# Electronic Supporting Information: Designing catalysts with deep generative models and computational data. A case study for Suzuki cross coupling reactions

Oliver Schilter,*[a,b] Alain Vaucher,[a,b] Philippe Schwaller[b,c] and Teodoro Laino[a,b]

## 1 DFT calculated energies

The workflow used by Meyer et al.[1] to calculate the binding energies was the following: The SMILES of the molecules were translated into Cartesian coordinates via OpenBable[2]. The steepest decent geometry optimization was used for 250 steps with an MMFF95 force field. 200 iteration Weighted Rotor conformation search was performed followed by 250 steps of conjugate gradient geometry optimization. These forced field-optimized conformers were further optimized. For Ni, Pd, Cu, Ag the base set B3LYP/3-21G[3] was used while the Pt and Au catalyst used B3LYP/def2-SVP[4] for the further geometry optimization. Using B3LYP-D3/def2-TZVP[4] the single point energy was calculated. The DFT calculation was performed in Gaussian09[5].

## 2 Random Forest benchmark

A random forest regressor model, implemented from scikit-learn with 200 trees, was trained on Morgan fingerprints of the catalysts with a length of 2048 to predict the binding energy.

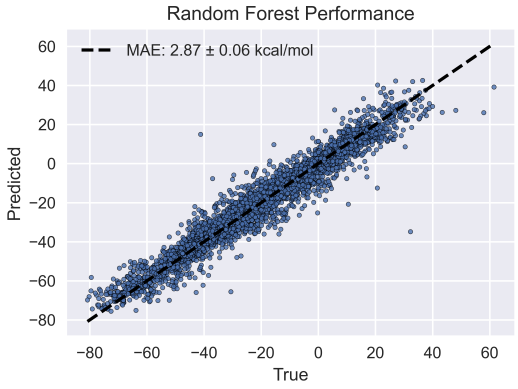


Fig. 1 Random Forest benchmark trained on Morgan Fingerprints length 2024.

## 3 Validity of a generated molecule

A generated molecule was considered as valid if the following list of requirements was fulfilled:

1. The sequence starts with a [Start]-token and ends with a [End]-token followed by [Padding]-tokens (example: [Start][C][C][.][C][.][Pd][End][Padding][Padding][Padding])
2. The molecular structure for the catalyst is containing exactly one transition metal and the two ligands separated by the [.]-token
3. Each of the ligands must be chemically valid (no open rings, no hypervalent C), therefore RdKit[6] must be able to parse the ligand sequence without raising an error.

## 4 Model architecture and training

All models were implemented using PyTorch[7] and trained using Pytorch Ligthing[8]. The molecule data is converted into tokens based on the tokenizer by Schwaller et al.[9]. The tokenized representation is embedded via an embedding layer which is inputted into the first of three layers of recurrent neural network cells (RNN) with a hidden state size of 256. The output of the last RNN layer is then propagated into a $\mu$ and $\sigma$ by two feed-forward layers with an output size of the latent dimension 32. From this output, distribution is then sampled from the latent space. This latent space is the input state of the RNN-based decoder. The decoder consists of 3 layers of RNN cells with a hidden state size of 256 between them. The last output layer is then used to apply a cross-entropy loss function.

Table 1 Model architecture used.

| Layer | Value |
|---|---|
| Embedding layer (sequence length) | 207 |
| Hidden size rnn encoder | 256 |
| n layers encoder | 3 |
| Dropout encoder | 0.1 |
| Feed forward layer after encoder | 1024 |
| Latent size | 32 |
| Hidden size decoder | 256 |
| Dropout decoder | 0.1 |
| n layers decoder | 3 |
| n property predictor layer | 3 |
| Dropout property predictor | 0.1 |
| Activation function property predictor | ReLu |
| Output size property predictor | 1 |

The following model training parameters were used for the SMILES and the SELFIES model. The training was performed on 1 Nvidia A-100 GPU. The model was trained using the ADAM optimizer with a learning rate of 0.0001 and batch size of 200. The KLD loss term was added with linearly increasing weight

[a] IBM Research Europe, Saümerstrasse 4, 8803 Rüschlikon, Switzerland. E-mail: oli@zurich.ibm.com
[b] National Center for Competence in Research-Catalysis (NCCR-Catalysis), Switzerland
[c] Current Address: EPFL Lausanne, Rte Cantonale, 1015 Lausanne, Switzerland.

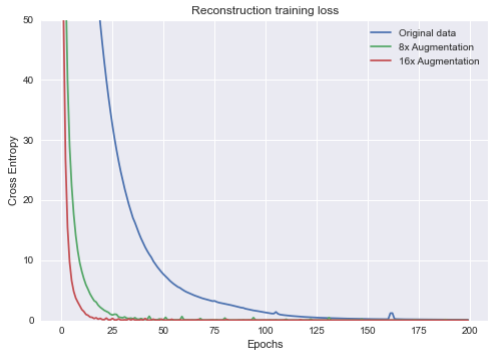


Fig. 2 reconstruction loss SMILES models.

over 100 epochs. The KLD loss, mean absolute error of the predictor, as well as cross entropy loss of the reconstruction loss of the decoder, were added with equal weight. After each epoch, the performance was evaluated on the validation set. After each evaluation, the model was saved if its total validation was lowest recorded during the training.

Table 2 Training parameters.

| Parameter | Value |
|---|---|
| Optimizer | Adam |
| Learning rate | 0.0001 |
| Batch size | 200 |
| Kl $\beta$ annealing gradient | linear |
| Kl $\beta$ annealing end | 0.1 |
| Kl $\beta$ annealing epochs | 100 |

## 5 Influence latent space dimension

To find the optimal latent space dimension a screening of a variety of different size latent spaces was performed. The overall lowest validation loss (reconstruction loss + property predictor loss + KLD loss) was taken as a measurement to determine the best-performing latent space size. As seen in Figure 3 the latent space with dimension 32 has the overall lowest loss.

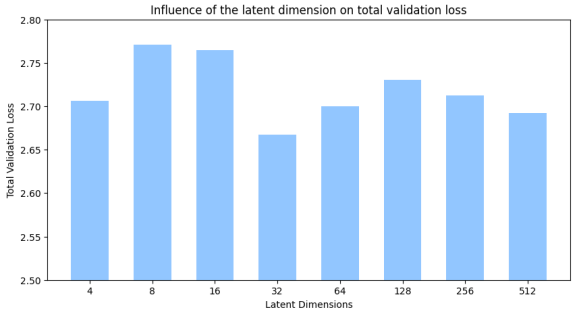


Fig. 3 The validation loss of a variety of latent spaces sizes for the nonaugmented SMILES model.

## 6 Entropy analysis

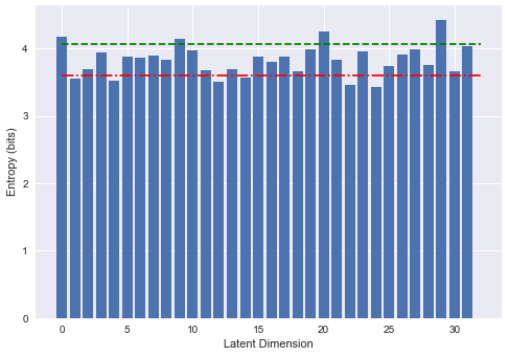


Fig. 4 The entropy values of each dimension (0 augmentation SMILES model) show that none of the dimensions is superfluous and all dimensions store meaningful information. The red and green lines correspond to one standard deviation above and below the mean value of entropy.

## 7 Molecule Generation

The molecule generation was performed with the stochastic gradient descent (SGD) implementation of Pytorch[7]. First all molecules in the dataset were encoded into their latent representation. Then the minimum and maximum value of each dimension over all the encoded molecules were calculated. These min and max values server as boundaries to generated a random point as initial starting point for the optimization. This point was evaluated regarding their predicted energy with the trained property prediction model and the mean absolute difference of the predicted energy and the target value was returned to the optimizer to minimize. At each step, although only for monitoring propose, the points were decoded by the trained model into SMILES or SELFIES. The optimizer then optimizes for 10 iterations the latent representation. The last generated molecule was then analyzed for their validity (see above). The learning rate for the SGD algorithm was chosen to be 0.2.

## 8 Optimization of a separately trained predictor

To demonstrate the advantages of simultaneously training and the resulting structuring of the latent space, a VAE was trained on the 0 augmented SMILES data without a predictor. In a second step, the training data was encoded into the latent representation with the trained (frozen weights) VAE. These vectors were used as an input for a separate feed-forward neural network to learn to predict the binding energy. This neural network had the same hyperparameters as the property predictor (see Table 1). A MAE loss of 4.8 kcal mol$^{-1}$ was achieved. The same gradient optimizer was used to see how effective this separate predictor neural network performs. As seen in Figure 5, the loss during the molecule generation stays high compared to the simultaneously trained model.

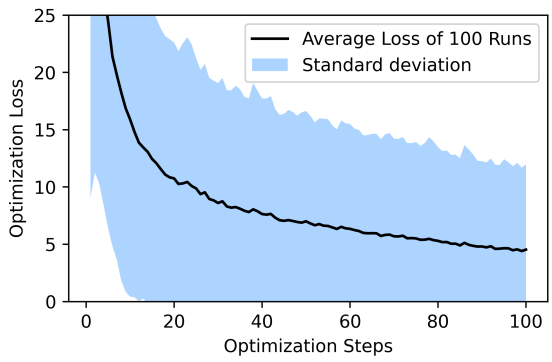


Fig. 5 Loss of the gradient based optimizer for a not simultaneously trained predictor

## 9 Metal Fraction generated molecules


Fig. 6 Metal distribution of the generated molecules with the SELFIES model (left), it can be seen that the model favors *Pd* metal followed by earth-abundant metals *Cu* and *Ni*, the same order of metals is also observed in the SMILES model (right)

## 10 Functional group analysis SMILES model

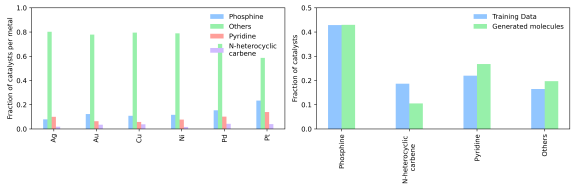


Fig. 7 (left) The distribution of the functional groups in the SMILES model, were phosphines are the dominant functional group (right) the functional group composition of each metal individually.


## Acknowledgements

This publication was created as part of NCCR Catalysis (grant number 180544), a National Centre of Competence in Research funded by the Swiss National Science Foundation.


## Notes and references


1 B. Meyer, B. Sawatlon, S. Heinen, O. A. von Lilienfeld and C. Corminboeuf, *Chem. Sci.*, 2018, **9**, 7069–7077.

2 N. M. O'Boyle, M. Banck, C. A. James, C. Morley, T. Vandermeersch and G. R. Hutchison, *Journal of cheminformatics*, 2011, **3**, 1–14.

3 W. J. Pietro, M. M. Francl, W. J. Hehre, D. J. DeFrees, J. A. Pople and J. S. Binkley, *Journal of the American Chemical Society*, 1982, **104**, 5039–5048.

4 F. Weigend and R. Ahlrichs, *Physical Chemistry Chemical Physics*, 2005, **7**, 3297–3305.

5 M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, B. Mennucci, G. A. Petersson, H. Nakatsuji, M. Caricato, X. Li, H. P. Hratchian, A. F. Izmaylov, J. Bloino, G. Zheng, J. L. Sonnenberg, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, J. A. Montgomery, Jr., J. E. Peralta, F. Ogliaro, M. Bearpark, J. J. Heyd, E. Brothers, K. N. Kudin, V. N. Staroverov, R. Kobayashi, J. Normand, K. Raghavachari, A. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, N. Rega, J. M. Millam, M. Klene, J. E. Knox, J. B. Cross, V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, R. L. Martin, K. Morokuma, V. G. Zakrzewski, G. A. Voth, P. Salvador, J. J. Dannenberg, S. Dapprich, A. D. Daniels, Farkas, J. B. Foresman, J. V. Ortiz, J. Cioslowski and D. J. Fox, *Gaussian09 Revision E.01*, Gaussian Inc. Wallingford CT 2009.

6 G. Landrum, 2022.

7 A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai and S. Chintala, 2019, 8024–8035.

8 W. Falcon et al., *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 2019, **3**, year.

9 P. Schwaller, T. Laino, T. Gaudin, P. Bolgar, C. A. Hunter, C. Bekas and A. A. Lee, *ACS central science*, 2019, **5**, 1572–1583.