal extent of LSTM restricts the structural complexity of chemical diagram that may be accommodated in sequence embeddings. In the language modeling domain, this shortcoming has been addressed through the emergence of Transformer networks, in which slot masking enhances the ability to learn longer term temporal structure of the chemical diagram [14].

1.2. Our approach

Thus, this paper proposes the use of transformer to encode-and-decode chemical diagrams into their SMILES representation. Moreover, to make our model robust against low input quality and perturbations, we designed a molecule segmentation model that masks everything but the chemical structure using pretrained segmentation encoders.

2. Dataset

We first decided on a resolution of 200×200 to render our images. This dimension size ensured that we best emulate real-life handwriting conditions and photographs with low resolution. Also, we have an added bonus of reduced training time.

2.1. SMILES string Preprocessing

For the smiles prediction task, we preprocessed all smiles strings into a `deepsmiles` representation [9]. A Deepsmile representation is a SMILES-like syntax suited to machine learning. Concretely,

- Rings are indicated using a single symbol instead of two
- branches do not use matching parentheses but rather use a right parenthesis as a 'pop' operator. This allows for a more compact representation by eliminating the requirement of matching parentheses.

For example, benzene is `c1ccccc1` in SMILES but `cccccc6` in DeepSMILES (where the 6 indicates the ring size). Also, the SMILES `C(Br)(OC)I` can be converted to the DeepSMILES `CBr)OC))I`, which shows the 'pop' operator.

2.2. Rendering

We obtained a dataset of 26 million SMILES strings freely-available from emolecules.com [2]. We then used the open-source chemistry drawing tool RDKIT [3] to synthetically generate 1,000,000 grayscale molecule images with a 200×200 dimension and a white background.

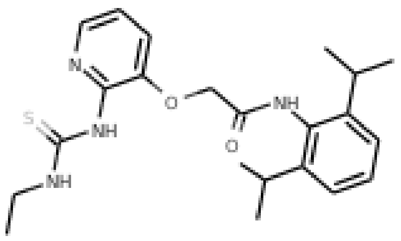

Figure 2. An example of a molecule's ground truth image.

2.3. Introducing noise to dataset (segmentation)

To augment the dataset with noise, we introduced lined-paper lines. Specifically, we introduced the following random parameters to the synthetically generated images:

- Orientation of Line: Vertical, Horizontal or both (grid)
- line spacing (width)
- line color (Green, grey or red)
- Line Opacity

The example shown in the previous section can now be seen with red lined paper as a noise:

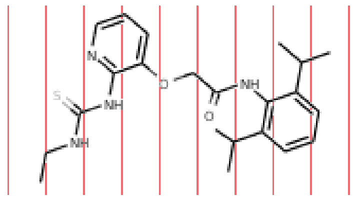

Figure 3. An example of the noise (red vertical lines) added onto the ground truth image shown before

We show augmentation variety with additional examples below:

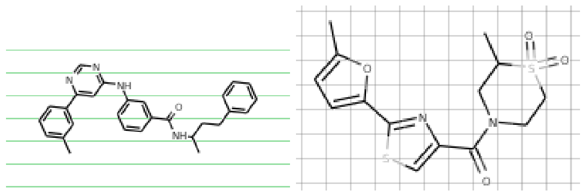

Figure 4. More examples of graph-paper lines. Notice the difference in opacity between the previous red lined paper example.

3. Architecture of Models & Training

Note that all models are implemented in pytorch [10].

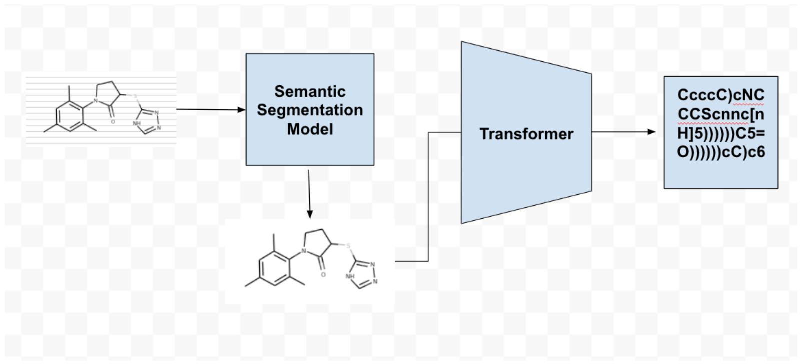

Figure 5. Our model's workflow. We use a semantic segmentation model to predict a mask. Then, apply the mask to the original image (to remove all pertubations), and feed this into the transformer model to obtain the deepsmiles.

3.1. Molecular Segmentation

We use an encoder-decoder format with a bottleneck reparameterization layer, inspired by Kingma's Variational Bayes paper [8]. We experiment with the following pre-trained models as encoders:

1. . `VGG16`
2. . `HRnet`

We adopt VGG16 [12] as the baseline model due to its reputation as feature extractor for many computer vision tasks in general. We also adopt HRnet, which is specialized for crowded scene semantic segmentation in high-resolution images [15].

The architecture is as follows: Note that D and BN indicate a dropout and batchnorm layer respectively. Also, a HRnet Basic Block consists of three (conv-relu-pool) blocks with a downsampled residual connection. Furthermore, the `ConvTranspose2D` layer is a pytorch layer that implements a 2D transposed convolution over an input image with multiple channels [10].

We use the following parameters to train both models using 1,000,000 images:

- Epochs: 4
- Split: 80% training, 20% testing
- Batch-Size: 50
- Optimizer: Adam with learning rate 1×10^{-5}, $\beta_1, \beta_2 = 0.9, 0.999$
- Dropout 0.2

After testing several configurations, that these hyperparameters gave us the best results. The unusually small learning rate 1×10^{-5} was used for a conservative and steady gradient descent that did not overshoot. The loss is computed using pixel-wise Binary Crossentropy of the predicted image mask $\hat{y}^{(i)}$ and the actual image mask $y^{(i)}$ for all $i \in \{1, \ldots, n\}$:

Layer	**VGG model** (output dims)	**HRnet Model** (output dims)
input	RGB Image (3, 200, 200)	
feature extractors	*VGG-16* 4096	*HRnet* (720, 50,50)
Additional encoding	$2\times$ (*Linear*) $2048 \rightarrow 1000$	$4\times$ *HRnet Basic Block* channel progression: 360, 180, 90, 25
		UnFlatten & Linear 1000
Reparam. Bottleneck	*Linear* 750	
Linear decoding	$3\times$(*Linear, D,BN*) $1000 \rightarrow$ BN $\rightarrow 2048 \rightarrow$ BN,D $\rightarrow 4096$	
Deconvolution	$9\times$(`ConvTranspose2d`, *ReLU*) `(kernel_size, stride)` $\in$ `(5,2),(5,1),(4,1),(2,1)` Channel progression: 800,400,100,80 70,60,50,32,1	
	Sigmoid (200,200)	
output	Image Mask (200,200)	

$$\text{pixel\_loss}(x, \hat{x}) = x \log(\hat{x}) + (1 - x) \log(1 - \hat{x}) \tag{1}$$

$$L(y, \hat{y}) = -\frac{1}{NWH} \sum_{i=0}^{N} \sum_{w}^{W} \sum_{h}^{H} \left(\text{pixel\_loss}(y_{w,h}^{(i)}, \hat{y^{(i)}}_{w,h}) \right) \tag{2}$$

Note that $0 \leq \hat{y}^{(i)}, y^{(i)} \leq 1$.

3.2. SMILES Transformer

Our transformer model is largely inspired from [16]. We adopted a bare-bones transformer model from [1].

After the molecular structure has been masked out, we'll extract the molecular diagram's features using a pre-trained resnet50 and we used pytorch's `adaptive_avg_pool2d` to get a 14×14 soft and local attention for each pixels. We chose resnet as it has been shown to perform best in image-captioning task by [14].

These attention features will then be passed into our transformer model as per Figure 5.

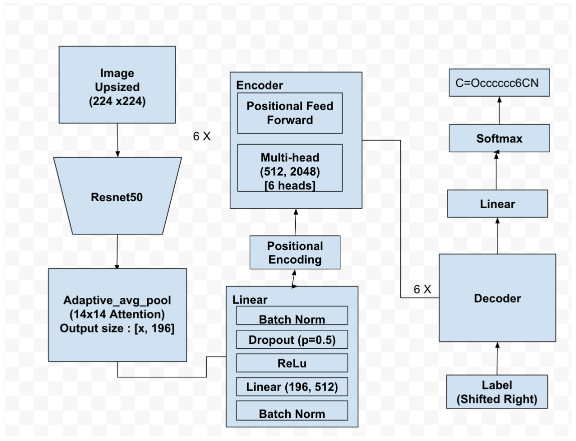

Figure 6. Diagram of our transformer architecture

Our transformer is trained using the following hyperparameters:
1. Input-encoding-size: 512
2. Input-decoder-size: 2048
2. Dropout-rate: 0.5
3. Attention-head: 6
4. Encoding-Decoding Layer: 6
5. Batch-size: 10
6. Epoch: 20
7. Noampt-warmup: 20,000
8. Temperature: 0.6
9. Learning Rate: 5e-4

From our experiment, an input-encoding-size of 512, input-decoding-size of 2048 and dropout 0.5 seemed to provide the best performance. From [14], more attention heads and a deeper encoding-decoding layer would certainly provide better performance due to model ensembling. However, we found 6 to be optimal for attention-head and encoding-decoding layers, as more than that, the training time took too long. We also chose a batch-size of 10 as that's the largest possible batch-size before we ran out of vRAM. Moreover, we chose a decoding temperature (which ranges from 0-1.0) of 0.6 so that our model decodes more "safely", picking tokens that our models are more confident on.

We use Adam optimization as per the suggestion of [16], with the following parameters $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. Also following [16], we increased our learning rate linearly before our number of iterations reaches `Noampt-warmup`. Thereafter, we decrease the learning rate proportionally to the inverse square root of the step number.

To evaluate our model's loss, we used KL-divergence between our predicted sequence and the ground truth. Thus, the loss is calculated using this formula

$$p(y|x_i) = \begin{cases} 1 \text{ if } y = y_i \\ 0 \text{ otherwise} \end{cases} \tag{3}$$

$$q_\theta = \frac{exp(z_{y_i})}{\sum_{j=1}^{K} exp(z_{y_j})} \tag{4}$$

$$L = -\sum_{i=1}^{n} \sum_{y=1}^{K} p(y|x_i) \log q_\theta(y|x_i) \tag{5}$$

The loss function essentially measures how much the predicted sequence one-hot encoding (from its `log_softmax`) diverges from the ground-truth one-hot encoding.

Moreover, we implemented a Self-Critical Policy Gradient inspired from [11] that will "reward" our model (give lower losses) if it minimizes the edit distance, based on the following equation.

$$\nabla J = \mathbb{E}_{x \sim p(s)} \mathbb{E}_{y \sim \pi(y|x)} \nabla log\pi(y|x) \cdot (R(x,y) - b(x)) \tag{6}$$

Reward R(x,y) is a negative edit distance (since we minimize it). The baseline b(x) represents how well the model fares on word x. In practice, this means that we compute baseline as a score of greedy translation, $b(x) = R(x, y_{greedy}(x))$.

As per [11], this loss evaluation will be triggered at later stage of training, replacing KL-divergence. In our experiment, self-critical training have exhibited better accuracy score; however, under self-critical training, our GPU only has enough vRAM to train with batch size of 1. Thus, due to time and resource constraint, we weren't able to test the extent to which self-critical training improves our model's accuracy.

4. Results

To conduct the experiments, we used an Nvidia RTX 2070 Graphics Card to accelerate training.

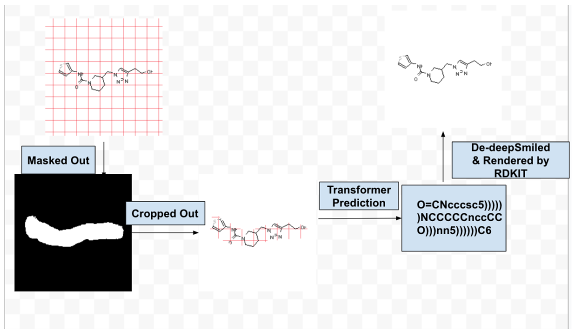

Figure 7. An actual predicted result through our model

4.1. Segmentation Results

For the segmentation task we evaluated the model's pixel accuracy. Concretely, for an image mask $y \in \{0,1\}^{W\times H}$ and its predicted mask $\hat{y} \in \{0,1\}^{W\times H}$, it is:

$$\text{pixel\_acc} = \frac{\sum_w^W \sum_h^H [y_{w,h} == \hat{y}_{w,h}]}{WH} \qquad (7)$$

where the indicator function $[\dots]$ returns 1 if true, 0 otherwise.

Using an optimal threshold value of 0.085 to create the binary masks from the predicted masks, we obtained the following testing statistics: As seen in the predicted result of

Encoder model	Pixel Accuracy
VGG16	70.4%
HRNet	77.1%

Table 1. Segmentation testing results on a test set of 200,000, training set of 800,000

figure (7), our segmentation model does very well in obtaining the overall molecule outline. However, it does not capture specific molecular lines inside the outline shape. This is shown by the existence of the red-grid line noise in figure (7).

4.2. SMILES prediction

We use norm-edit distance to evaluate our model's prediction accuracy, which is calculated using the following equation

$$\text{Norm-Edit Dist} = \frac{\text{ed}(w, w^*)}{\max(\text{len}(w), \text{len}(w^*))}$$

Edit-distance $\text{ed}(w, w^*)$ measures the number of insertions and deletions to make word w equals to word w^*.

Broadly speaking, norm-edit distance measures the proportion of errors between predicted SMILES from the ground truth.

Our model overall perform pretty well at 70% and a further improvement to 75% with Self-Critical Training as seen in Table 2.

Model	Normalized Edit Distance
Baseline	70.3%
+Self-Critical Policy	75.6%

Table 2. SMILES prediction results on a test set of 300,000

Furthermore, our model seems to perform better with smaller molecules and even perform just slightly worse than the state-of-the-art for longer and more complicated molecules as seen in Table 3.

Molecule Length	Normalized Edit Distance
≤ 10	79.1%
11-29	77.3%
≥ 30	73.5 %

Table 3. SMILES prediction results (with self-critical training) segmented by molecule lengths.

5. Discussion

Although our model's performance fell short of the state of the art [13], we obtained a comparable performance to it despite using only 1 million data points vs 57 million. Moreover, due to time constraints, we weren't able to test the limits of our model under self-critical training. Additionally, currently we're using resnet-50 pre-trained on ImageNet Dataset - a completely different domain from molecular diagrams. So, there could be some domain adaptation issue, lowering the quality of features extracted. Another problem is currently our transformer encodes position at a pixel level, introducing a lot of noise during the self-attention of the transformer.

In the future, we propose training a bottomUp, topDown Attention model [5], which will cluster relevant pixels into image regions (marked by boxes). We can then use these box coordinates to encode position in the transformer inspired by [6]. This will allow a more meaningful feature-level self-attention mechanism in the transformer. However, for this approach, we'll have to make our own data set, marking every relevant features (element letters, bond types, corners). This is, unfortunately, infeasible within this project's time frame, but certainly a worthwhile long-term project.

For semantic segmentation, we propose using a deeper Deconvolution network and potentially getting rid of a flattening and linear bottleneck layer. We predict that this would maintain the network's translational invariance to detect various line noise positions and orientations within the molecule's outline boundary and increase pixel accuracy.

References

[1] Annotated transformer. `https://nlp.seas.harvard.edu/2018/04/03/attention.html`. Harvard's NLP Group guided code for Transformer.

[2] emolecules.com. `https://www.emolecules.com/info/products-data-downloads.html`. site to download SMILES dataset. Accessed 2020 Mar 16.

[3] Rdkit: Open-source cheminformatics. `http://www.rdkit.org`. Accessed 2020 Mar 16.

[4] Smiles - a simplified chemical language. `https://www.daylight.com/dayhtml/doc/theory/theory.smiles.html`. Accessed 2020 Mar 16.

[5] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang. Bottom-up and top-down attention for image captioning and visual question answering. *CVPR*, abs/1707.07998, 2017.

[6] S. Herdade, A. Kappeler, K. Boakye, and J. Soares. Image captioning: Transforming objects into words. *CVPR*, abs/1906.05963, 2020.

[7] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[8] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In Y. Bengio and Y. LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[9] N. O'Boyle and A. Dalke. Deepsmiles: An adaptation of smiles for use in machine-learning of chemical structures, Sep 2018.

[10] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[11] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel. Self-critical sequence training for image captioning. *CVPR*, 2017.

[12] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[13] J. Staker, K. Marshall, R. Abel, and C. McQuaw. Molecular structure extraction from documents using deep learning. 2019.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[15] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao. Deep high-resolution representation learning for visual recognition. *TPAMI*, 2019.

[16] J. Yu, J. Li, Z. Yu, and Q. Huang. Multimodal transformer with multi-view visual representation for image captioning. *JOURNAL OF LATEX CLASS FILES*, abs/1905.07841, 2019.