THERE AND BACK AGAIN: A RETROSYNTHESIS TALE

A PREPRINT

Pavel Karpov[*]
Institute of Structural Biology
Helmholtz Zentrum München
Germany, Munich
carpovpv@gmail.com

Igor V. Tetko[†]
Institute of Structural Biology
Helmholtz Zentrum München
Germany, Munich
itetko@vcclab.org

Guillaume Godin
Firmenich International SA,
Research&Development Division,
Switzerland, Geneva
guillaume.godin@firmenich.com

April 1, 2019

ABSTRACT

We describe a Transformer model for a retrosynthetic reaction prediction task. The model is trained end-to-end on 45033 experimental reaction examples extracted from USA patents. Our final model can successfully predict the reactants set for 44.2% of cases on the external test set. During the training procedure, we applied different learning rate schedules and knowledge distillation. These techniques can prevent overfitting and thus can be a reason to get rid of internal validation dataset that is advantageous for deep models with millions of parameters. We investigate the influence of the temperature while decoding the model output probabilities and find that at T=1.3 the final model behaves like a consensus of models and improve accuracy up to 6%. We also discuss the synergy of applying together the retrosynthetic model to predict reagents and a direct reaction model trained on the same data which makes a prognosis for the product base on reactants.

***K*eywords** Reaction outcomes prediction · Computer retrosynthesis · Character-based models · Transformer

1 Introduction

New chemical compounds drive technological advances in material, agricultural, environmental, and medical sciences, thus, embracing all fields of scientific activities which have been bringing social and economic benefits throughout human history. Design of chemicals with predefined properties is an arena of QSAR/QSPR (Quantitative Structure Activity/Property Relationships) approaches aimed at finding correlations between molecular structures and their desired outcomes and then applying these models to optimise activity/property of compounds.

The advent of deep learning[1] gave a new impulse for virtual modeling and also opened a venue for a promising set of generative methods based on Recurrent Neural Networks[2], Variational Autoencoders[3], and Generative Adversarial Networks trained with reinforcement learning[4, 5]. These techniques are changing the course of QSAR studies from the observation to the invention: from a virtual screening of available compounds to direct synthesis of new candidates. Generative models can produce big sets of promising molecules and impaired with SMILES-based QSAR methods[6] provide a strong foundation for creating highly optimized focussed libraries, but estimation of synthetic availability of these compounds is an open question though several scores based on fragmentation[7] and machine learning[8] approaches have been developed. To synthesize a molecule, one should have a plan of a multi-step synthesis and also a set of available reactants. Finding an optimal combination of reactants, reactions, and conditions to obtain the compound with good yield, sufficient quality, and quantity is not a trivial task even for experts in organic chemistry. Recent advances in the computer-aided synthesis planning are reviewed in [9, 10].

The retrosynthetic analysis worked out by E. J. Corey [11] tries to account for all factors while deriving the synthetic route. It iteratively decomposes the molecule on simpler blocks till all of them become available either by purchase or

[*]Helmholtz Zentrum München – Research Center for Environmental Health (GmbH), Institute of Structural Biology, Ingolstädter Landstraße 1, D-85764 Neuherberg, Germany, https://www.helmholtz-muenchen.de/stb/index.html

[†]BigChem GmbH, https://ochem.eu

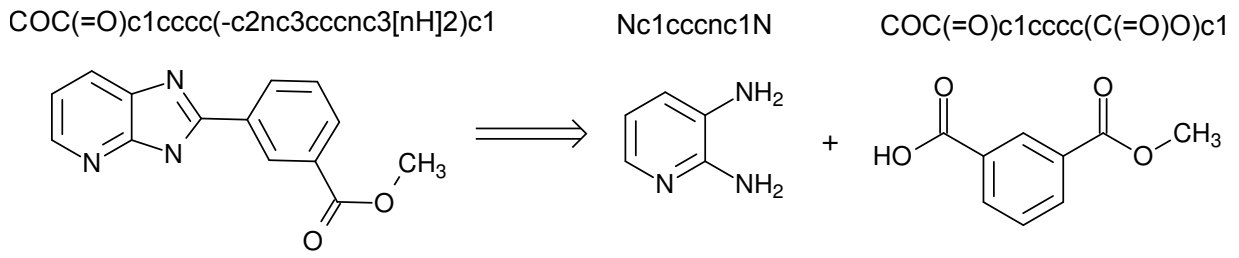

Figure 1: An example of a retrosynthetic reaction: on the left side of the arrow the target molecule is depicted, and on the right side the one possible set of reactants that can lead to the target is shown in common chemistry-like scheme and using SMILES notation. Here two successive amidation reactions result in cyclisation and aromatization.

by synthesis described in the literature. At each step, fig. 2, all possible disconnections (rules) with known reactions simplify the target molecule bringing to the scene less complex compounds. Some of them may be already available, while the others undergo the next step of retrosynthesis decomposition. Due to the recursive nature of the procedure, it can deal with thousands of putative compounds so computational retrosynthetic approaches can greatly help chemists in finding the best routes. Managing of the database of such rules is complicated and more critical the models based on it are not ready to accommodate new reactions and will always be outdated. Almost more than 60 years of developing rule-based systems ended with no remarkable success in synthesis planning programs. Another approach to tackle the problem is to use so-called template-free methods inspired by the success of machine-translation. They don't require the database of templates and rules due to an inherent possibility to derive this information during training directly from a database of organic reactions with clearly designated roles of reactants, products, reagents, and conditions.

The analogy between machine translation and retrosynthesis is evident: each target molecule has its predecessors from which it can be synthesized as every meaningful sentence one can translate from source language to target one. If all parts of a reaction are written in SMILES notation, then our source and target sentence are composed of valid SMILES tokens as words. The main goal of the work is to build a model which could for a given target molecule for example[3] COC(=O)c1cccc(-c2nc3cccnc3[nH]2)c1 in fig. 2 correctly predict the set of reactants. Namely, it should predict Nc1cccnc1N.COC(=O)c1cccc(C(=O)O)c1 in this case.

Neural sequence-to-sequence (seq2seq) approach has been recently applied for a direct reaction prediction task [12, 13] with outstanding statistical parameters of final models – 90.4% of accuracy on test set. Seq2seq modeling has been also tested on retrosynthesis task[14], but due to the complex nature of retrosynthesis itself and difficulty in estimating the correct predictions of reactants [4], accuracy on the test set was moderate 37.4% but still comparable to rule-based systems 35.4%. We questioned about the possibility of improvement models for one-step retrosynthesis utilizing modern neural network architectures and training techniques. Applying the Transformer Model[15], together with cyclical learning rate schedule[16], and knowledge distillation[17] resulted in a single model with accuracy 44.2%, that is more than 6% higher compare to the baseline model[14].

Our main contributions are:

- We show that for this particular task there is no need to use a vliadation dataset for early-stopping or other parameters optimization. We trained all the parameters directly from the training dataset.
- Applying weights averaging, snapshot learning and knowledge distillation helped us to train the most precise model for one-step retrosynthesis prediction.
- We show that different chemistry-relevant SMILES tokenization approaches cannot improve the accuracy of the models compared to the simplest letter based schema.
- Increasing the temperature while performing a beam-search procedure improves the accuracy up to 4%. Almost the same effect as consensus does.
- Using both retrosynthetic model for reactant prediction and a direct model trained on the same data for reaction outcome prognosis allows improving precision of the retrosynthesis model.

[3] This reaction is in the test set and it was correctly predicted by our model. Probability landscapes in the following figures are drawn from this example.

[4] A target molecule usually can be synthesized with different reactions starting from different sets of reactants. The predictions of the model may be correct from organic chemist point of view but differ from the reactant set in ground truth. This may lead to underestimation of effectiveness of models.

2 Approach

2.1 Dataset

In this study we used the same dataset of reactions as in [14]. This dataset was filtered from the USPTO database[18] originally derived from the USA patents and contains 50 000 reactions classified into 10 reaction types[19]. The authors[14] further preprocessed the database by splitting multiple products reactions into multiple single products reactions. The resulting dataset contains 40 029, 5 004, and 5 004 reactions for training, validation, and testing respectively. Information about the reaction type was discarded as we aimed at building a general model using SMILES of products and reactants only.

2.2 Model input

The seq2seq models were developed to support machine translation where the input is a sentence in one language, and the output is a sentence with approximately the same meaning but in another language. String nature of data implies some tokenization procedures similar to word2vec to be used for preprocessing the input. Most of works in cheminformatics dealing with SMILES tokenize the input with a regexp equal or similar to [12].

```
token_regex = "(\[[^\]]+]|Br?|Cl?|N|O|S|P|F|I|b|c|n|o|s|p|\(|\)|
\.|=|#|-|\+|\\\\|\/|:|~|@|\?|>|\*|\$|\%[0-9]{2}|[0-9])".
```

Though such tokenization is more similar to way chemists think, it also has some drawbacks that confuse network by putting forward low represented molecular parts. For example, after applying this regexp to the database one can see some not frequent moieties such as [C@@], [C@@H], [S@@],[C@],[C@H],[N@@+],[se],[C-],[Cl+3]. The thing in brackets according to SMILES specification can be quite a complex gathering not only the element's name itself, but also its isotopic value, stereochemistry configuration, the formal charge, and the number of hydrogens[5]. Strictly speaking, to do tokenization right one should also parse the content of brackets just increasing the number of possible words in the vocabulary what eventually leads to the simplest tokenization only with letters. We tried different schemes of tokenization in this work but did not see any improvements in using them over character-based method.

Our final vocabulary has length of 66 symbols[6]:

```
chars = " ~#%()+-.0123456789=@ABCDEFGHIKLMNOPRSTVXYZ[\\]abcdefgilmnoprstuy$"
```

To convert a token to a dense vector we used a trainable embedding [7] of size 64. It is well known that training neural networks in batches is more stable, faster, and leads to more accurate models. To facilitate batch training we also used masks of input strings of shape (batch_size, max_length) with elements equal to 1 for those positions where are valid SMILES symbols and 0 everywhere else.

2.3 Transformer model

We used a promising Transformer [15] model for this study which is a new generation of encoder-decoder neural networks family. The architecture is suited for exploration of the internal representation of data by deriving questions (Q) the data could be asked for, keys for its indexed knowledge (K), and answers written as values (V) corresponding to queries and keys. Technically these three entities are simply matrixes learned during the network training. Multiplying them with the input (X) gives keys (k), questions (q), and values (v) relevant to a current batch. Equipped with these calculated parameters of the input the self-attention layers transforms it pointing out to some encoding (decoding) parts based on the attention vector.

The Transformer has wholly got rid of any recurrences or convolutional operations. To tackle distances between elements of a string a positional encoding matrix was proposed with elements equal to the values of trigonometric functions depending on the position in a string and also the position in the embedding direction. Summed with learned embeddings positional encodings do their job linking far located parts of the input together. The output of self-attention

[5] We do not want to exclude stereochemistry information from our model as well as charges and explicit hydrogens that will lead to reducing of the dataset. Moreover, work in generative models showed excellent abilities of models to close cycles, for example, c**1**cc(COC)cccc**1**. If the model can capture such a long distance relation why should it be cracked on much simplier substrings enclosed by brackets?

[6] This vocabulary derived from the complete USPTO set and is a little bit wider than needed for this study. But for future extending of the models it is better to fix the input shape to the biggest possible value.

[7] The encoder and the decoder use their own embeddings in this study.

layers is then mixed with original data, layer-wise normalized, and passed position-wise through a couple of ordinary dense layers to go further either in next level of self-attention layers or to a decoder as an information-rich vector representing the input. The decoder part of Transformer resembles the encoder but has an additional self-attention layer which corresponds to encoder's output.

Transformer model shows the state-of-the-art results in machine translation and reaction prediction outcomes [13]. The latter work showed that training the Transformer on large and noisy datasets results in a model that can outperform not only other machine models but also well qualified and experienced organic chemists.

2.4 Model inference

The model tries to estimate the probability of the next symbol over the model's vocabulary given all previous symbols in the string. Technically, the Transformer model first calculates logits, z_i, and then transforms them to probabilities.

$$z_i = Transformer(\{x_1, x_2, x_3, ..., x_L\}, \{y_1, y_2, y_3, ..., y_{i-1}\}) \tag{1}$$

Here x_i is the input of the models at i position; L – the length of the input string; y_i is the decoded output of the model up to position $(i-1)$; and z_i – logits that are to be converted to probabilities:

$$q_i = \frac{exp(z_i/T)}{\sum_{j=0}^{V} exp(z_j/T)} \tag{2}$$

where V is the size of the vocabulary (66 in this work) and T stands for the temperature[8] usually assigned to 1.0 in standard softmax layers. With higher T the landscape of the probability distribution becomes more smooth. During the training the model adapts its weights to better predict q_i, so $y_i = q_i$.

During the inference however we have several possibilities how to convert q_i into y_i, namely greedy and beam search. The first one picks up a symbol with maximum probability whereas the second one at each step holds $top-K$ (K = beam's size) suggestions of the model and summarises the overall likelihood for each of K final decodings. The beam search allows better inference and the probability landscape exploration compared to the greedy search because at a particular step of decoding it may choose a symbol with less than maximum probability, but the total likelihood of the result can be higher due to more significant probabilities on the next steps.

2.5 Training heuristics

Training a Transformer model is a challenge, and several heuristics have been proposed[16], some of them were used in this study:

Using as bigger batch size as possible. However, due to our hardware limitations we could not set the batch size more then 64[9];

Iincreasing the learning rate at the beginning of training up to warmup steps[10]. The authors of the original Transformer paper [15] used 4 000 steps for warming. The Transformer model for reaction prediction task from [13] used 8 000 steps. We have also tried different values for warmup and eventually found that 16 000 works well with our model.

Applying cyclic learning rate schedules. This tips can generally improve any model[20] through better loss landscape exploration with bigger learning rates after the optimiser felt down to some local minima. For this study we used the following scheme for learning rate calculation depending on the step:

$$u(step) = \begin{cases} warmup + (step \bmod cycle), & \text{if } step \geq cycle \\ step, & \text{otherwise} \end{cases}$$

where $cycle$ stands for the number of steps while the learning rate is decreasing before raising to the maximum again.

$$\lambda(step) = factor * \frac{min(1.0, u(step)/warmup)}{max(u(step), warmup)} \tag{3}$$

where $factor$ is just a constant. Big values of $factor$ introduce numerical instability during training, so after several trials we set $factor = 20.0$. The curve for learning rate in this study is shown in fig. 2, plot (4, f).

[8] Similar to formula of Boltzmann (Gibbs) distribution used in statistical mechanics.

[9] Our first implementation of the model required a lot of memory to deal with masks of reactants and products. Though later we improved the code we still remained this size for consistency of the results.

[10] In our implementation 1 step is equivalent to 1 batch. The number of reactions for training is 40 029 + 5 004, so one epoch is equal to 704 batches.

Averaging weights during last steps (usually 10- 20) of training or at minima of learning rates in case of snapshop learning [21].

2.6 Knowledge distillation

After training a (teacher) model it is possible to get predictions for the training set and then train another (student) model on these predictions. This second model usually outperforms the first one. This approach was further developed in[17] where the algorithm for knowledge distillation was proposed. The key idea is to calculate probabilities for training dataset at higher temperature T, eq. 2, and then use this values as targets along with the correct labels. The final loss function:

$$Loss = -\sum ylog(q) - T^2 * \sum q_{teacher} log(q) \tag{4}$$

contains terms for hard and soft labels and, thus, the model will learn to distinguish subtle differences in final probability distribution. We applied this technique for further training our models and found that this simple approach can improve the accuracy up to 2%.

3 Results

For this study, we implemented The Transformer model in Tensorflow library to support its integration in our in-house programs set (`https://github.com/bigchem/retrosynthesis`). All values reported are averages for three repetitive runs. Preliminary modeling showed that the architecture with 3 layers and 8 attention works well for the datasets, though we tried combinations of 2, 6, 8, 10 layers with 6, 8, 10, 12 heads. So all calculations were performed with these values fixed. The number of learnable parameters of the model is 1 882 176, embedding layer common for product and reactants has size 64.

Following the standard machine learning protocol, we trained our first models (T1) using three datasets for training, validation, and external testing (8:1:1) as was done in [14]. Learning curves for T1 are depicted in fig. 2, (d) and (e) for training and validation loss, respectively, (g) shows the original learning rate schedule developed by the authors of the Transformer but with 16 000 warmup steps. On reaching cross-entropy loss about 0.1 on the validation dataset, it stagnates without noticeable fluctuations as training loss steadily decreases. After warming up phase the learning rate begins fading and eventually after 1 000 epochs its value reaches $2.8 * 10^{-5}$ inevitable causing to stop training because of too small updates.

During the decoding procedure, we explored the influence of the temperature parameter on the final quality of prediction and surprisingly found that inferring at higher temperatures gives better result then at T=1. This observation similarly repeated for all our models. Fig. **??** shows the influence of this parameter on the reactants prediction of the part of the training set. Clearly, at T=1.3 the model reaches the maximum of chemically-based accuracy. This fact one can explain that at higher temperatures the landscape of output probabilities of the model is softer letting the beam-search procedure to find more suitable ways during decoding. Of course, the temperature influences only relative distances between peaks, so it does not affect the greedy search method.

Table 1: Accuracy (%) of the models on test set when all reactants were correctly predicted.

Model	Greedy	Top-1	Top-3	Top-5	Description
Seq2Seq		37.4	52.4	57.0	Literature result from [14] based on Seq2Seq architecture.
T1					Transformer Model trained with validation control set (early stopping, 200 epochs).
$T1_1$					The same as T1, but without early stopping (1000 epochs).
T2	39.3	41.1	60.5	66.6	Transformer Model trained on both training and validation sets for 1000 epochs.
$T2_1$		41.8	61.3	67.2	T = 1.3 during the decoding stage of the T2 model.
T3					Transformer Model trained with cyclic learning rate schedule for 6000 epochs.
$T3_1$					T = 1.3 during the decoding stage of T3 model
T4					Transformer Model trained with distillation approach [17] for 1000 epochs.
$T4_1$					The same as T4 but decoding at T=1.3

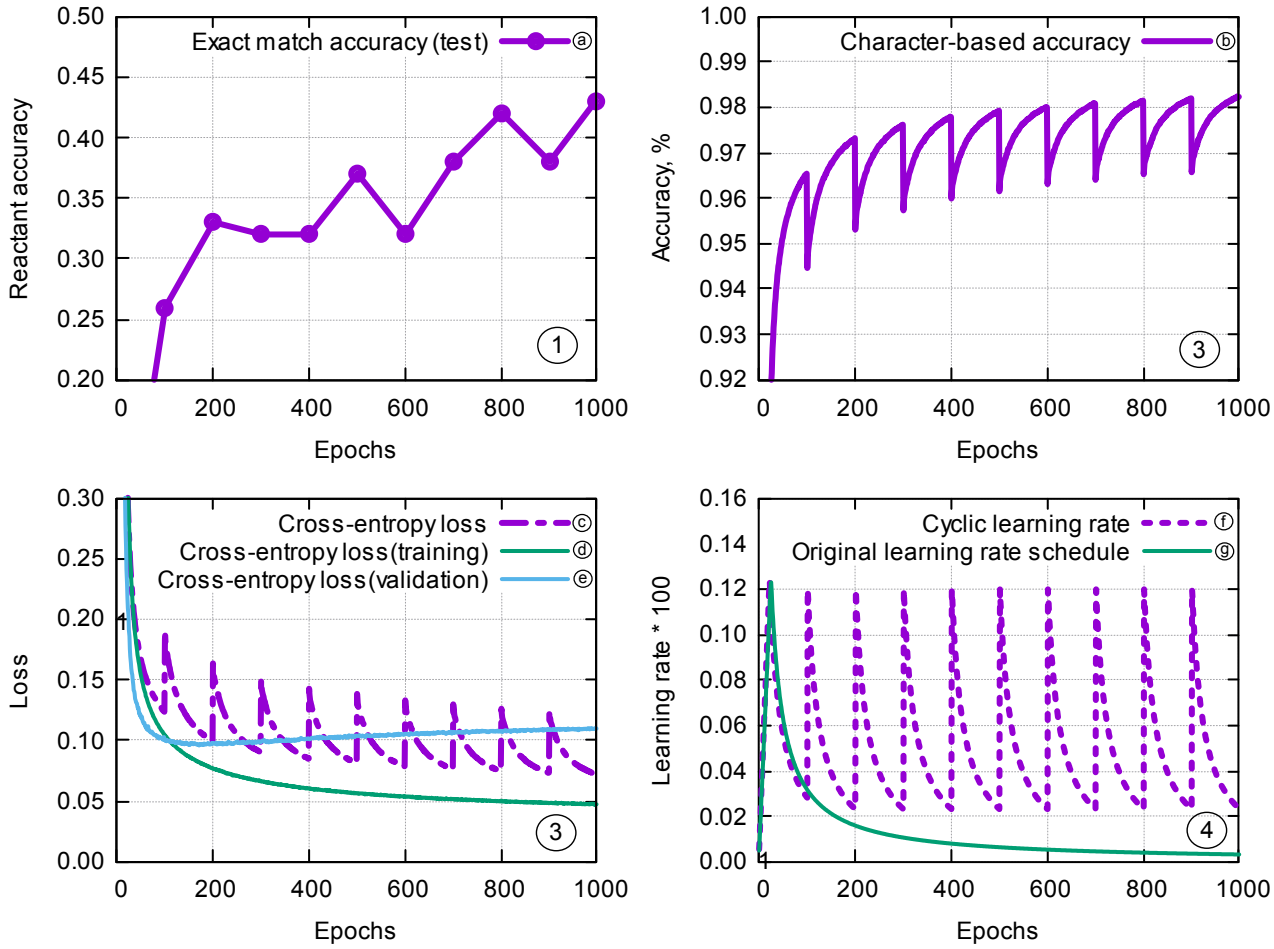

Figure 2: Summary of learning curves for the Transformer model: 1) Accuracy of correctly predicted reactants in test set (a); 2) character based accuracy (b); 3) cross-entropy loss with cycling learning rate (f) for the entire dataset (c) and losses for training (d) and validation (e) sets with monotonically decreasing learning rate (g); 4) learning rate schedules as a function of epoch.

If we apply the early stopping technique, we should select the model around 200 epoch. Effectiveness of such a model marked $T1_1$ in table 1 resulted in TOP-1 40.16% on the test set. If we choose the last one model obtained at 1 000 epoch, then the model $T1_2$ gives us slightly better values. In this case, we did not see any need of the validation dataset and keeping in mind that our model has almost 2 millions of parameters we decided to combine training and validation sets and train our next models on both data, e.g., without validation. The model $T2$ was trained on all data and with the same learning rate schedule as $T1$. The results obtained when applying T2 to the test set are better than for T1 set.

Then we trained our model with cyclic learning rate schedule, eq. 3, fig. **??** (f) for better exploration of loss landscape. During training, we also saved the character-based accuracy of the model, fig. (b). This snapshot training regime [21] produces a set of different weights at each minimum of learning rate. Averaging them is to some extent equivalent to a consensus of models but within one model [20]. We tried different averaging regimes $T3_{2-3}$ but did not see any remarkable improvements.

Our T3 model outperforms [14] by 3.8% with beam search and more critical by 3.0% with the greedy search. The latter one is much faster and consequently more suitable for virtual screening campaigns. To improve the quality of this model, we applied the knowledge distillation approach. First, the soft labels were calculated using T3 model, and, second, we traing T4 model from scratch but with two terms in the objective function: one for hard and one for soft labels, eq. 4.

Our T3 model outperforms [14] by 3.8% with beam search and more critical by 3.0% with the greedy search. The latter one is much faster and consequently more suitable for virtual screening campaigns. To improve the quality of

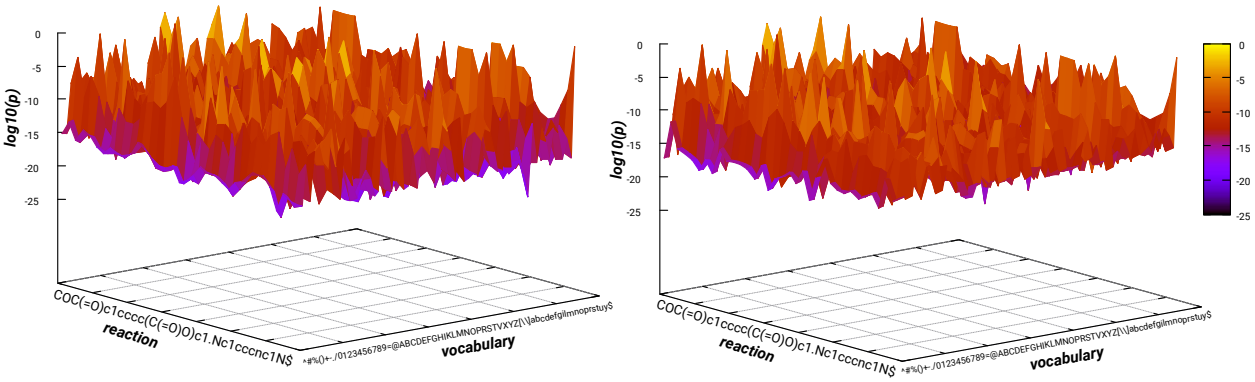

Figure 3: Landscapes of probabilities for predicting example reaction, fig. 2, with T3 model(left) and T4 (distilled) model (right).

this model, we applied the knowledge distillation approach. First, the soft labels were calculated using T3 model, and, second, we traing T4 model from scratch but with two terms in the objective function: one for hard and one for soft labels, eq. 4.

4 Discussion

Some discussion...

All source code and also models built are available online via github

`https://github.com/bigchem/retrosynthesis`

Acknowledgments

This study has been partially supported by ERA-CVD (`httpd://era-cvd.eu`) "Cardio-Oncology" project, BMBF 01KL1710. The authors thank NVIDIA Corporation for donating Quadro P6000 and Titan Xp graphics cards for this research.

References

[1] Chen, H., Engkvist, O., Wang, Y., Olivecrona, M. & Blaschke, T. The rise of deep learning in drug discovery. *Drug Discovery Today* **23**, 1241 – 1250 (2018).

[2] Ertl, P., Lewis, R., Martin, E. & Polyakov, V. In silico generation of novel, drug-like chemical matter using the lstm neural network. *arXiv* (2017). `1712.07449v2`.

[3] Gómez-Bombarelli, R. *et al.* Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science* **4**, 268–276 (2018).

[4] Guimaraes, G. L., Sanchez-Lengeling, B., Outeiral, C., Farias, P. L. C. & Aspuru-Guzik, A. Objective-reinforced generative adversarial networks (organ) for sequence generation models. *arXiv* (2017). `1705.10843v3`.

[5] Olivecrona, M., Blaschke, T. & hongming Chen, O. E. Molecular de-novo design through deep reinforcement learning. *J Cheminform.* **9**, 1758–2946 (2017).

[6] Kimber, T. B., Engelke, S., Tetko, I. V., Bruno, E. & Godin, G. Synergy effect between convolutional neural networks and the multiplicity of smiles for improvement of molecular prediction. *arXiv* (2018). `1812.04439v1`.

[7] Ertl, P. & Schuffenhauer, A. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *J. Cheminform.* **1**, 8 (2009).

[8] Coley, C. W., Rogers, L., Green, W. H. & Jensen, K. F. Scscore: Synthetic complexity learned from a reaction corpus. *Journal of Chemical Information and Modeling* **58**, 252–261 (2018). URL `https://doi.org/10.1021/acs.jcim.7b00622`. PMID: 29309147.

[9] Coley, C. W., Green, W. H. & Jensen, K. F. Machine learning in computer-aided synthesis planning. *Accounts of Chemical Research* **51**, 1281–1289 (2018). PMID: 29715002.

[10] Engkvist, O. *et al.* Computational prediction of chemical reactions: current status and outlook. *Drug Discovery Today* **23**, 1203 – 1218 (2018).

[11] Corey, E. J. & Cheng, X.-M. *The Logic of Chemical Synthesis* (Wiley-Interscience, 1995).

[12] Schwaller, P., Gaudin, T., Lanyi, D., Bekas, C. & Laino, T. Found in translation: Predicting outcomes of complex organic chemistry reactions using neural sequence-to-sequence models. *ArXiv* (2018). 1711.04810v2.

[13] Schwaller, P. *et al.* Molecular transformer for chemical reaction prediction and uncertainty estimation. *arXiv* (2018). 1811.02633v1.

[14] Liu, B. *et al.* Retrosynthetic reaction prediction using neural sequence-to-sequence models. *ACS Central Science* **3**, 1103–1113 (2017).

[15] Vaswani, A. *et al.* Attention is all you need. *ArXiv* (2017). 1706.03762.

[16] Popel, M. & Bojar, O. Training tips for the transformer model. *arXiv* (2018). 1804.00247v2.

[17] Hinton, G., Vinyals, O. & Dean, J. Distilling the knowledge in a neural network. *arXiv* (2015). 1503.02531v1.

[18] Lowe, D. M. *Extraction of chemical structures and reactions from the literature*. Ph.D. thesis (2012). URL https://www.repository.cam.ac.uk/handle/1810/244727.

[19] Schneider, N., Stiefl, N. & Landrum, G. A. What's what: The (nearly) definitive guide to reaction role assignment. *Journal of Chemical Information and Modeling* **56**, 2336–2346 (2016). PMID: 28024398, https://doi.org/10.1021/acs.jcim.6b00564.

[20] Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D. & Wilson, A. G. Averaging weights leads to wider optima and better generalization 1803.05407v3.

[21] Huang, G. *et al.* Snapshot ensembles: Train 1, get m for free 1704.00109v1.

Appendix

In this section we report all characteristics of repetitive runs of all models built in this study. Average values based on this data are summed in table 1. For every run we report learning curves with loss functions and learning rate, then model relevant information are collected in a table including greedy search decoding accuracies, beam search accuracies for Top-1, Top-3, and Top-5.

4.1 T1 model

4.1.1 Model 1 (run-100)

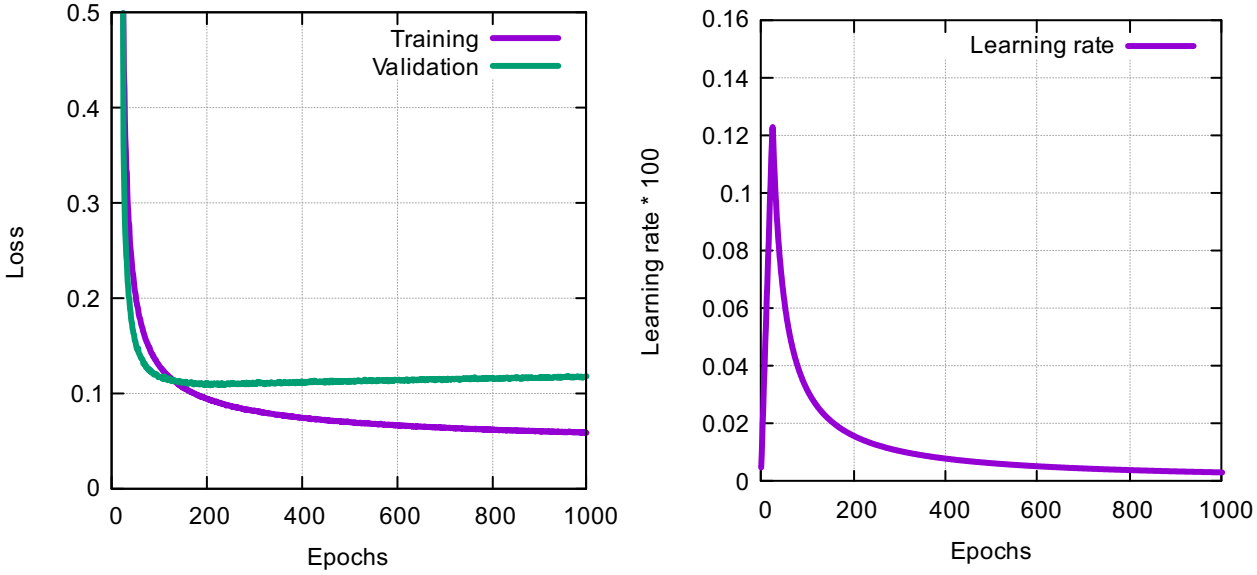

Figure 4: Learning curves and rate schedule for T1 model. Losses for training and validation datasets on the left, usual learning rate schedule on the right.

Table 2: Parameters of T11 run.

Parameter	Value	Top-1	Top-3	Top-5
Early stopping loss (training / validation)	0.09345 / 0.10864			
Last training loss (training / validation)	0.05867 / 0.11819			
Epoch (early stopping)	208			
Greedy search (early stopping, epoch 208)		32.83		
Greedy search (last, epoch 999)		34.75		
Greedy (last 10 average, epoch, 990 - 999)		35.57		
Beam, T = 1.0 (early)		35.57	54.14	59.13
Beam, T = 1.3 (early)		35.93	54.59	59.51
Beam, T = 1.0 (last)		37.15	55.22	60.49
Beam, T = 1.3 (last)		37.67	55.97	61.01
Beam, T = 1.0 (last 10 average)		37.83	56.08	60.63
Beam, T = 1.3 (last 10 average)		**38.15**	**56.71**	**61.31**

`https://github.com/bigchem/retrosynthesis/papers/retrosynthesis/models/t1-1-early.h5`

`https://github.com/bigchem/retrosynthesis/papers/retrosynthesis/models/t1-1-last.h5`

`https://github.com/bigchem/retrosynthesis/papers/retrosynthesis/models/t1-1-avg.h5`

4.1.2 Model 1 (run-101)

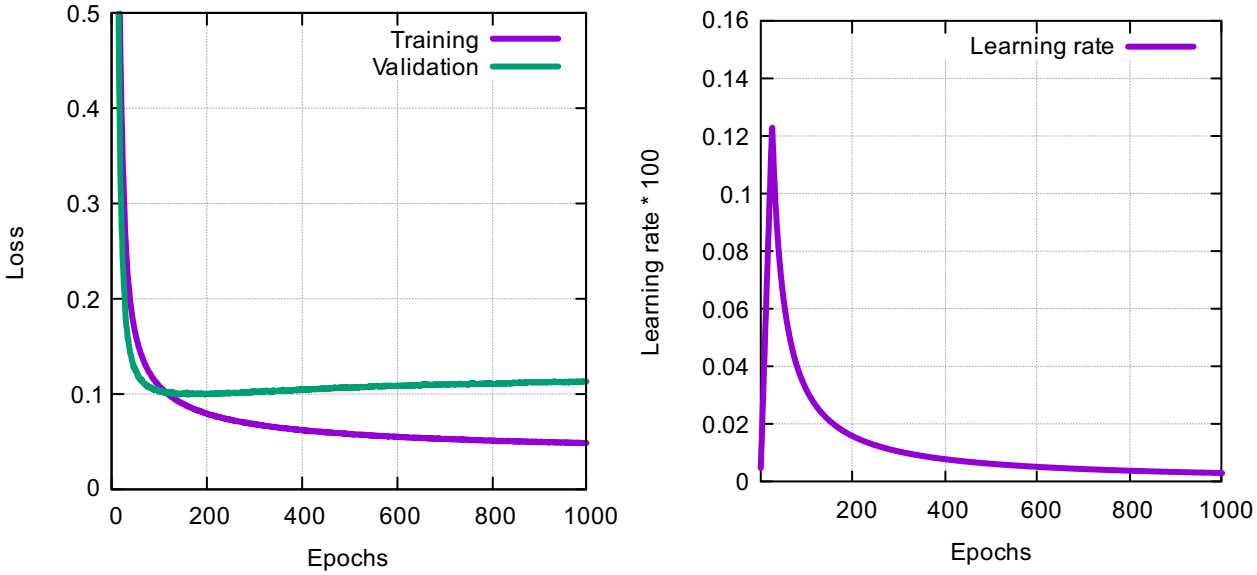

Figure 5: Learning curves and rate schedule for T1 model. Losses for training and validation datasets on the left, usual learning rate schedule on the right.

Table 3: Parameters of T12 run.

Parameter	Value	Top-1	Top-3	Top-5
Early stopping loss (training / validation)	0.08681 / 0.09926			
Last training loss (training / validation)	0.04863 / 0.11316			
Epoch (early stopping)	162			
Greedy search (early stopping, epoch 162)		34.63		
Greedy search (last, epoch 990)		38.07		
Greedy (last 10 average, epoch, 990 - 999)		38.19		
Beam, T = 1.0 (early)		37.45	56.71	62.43
Beam, T = 1.3 (early)		38.31	57.93	63.35
Beam, T = 1.0 (last)		39.83	58.63	64.45
Beam, T = 1.3 (last)		40.67	59.53	64.97
Beam, T = 1.0 (last 10 average)		40.22	59.05	64.91
Beam, T = 1.3 (last 10 average)		**40.85**	**59.85**	**65.45**

`https://github.com/bigchem/retrosynthesis/papers/retrosynthesis/models/t1-2-early.h5`

`https://github.com/bigchem/retrosynthesis/papers/retrosynthesis/models/t1-2-last.h5`

`https://github.com/bigchem/retrosynthesis/papers/retrosynthesis/models/t1-2-avg.h5`

4.1.3 Model 1 (run-102)

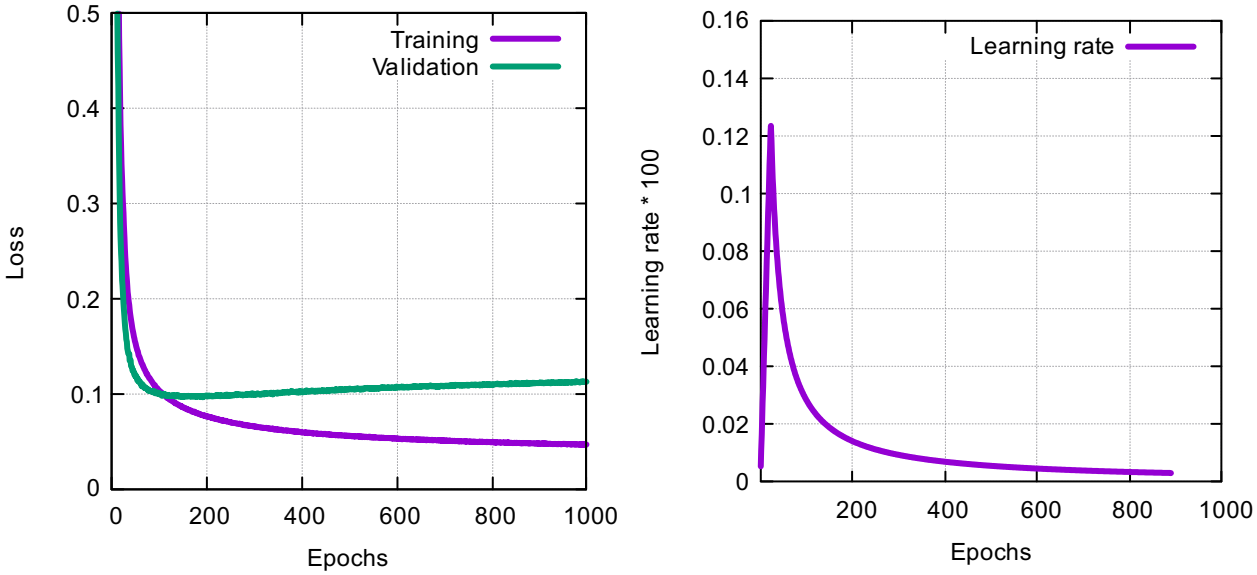

Figure 6: Learning curves and rate schedule for T1 model. Losses for training and validation datasets on the left, usual learning rate schedule on the right.

Table 4: Parameters of T13 run.

Parameter	Value	Top-1	Top-3	Top-5
Early stopping loss (training / validation)	0.07795 / 0.09662			
Last training loss (training / validation)	0.04654 / 0.11285			
Epoch (early stopping)	193			
Greedy search (early stopping, epoch 193)		35.63		
Greedy search (last, epoch 998)		38.13		
Greedy (last 10 average, epoch, 990 - 999)				
Beam, T = 1.0 (early)		38.75	58.55	64.47
Beam, T = 1.3 (early)		39.53	59.25	65.23
Beam, T = 1.0 (last)		40.05	59.35	64.27
Beam, T = 1.3 (last)		40.41	60.49	64.85
Beam, T = 1.0 (last 10 average)				
Beam, T = 1.3 (last 10 average)				

`https://github.com/bigchem/retrosynthesis/papers/retrosynthesis/models/t1-3-early.h5`

`https://github.com/bigchem/retrosynthesis/papers/retrosynthesis/models/t1-3-last.h5`

`https://github.com/bigchem/retrosynthesis/papers/retrosynthesis/models/t1-3-avg.h5`

4.1.4 Model 2 (run-200)

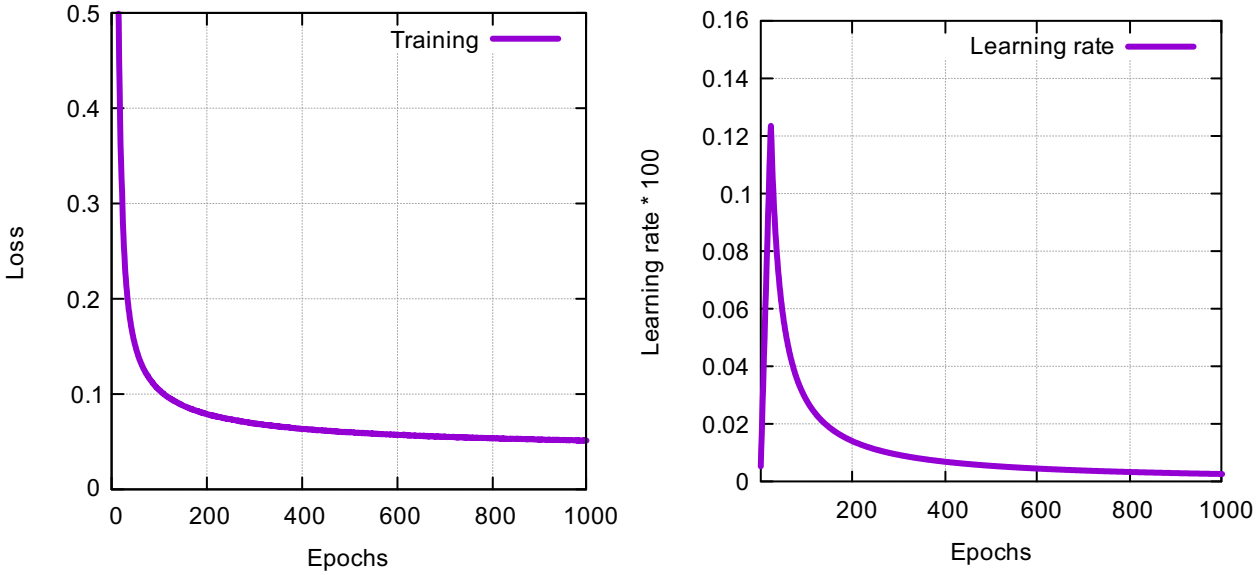

Figure 7: Learning curves and rate schedule for T2 model. Loss for training on the left, usual learning rate schedule on the right.

Table 5: Parameters of T21 run.

Parameter	Value	Top-1	Top-3	Top-5
Last training loss	0.05108			
Greedy search (last, epoch 981)		38.85		
Greedy (last 10 average, epoch, 990 - 999)		39.25		
Beam, T = 1.0 (last)		40.57	60.19	66.77
Beam, T = 1.3 (last)		41.17	60.65	67.28
Beam, T = 1.0 (last 10 average)		40.98	60.29	67.00
Beam, T = 1.3 (last 10 average)		**41.61**	**60.85**	**67.37**

`https://github.com/bigchem/retrosynthesis/papers/retrosynthesis/models/t2-1-last.h5`

`https://github.com/bigchem/retrosynthesis/papers/retrosynthesis/models/t2-1-avg.h5`

4.1.5 Model 2 (run-201)

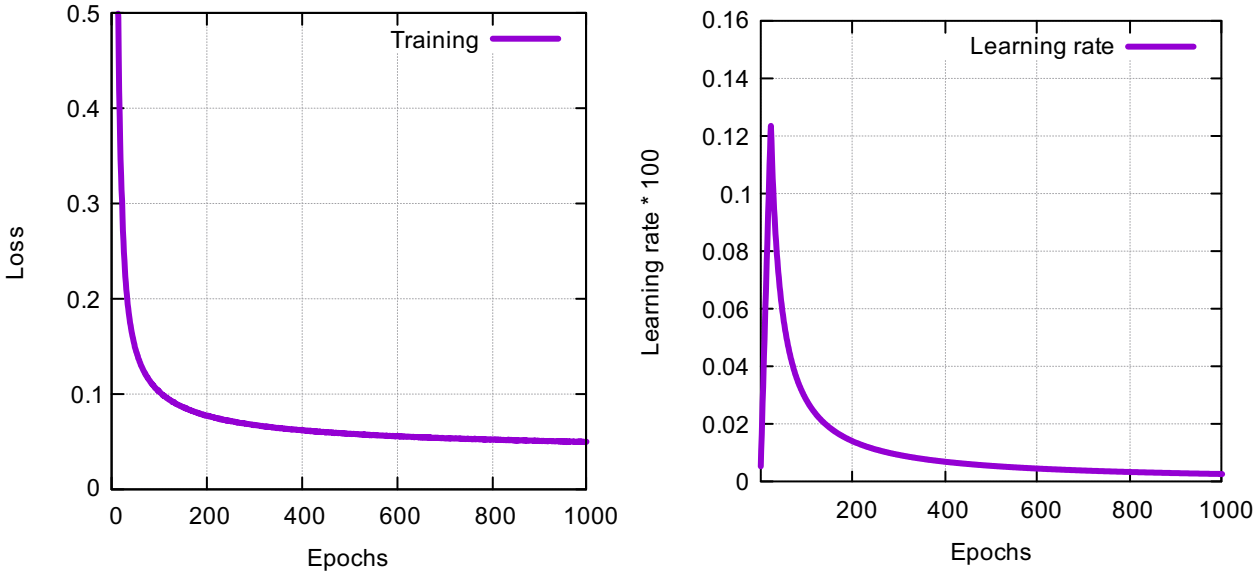

Figure 8: Learning curves and rate schedule for T2 model. Loss for training on the left, usual learning rate schedule on the right.

Table 6: Parameters of T22 run.

Parameter	Value	Top-1	Top-3	Top-5
Last training loss	0.04943			
Greedy search (last, epoch 996)		38.79		
Greedy (last 10 average, epoch, 990 - 999)		39.11		
Beam, T = 1.0 (last)		40.98	60.39	66.38
Beam, T = 1.3 (last)		41.58	61.21	66.94
Beam, T = 1.0 (last 10 average)		41.14	60.59	66.41
Beam, T = 1.3 (last 10 average)		**41.87**	**61.35**	**67.08**

`https://github.com/bigchem/retrosynthesis/papers/retrosynthesis/models/t2-2-last.h5`

`https://github.com/bigchem/retrosynthesis/papers/retrosynthesis/models/t2-2-avg.h5`

4.1.6 Model 2 (run-202)

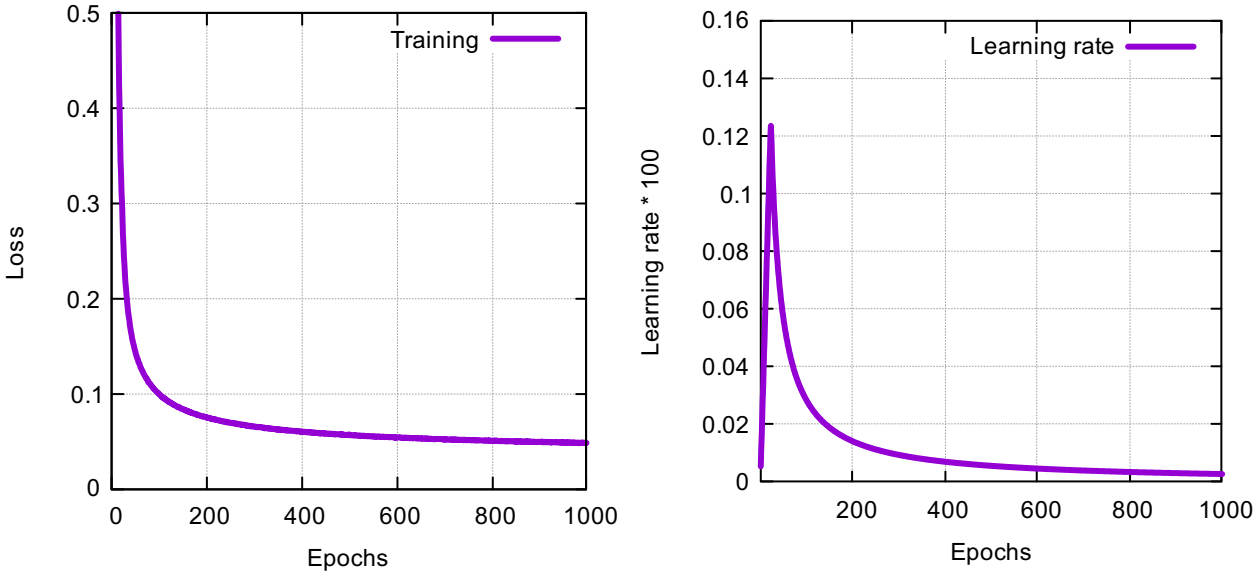

Figure 9: Learning curves and rate schedule for T2 model. Loss for training on the left, usual learning rate schedule on the right.

Table 7: Parameters of T23 run.

Parameter	Value	Top-1	Top-3	Top-5
Last training loss	0.04826			
Greedy search (last, epoch 984)		39.79		
Greedy (last 10 average, epoch, 990 - 999)		39.69		
Beam, T = 1.0 (last)		41.55	60.79	66.57
Beam, T = 1.3 (last)		41.85	61.63	67.32
Beam, T = 1.0 (last 10 average)		41.45	60.75	66.55
Beam, T = 1.3 (last 10 average)		**42.02**	**61.73**	**67.19**

`https://github.com/bigchem/retrosynthesis/papers/retrosynthesis/models/t2-3-last.h5`

`https://github.com/bigchem/retrosynthesis/papers/retrosynthesis/models/t2-3-avg.h5`

4.1.7 Model 3 (run-3)

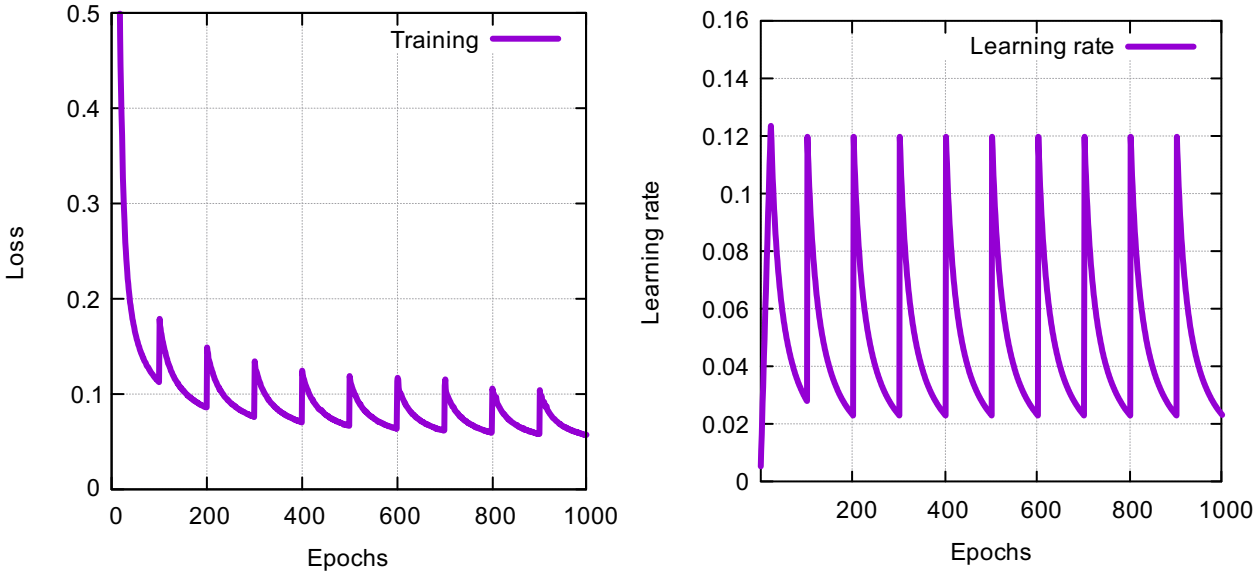

Figure 10: Learning curves and rate schedule for T3 model. Loss for training on the left, usual learning rate schedule on the right.

Table 8: Parameters of T31 run.

Parameter	Value	Top-1	Top-3	Top-5
Last training loss	0.05712			
Greedy search (last, epoch 1000)		39.19		
Greedy (last 10 average, epoch, 991 - 1000)		40.23		
Beam, T = 1.0 (last)		40.97	61.23	67.79
Beam, T = 1.3 (last)		41.22	62.07	67.85
Beam, T = 1.0 (last 10 average)		41.61	62.13	68.01
Beam, T = 1.3 (last 10 average)		42.23	62.87	68.31