Comparison between SMILES-based Differential Neural Computer and Recurrent Neural Network architectures for de novo molecule design

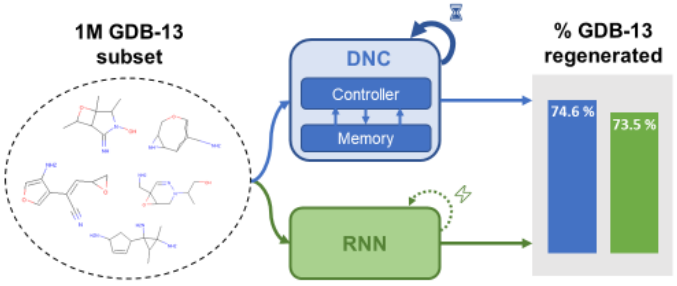
Simon Johansson[§], Oleksii Prykhodko[§], Josep Arús-Pous[§⊥], Ola Engkvist[§], Hongming Chen[§]*

§ Hit Discovery, Discovery Sciences, R&D, AstraZeneca Gothenburg, Sweden

⊥ Department of Chemistry and Biochemistry, University of Bern, Freiestrasse 3, 3012 Bern, Switzerland.

* Corresponding author: simon.johansson@astrazeneca.com

Abstract: In recent years, deep learning for de novo molecular generation has become a rapidly growing research area. Recurrent neural networks (RNN) using the SMILES molecular representation is one of the most common approaches used. Recent study shows that the differentiable neural computer (DNC) can make considerable improvement over the RNN for modeling of sequential data. In the current study, DNC has been implemented as an extension to REINVENT, an RNN-based model that has already been used successfully to make de novo molecular design. The model was benchmarked on its capacity to learn the SMILES language on the GDB-13 and MOSES datasets. The DNC shows improvement on all test cases conducted at the cost of significantly increased computational time and memory consumption.



Introduction

Machine learning (ML) and particularly Deep Learning (DL) are on the rise. DL has been successfully used to tackle problems were previously regarded as difficult to handle such as image classification [1], face recognition [2] or playing Go [3]. But the most interesting recent development are DL generative models, which can create content similar to what it has been trained with. They have been applied to fields such as image generation [4], language translation [5] as well as molecular design [6]. The last one is especially interesting because generative models trained with a set of molecules have shown to generate novel and meaningful compounds that are not present in the training set but share similar physicochemical properties with them. Furthermore, focused compound sets can be obtained by using advanced techniques such as transfer [6] or reinforcement learning [7]. A big advantage of generative de novo molecule design methods is that are data driven capable of learning the underlying probability distribution over a large set of chemical structures. The search over the chemical space can be reduced to only molecules seen as reasonable, without introducing the rigidity of a rule-based approach. One of the most mature architectures available today is using a Recurrent Neural Network (RNN) that generates molecules in the SMILES notation and uses Long Short-Term Memory (LSTM) [8] cells to keep track of the cyclic structure of molecules. Other architectures have also shown good results, such as Variational AutoEncoders (VAEs) [9, 10], Generative Adversarial Networks (GANs) [11, 12], Graph Generative Models (GGMs) [13, 14], etc. [15].

In 2014 the Neural Turing Machine (NTM) was introduced by DeepMind [16]. The model was inspired by the architecture of the modern computer. Later, the differentiable neural computer architecture (DNC) was introduced [17] and shown to vastly outperform RNNs. By

combining a flexible-sized memory with a controller neural network, it can process information and store the results into the external memory. The model learns what to write and erase from the memory as well as what and when to read from it. In molecular generation, the first application of DNC, RANC [18], was proposed as an improvement of ORGANIC [11]. Result showed that SMILES sequences generated by RANC were longer, more diverse and more complex than in ORGANIC.

However, a systematic analysis of the DNC architecture in molecular generation using the SMILES notation has not been reported in literature. In this study, we have compared the output chemical space generated by a basic LSTM-based RNN model and an enriched model with a DNC. To be able to perform this analysis, models were trained with a subset of GDB-13 [19], a database that enumerates most fragment-like molecules up to 13 heavy atoms in size. After training the models were sampled and by assessing how much of the entire database was generated the quality of the models could be determined [20]. Additional models were also trained using the randomized SMILES variant, which has been shown to perform better than the canonical SMILES [21]. Furthermore, to train models that generate drug-like molecules, models were also trained and evaluated using the MOSES framework [22].

Methods and materials

Differentiable Neural Computer

A differentiable neural computer has three essential parts: a controller, a memory, read and write heads (Figure 1). The controller has the same role as a processor in a computer. It processes input information and stores the results into the memory. Storing is done using the

write heads and stored information is loaded using read heads. The model learns what to write and erase from the memory as well as what and when to read from it.

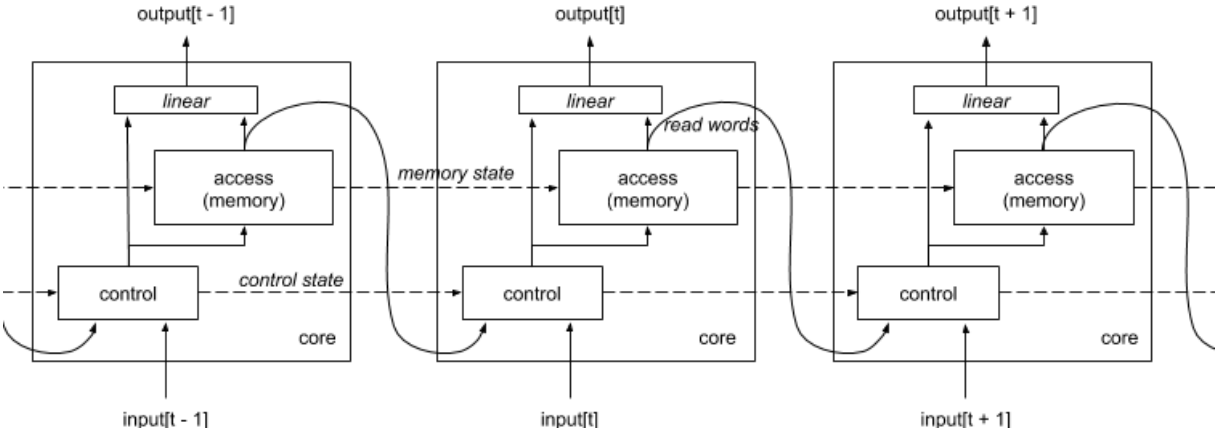


Figure 1: Schematic representation of the differentiable neural computer. Obtained from the GitHub repository of Google DeepMind DNC. [23]

Controller

The controller is responsible for taking an input in, reading from and writing to the memory, and producing an output. Any RNN can be defined as a controller for DNC. At every time step, the controller emits an output that goes through an interface layer, which decides the extent of each memory operation. For writing, it can choose to store information at a new, unused location or at a location that already contains information. This allows the controller to update what is stored at the location. If all the locations in the memory are used up, the controller can decide to free the locations, much like how a computer can reallocate memory that is no longer needed.

Memory

A memory of a DNC is represented by an $N \times M$ matrix where a value may be stored for later retrieval. The memory can be searched based on the content of each location. Writing into the memory creates associative temporal links that can be retraced backwards to build the

computational graph and consequentially perform backpropagation and train the memory. The read-out information can be used to produce answers to questions or actions to take in an environment. Together, these operations give DNCs the ability to make choices about how they allocate memory, store information in the memory, and easily find it.

Read and write operations in memory

The role of the write and read heads is to store the information in the memory and obtain it when it is later needed. To do that, the heads use 3 different forms of differentiable attention. The first one is content lookup; whose goal is to provide a flexible mechanism of navigating through the memory by comparing a key vector emitted from the controller to the content of each memory location using a similarity score. The second attention mechanism records transition between consequently written memory location in a $N \times N$ temporal link matrix where N is the size of the memory. This allows DNC to recover sequences in the order that it wrote them. The third form is designed to allocate the memory for writing. The usage of each memory cell is represented as a number in $[0,1]$ and a weighting that picks up unused locations is delivered to the write head. After each write the usage vector increases its value as well as decreasing over time. This allows the controller to reallocate and delete the information that is no longer required.

RNN model

The baseline RNN closely follows that of the best model found in [21]. It is an LSTM with 3 hidden layers of size 512, with an embedding layer of 128. The batch size for all models used was 512. For the DNC, the controller used the same settings as that of the baseline.

Datasets

The GDB-13 database is a publicly available collection of all chemically stable molecules with up to 13 heavy atoms of types C, N, O, S and Cl (hydrogen atoms are implicit). The number of SMILES strings that can be parsed in RDKit version 2018.09.1.0 [24] in the database consists of 975,820,187 compounds [25]. The training set is a randomly sampled subset of size 1 million, with a separate validation set of size 10,000. dataset contains 24 different tokens excluding the start and end tokens.

The dataset used for the MOSES [22] benchmark was downloaded from their GitHub repository, and is based on a custom curation of the ZINC [26] Clean Leads collection. After filtering, the training dataset contains around 1.6 million compounds, with two separate test sets of around 176,000 each: a test set and a scaffold test set, which includes scaffolds not present in the training set.

Training process

When training on the GDB-13 dataset, the learning rate strategy differed between the canonical and the randomized smiles. For the canonical SMILES, a starting learning rate of $5 \cdot 10^{-4}$ was used, which was halved every 15 epochs for 165 epochs. For the randomized SMILES, the same starting learning rate was used, but instead it was lowered by 20% of its current value every 35 epochs for 300 epochs. The more aggressive reduction of learning rate on canonical SMILES was done to counteract the overfitting that occurred, whilst the randomized SMILES do not encounter the same problem, as we don't feed the same SMILES to the model every epoch. Following [21], the RNN baseline used a dropout of 0.25 for the random smiles which also turned out to be the best setting for the DNC model in this case too. For the

canonical smiles, a dropout of 0.5 gave the best results. The DNC parameters used were 32 memory cells of length 20 and 8 read heads for both SMILES types.

MOSES models were trained with a starting learning rate of $5 \cdot 10^{-4}$, which was halved every 30 epochs and trained for a total of 100 epochs. The concept of UC-JSD does not exist in the same fashion for this data set, as we do not have a validation set. It would be possible to use one of the MOSES test sets to get the best possible results, but this was omitted as this would compromise the integrity of the benchmark and artificially increase the probability of higher benchmark score. Both models used a dropout of 0.5, in accordance with our experience of canonical SMILES from GDB-13. For the MOSES models, the canonical models used 8 memory cells of length 20 and 8 read heads. The ADAM optimizer [27] was used in all models with default parameters $\beta_1 = 0.9,\ \beta_2 = 0.999, \epsilon = \ 10^{-8}$.

Obtaining the best epoch

During the training process, the Uniformity-Completeness Jensen-Shannon Divergence (UC-JSD) metric [21] of each model was monitored to obtain the best epoch, since it was shown to have high negative correlation with the learning capabilities on the GDB-13. This metric is calculated as the Jensen Shannon Divergence (JSD) between the raw Negative Log-Likelihoods of the training set, validation set and a sampled set at a given time of the training process (e.g., after finishing an epoch).

Once the models were trained and the best epoch assessed, $2 \cdot 10^9$ compounds were sampled from models to examine the coverage of the whole GDB-13 database [25] from each model.

Measuring significance of model performance

It was shown in [21] the statistically significant level for the difference of coverage between different models using confidence intervals, where the significant difference between the GDB-13 coverages C_1, C_2, can be expressed as

$$C_1 - C_2 \geq 4Z_\alpha \sigma^*,$$

(Equation 1)

Where, using the sample size $k = 2 \cdot 10^9$, we get the value $\sigma^* = 8.952 \cdot 10^{-6}$. Z_α is computed using Student's t-distribution with one degree of freedom.

Technical notes

The DNC implementation used in this research was based on already existing code [28]. The software was coded in Python 3.6.8. Models were run using PyTorch 1.0 [29] on Nvidia Tesla K80 (Kepler) using CUDA 9.1 driver 390.87 in the GDB-13 canonical SMILES models, and on Nvidia Tesla V100 (Volta) using CUDA 9.1 driver 390.30 on the randomized SMILES and MOSES models.

Results and discussion

Performance comparison on models for GDB-13 database

DNC and regular RNN were trained on the GDB-13 training set for both canonical and randomized SMILES. During the training, the UC-JSD was constantly monitored as the model quality metric for comparison. Results show (Figure 2A-B) that the DNC model has lower UC-JSD than the RNN model until around epoch 50 on the canonical SMILES, after which the

curves jump above that of the RNN LSTM models. This implies that some over-fitting existed in the DNC model that started around epoch 35. Regarding the percent of valid molecules (Figure 2C-D), all models achieve high percentage of valid SMILES after training around 25 epochs. The DNC models have slightly higher validity than that of the RNN models for both types of SMILES, but the differences are unremarkable. Like [21], $2 \cdot 10^9$ compounds were sampled from the trained models and the coverage of GDB-13 for the respective model is presented in Table **1**.

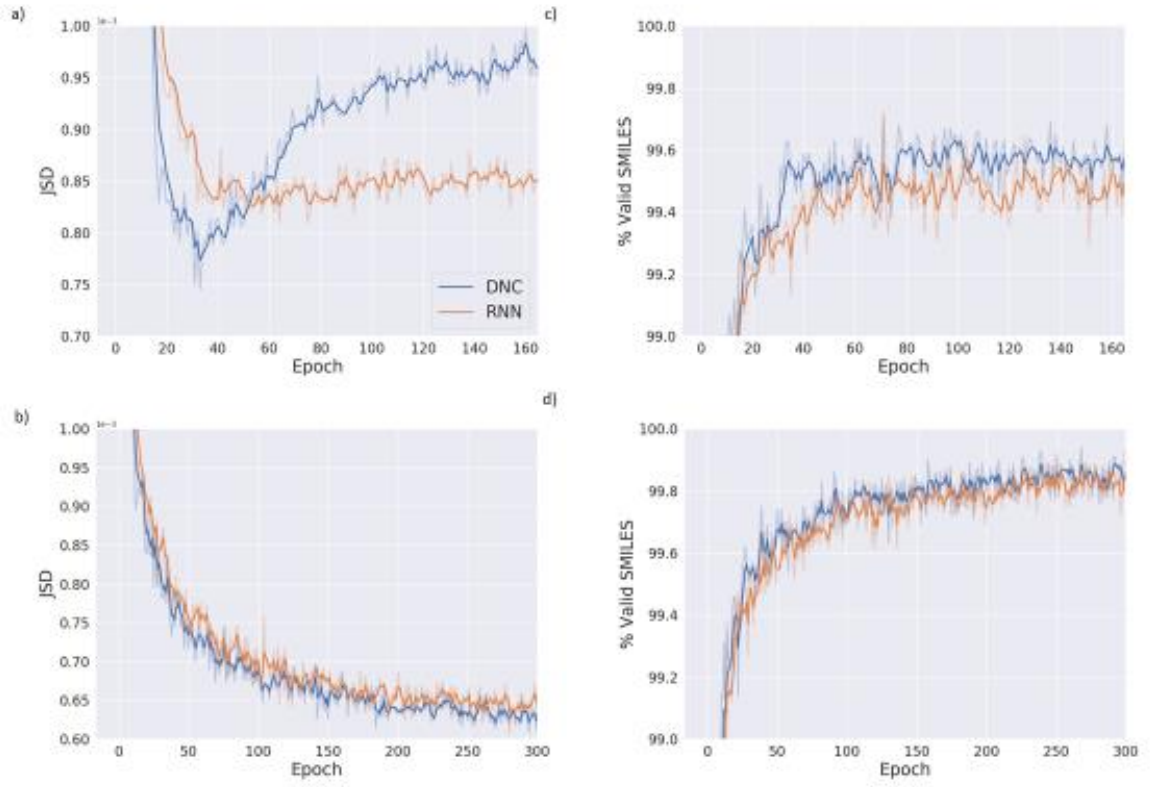


Figure 2: (a) and (b) Plots of the UC-JSD metric during training for the DNC and RNN models. Models are trained with canonical SMILES (a) and randomized SMILES (b). **(c) and (d)** Plots of the % valid SMILES strings generated from a 10,000 sample during training for the DNC and RNN models. Models are trained with canonical SMILES (a) and randomized SMILES (b).

Notice that both RNN and DNC models trained with randomized SMILES have higher validity than those of the models trained with canonical SMILES, as well as higher coverage of the

GDB-13 chemical space. This is consistent to our previous report [21] and demonstrates that training with randomized smiles is an efficient data augmentation strategy for achieving better generalization.

Model	% Valid	IN GDB-13	% uniq. GDB-13	Max. % uniq. GDB-13	UCC
RNN Can.	99.44	1.71 B	73.43	82.65	0.64
DNC Can.	99.46	1.71 B	74.57	82.67	0.66
RNN Rand.	99.82	1.83 B	82.80	84.61	0.85
DNC Rand.	99.85	1.83 B	82.89	84.67	0.85

Table 1: Comparison sampling results of models trained on the GDB-13 dataset. **Legend:** Model name (Model); Percent of valid SMILES (% valid); Total number of molecules in GDB-13 including repeats (IN GDB-13); Percent of unique molecules in GDB-13 compared to the total of the database (% uniq. GDB-13); Maximum percent of molecules in GDB-13 that would be sample if the model were ideal (see [20]) (Max. % uniq. GDB-13); Ratio of the uniformness, completeness and closedness of the model [21], the closer to 1 the better (UCC).

The difference in coverage between the models at the respective SMILES variants can be considered significant using the conservative confidence interval described in Equation 1. In general, our results demonstrate that the DNC models have better performance than the regular RNN models in terms of learning the underlining SMILES character distribution and model coverage for the GDB-13 database.

Performance on the MOSES benchmark

Results indicate that, as with the GDB-13 models, the DNC model performs at the same level or slightly better than the RNN model (Table 2). Both models perform ostensibly better than the baseline in most metrics but without any significant difference between them. For

instance, the Fréchet ChemNet Distance [30], a measure that compares how bioactive molecules from the training and generated sets are, is at least half that of the previously best model (VAE). There are two exceptions: validity and scaffold similarity score to the TestSF set. In the first case this is because of the architecture of the Junction Tree VAE [31], which is never able to generate an invalid molecule. On the other hand, the scaffold similarity score can be lower because the models trained here are slightly biased to the training set. The distribution of the auxiliary metrics compared against the test set is provided as supplementary information and shows the same trend. Both models are better than the baseline on almost all metrics, but there is no clear difference between them. Lastly, models trained with randomized SMILES show a much better performance than those trained with canonical SMILES [21] and even have in some cases better values than the baseline. For instance, the FCD is half that of the canonical SMILES models and the internal diversity is even better than the baseline. This may indicate some limitations on the scope of the benchmarking suite.

Difference in training time

The tests conducted show varying degrees of improvement, from quite substantial (on canonical GDB-13) to minor (randomized SMILES GDB-13 and MOSES). This improvement comes at the cost of increased computational time and memory consumption (Table 2). Models trained with a DNC take up to three times more to compute compared to the regular RNN.

DB	SMILES	Model	Total time	Sampled epoch	Time @ sampled epoch
GDB-13	Canonical	RNN	11:58	75	5:26
		DNC	32:09	33	6:26
	Randomized	RNN	23:37	280	22:05

		DNC	61:35	294	60:25
MOSES	Canonical	RNN	15:54	100	15:54
		DNC	52:25	100	52:25
	Randomized	RNN	73:02	403	64:13
		DNC	169:05	282	159:05

Table 2: Time required to train the different models. Legend: Database used (DB), SMILES variant (SMILES), Model Architecture (Architecture), Time at which it finished training in full (Total time), epoch where the lowest UC-JSD (Best epoch), Time taken to train up to the best epoch (Best time). Time is in hh:mm. The models trained on the MOSES canonical training set was sampled at the final epoch, rather than using any heuristic for best epoch.

Model	FCD (↓)		SNN (↑)		Frag (↑)		Scaf (↑)		Valid (↑)	IntDiv (↑)	IntDiv^2 (↑)	Filters
	Test	TestSF	Test	TestSF	Test	TestSF	Test	TestSF				
Train	0.0080	0.4755	0.6419	0.5859	1	0.9986	0.9907	0.0	1	0.8567	0.8508	1
CharRNN	0.0914	0.5429	0.579	0.5486	0.9998	0.9984	0.9184	**0.1289**	0.9598	0.8566	0.8506	0.9897
AAE	0.3945	1.0003	0.6197	0.5747	0.9952	0.9939	0.8655	0.1001	0.9965	0.8565	0.8503	0.9974
VAE	0.0844	0.5412	0.6226	0.5766	0.9996	0.9982	0.9331	0.0616	0.9691	0.8565	0.8505	0.9963
JTN-VAE	0.4224	0.9962	0.5561	0.5273	0.9962	0.9948	0.8925	0.1005	**1**	0.8512	0.8453	0.9778
RNN	0.0420	0.4988	0.6300	0.5817	0.9998	**0.9987**	0.9579	0.0621	0.9940	0.8567	0.8508	0.9977
DNC	0.0430	0.4945	**0.6327**	**0.5838**	0.9999	0.9985	0.9564	0.0590	0.9946	0.8567	0.8508	**0.9978**
RNN Rand.	**0.0212**	**0.4715**	0.6233	0.5788	**1.000**	0.9986	0.9750	0.0948	0.9955	0.8562	0.8503	0.9977
DNC Rand.	**0.0212**	0.4970	0.6224	0.5778	0.9999	0.9985	**0.9755**	0.0887	0.9949	**0.8568**	**0.8509**	0.9977

Table 3: Results of the MOSES benchmark [22] on the DNC and RNN models alongside the baseline model results. Arrows indicate when a given metric has to be minimized (↓) or maximized (↑). **Legend**: Model trained (Model), Fréchet ChemNet Distance (FCD), average similarity of generated molecules to the nearest molecule from the test set (SNN), cosine distance average between vectors of fragment frequencies (Frag), cosine distance average between vectors of scaffold frequencies (Scaf), ratio of valid molecules (Valid), average of the Tanimoto similarity between all pairs of molecules in the sampled dataset (IntDiv), the same as before but with the squared Tanimoto similarity (IntDiv^2), ratio of sampled molecules that passed a set of filters (PAINS, etc.) (Filters). Test and TestSF are two reference datasets, one containing molecules and the other scaffolds not present in the training set.

Why does the DNC architecture bring no substantial improvements?

The main difference between the DNC and the RNN is the size of the memory and the self-attention mechanism implemented within. In our research we have used as the DNC controller the same RNN architecture, implying that the DNC is an upgrade of the RNN model that has many more parameters to train (thus the increased training times). In our opinion, the reason why the DNC does not perform substantially better is because of the nature of the data. Given that SMILES strings are not excessively long, and the vocabulary size is generally quite small (less than 50 tokens), we think that the hidden states of LSTM cells are enough to store information through the entire generation process. For the same reason, self-attention does not play a crucial role in generating SMILES because sequences are generally short.

Conclusions

This study shows that adding a DNC on top of a regular RNN model improves its learning capabilities. This is especially noted on datasets that hold extremely complex and convoluted structures, such as GDB-13, but less in drug-like molecule datasets such as that used in MOSES. This could be due to the simplicity of the molecules in the MOSES dataset or to MOSES not being able to capture the differences between the two models. In any case, the improvement of adding DNC into LSTM model is not particularly large, probably because that the regular LSTM model alone can already learn the SMILES grammar very well. We think that the DNC architecture is more powerful than the RNN but has many more hyperparameters to optimize and with the limitations of current hardware an exhaustive exploration could not be performed. Moreover, SMILES strings are short (i.e., rarely have more than 100 tokens), thus effectively not benefiting from the main strength of the DNC. For these reasons, we would not

currently recommend using the DNC architecture as a substitute of the RNN. Finally, results indicate that benchmarks for molecular generation might need to include computational time and general resource usage as a metric for model performance.

Declarations

Funding

Josep Arús-Pous is supported financially by the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement no. 676434, "Big Data in Chemistry" ("BIGCHEM" http://bigchem.eu).

Author contributions

Simon Johansson and Oleksii Prykhodko planned and performed the research. Oleksii Prykhodko implemented the DNC. Simon Johansson provided the mathematical analysis and conducted the benchmarking. Josep Arús-Pous contributed with data processing and software development of the baselines, as well as designing the random SMILES implementation. Ola Engkvist and Hongming Chen co-supervised the project. Simon Johansson, Josep Arús-Pous and Hongming Chen wrote the manuscript.

Acknowledgements

We would like to thank Michael Withnall and Amol Thakkar for their technical and scientific insight. We want to thank Panagiotis-Christos Kotsias for his help in implementing the MOSES benchmark and Owen Liu for his assistance in the mathematical analysis. Finally, we want to thank Graham Kemp for his support.

References

1. Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet Classification with Deep Convolutional Neural Networks. In: Pereira F, Burges CJC, Bottou L, Weinberger KQ (eds) Adv. Neural Inf. Process. Syst. 25. Curran Associates, Inc., pp 1097–1105
2. Weiss G, Goldberg Y, Yahav E (2018) On the Practical Computational Power of Finite Precision RNNs for Language Recognition.
3. Silver D, Schrittwieser J, Simonyan K, et al (2017) Mastering the game of Go without human knowledge. Nature 550:354
4. Zhu J, Park T, Isola P, Efros AA (2017) Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In: 2017 IEEE Int. Conf. Comput. Vis. pp 2242–2251
5. Sutskever I, Vinyals O, Le Q V (2014) Sequence to Sequence Learning with Neural Networks. CoRR abs/1409.3:
6. Segler MHS, Kogej T, Tyrchan C, Waller MP (2018) Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks. ACS Cent Sci 4:120–131
7. Olivecrona M, Blaschke T, Engkvist O, Chen H (2017) Molecular de-novo design through deep reinforcement learning. J Cheminform 9:48
8. Hochreiter S, Schmidhuber J (1997) Long Short-Term Memory. Neural Comput 9:1735–1780
9. Gómez-Bombarelli R, Wei JN, Duvenaud D, Hernández-Lobato JM, Sánchez-Lengeling B, Sheberla D, Aguilera-Iparraguirre J, Hirzel TD, Adams RP, Aspuru-Guzik A (2018) Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. ACS Cent Sci 4:268–276
10. Blaschke T, Olivecrona M, Engkvist O, Bajorath J, Chen H (2018) Application of Generative Autoencoder in De Novo Molecular Design. Mol Inform. doi: 10.1002/minf.201700123
11. Sanchez-Lengeling B, Outeiral C, Guimaraes GL, Aspuru-Guzik A (2017) Optimizing

distributions over molecular space. An Objective-Reinforced Generative Adversarial Network for Inverse-design Chemistry (ORGANIC). ChemRxiv 1–18

12. Prykhodko O, Johansson S, Kotsias P-C, Bjerrum EJ, Engkvist O, Chen H (2019) A De Novo Molecular Generation Method Using Latent Vector Based Generative Adversarial Network. doi: 10.26434/chemrxiv.8299544.v1

13. Li Y, Vinyals O, Dyer C, Pascanu R, Battaglia P (2018) Learning Deep Generative Models of Graphs. Iclr 1–16

14. Li Y, Zhang L, Liu Z (2018) Multi-objective de novo drug design with conditional graph generative model. J Cheminform 10:1–24

15. Chen H, Engkvist O, Wang Y, Olivecrona M, Blaschke T (2018) The rise of deep learning in drug discovery. Drug Discov Today 23:1241–1250

16. Graves A, Wayne G, Danihelka I (2014) Neural Turing Machines.

17. Graves A, Wayne G, Reynolds M, et al (2016) Hybrid computing using a neural network with dynamic external memory. Nature 538:471–476

18. Putin E, Asadulaev A, Ivanenkov Y, Aladinskiy V, Sanchez-Lengeling B, Aspuru-Guzik A, Zhavoronkov A (2018) Reinforced Adversarial Neural Computer for de Novo Molecular Design. J Chem Inf Model 58:1194–1204

19. Blum LC, Reymond JL (2009) 970 Million druglike small molecules for virtual screening in the chemical universe database GDB-13. J Am Chem Soc 131:8732–8733

20. Arús-Pous J, Blaschke T, Ulander S, Reymond JL, Chen H, Engkvist O (2019) Exploring the GDB-13 chemical space using deep generative models. J Cheminform 11:20

21. Arús-Pous J, Johansson S, Prykhodko O, Bjerrum EJ, Tyrchan C, Reymond J-L, Chen H, Engkvist O (2019) Randomized SMILES Strings Improve the Quality of Molecular Generative Models. doi: 10.26434/chemrxiv.8639942.v1

22. Polykovskiy D, Zhebrak A, Sanchez-Lengeling B, et al (2018) Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models.

23. Deepmind Differentiable Neural Computer GitHub repository.

https://github.com/deepmind/dnc.

24. Landrum G (2014) RDKit: Open-source cheminformatics. http://www.rdkit.org/

25. Arús-Pous J, Blaschke T, Ulander S, Reymond J-L, Chen H, Engkvist O (2019) Exploring the GDB-13 chemical space using deep generative models. J Cheminform 11:20

26. Irwin JJ, Sterling T, Mysinger MM, Bolstad ES, Coleman RG (2012) ZINC: A free tool to discover chemistry for biology. J Chem Inf Model 52:1757–1768

27. Kingma DP, Ba J (2015) Adam: A Method for Stochastic Optimization. In: Int. Conf. Learn. Represent. San Diego, p 13

28. Pytorch implementation of Differentiable Neural Computer. https://github.com/ixaxaar/pytorch-dnc.

29. Paszke A, Chanan G, Lin Z, Gross S, Yang E, Antiga L, Devito Z (2017) Automatic differentiation in PyTorch. Adv Neural Inf Process Syst 30 1–4

30. Preuer K, Renz P, Unterthiner T, Hochreiter S, Klambauer G (2018) Fréchet ChemNet Distance: A Metric for Generative Models for Molecules in Drug Discovery. J Chem Inf Model 58:1736–1741

31. Jin W, Barzilay R, Jaakkola T (2018) Junction Tree Variational Autoencoder for Molecular Graph Generation.