SUPPORTING INFORMATION

Inverse Ligand Design: A Generative Data-Driven Model for Optimizing Vanadyl-Based Epoxidation Catalysts

José Ferraz-Caetano[1], Filipe Teixeira[2], M. Natália D. S. Cordeiro[1], Tomoyuki Miyao[3,4]

1) LAQV-REQUIMTE – Department of Chemistry and Biochemistry – Faculty of Sciences, University of Porto - Rua do Campo Alegre, S/N, 4169-007 Porto, Portugal

2) CQUM – Centre of Chemistry, University of Minho, Campus de Gualtar, 4710-057 Braga, Portugal

3) Graduate School of Science and Technology, Nara Institute of Science and Technology, 8916-5 Takayama-cho, Ikoma, Nara 630-0192, Japan

4) Data Science Center, Nara Institute of Science and Technology, 8916-5 Takayama-cho, Ikoma, Nara 630-0192, Japan

Corresponding authors: José Ferraz-Caetano (jose.caetano@fc.up.pt)
M. Natália D. S. Cordeiro (ncordeir@fc.up.pt)
Tomoyuki Miyao (miyao@dsc.naist.jp)

INDEX

Annex S1 – Experimental Epoxidation Reactions Database

The complete database (including bibliographical sources), the list of calculated and experimental descriptors, and files used in model development are available at the following repository: https://github.com/jfcaetano/GenESA. In the database file, we include individual digital identification of all bibliographical sources used to populate the dataset. The database is also presented in our previous article[1]: https://doi.org/10.1039/D3NJ05784D and we also present a synopsis of the same dataset in the version deposited in the Zenodo platform.

This database stems from the methodology built for a machine learning (ML) ready database of vanadium-catalysed epoxidation of small alcohols and alkenes (ESA). It amasses 273 distinct reactions, featuring five different vanadium-catalyst scaffolds: vanadyl(IV) sulphate – [$VOSO_4$], vanadyl acetylacetonate-salen – [VO(salen)], vanadyl isopropoxide – [$VO(O\mathit{i}Pr)_3$], vanadyl acetylacetonate – [$VO(acac)_2$], and vanadyl dichloride-salen – [VCl_2(salen)].

This dataset is a complete collection of descriptors that include experimental reaction descriptors. Molecular descriptors were calculated by the open-source RDKit software, using each SMILES string (generated from the 3D structure) as input, and were assorted into different chemical groups: volume surface area, electronic and structural descriptors. The data underwent rigorous processing, including unit normalization, handling of missing values, and format standardisation. The final dataset comprised more than 90 000 data entries, while the targets were specific reaction outcomes such as yield and enantiomeric excess.

The digital representation (mol file) of each chemical structure is also presented in the repository, in the folder "Mol Files". The chemical species are sorted in each descriptor name: '_Cat' for Catalyst, '_Sol' for Solvent, '_Lig' for Ligand and '_Sub' for Substrate.

We also provide code for descriptor generation in the online repository.

Annex S2 – RDKit Full Descriptor List

Descriptors used in Machine Learning (ML) model development (Table S1 and S2) retrieved from RDKit software. The respective bibliographical foundation of each descripted is described in detail in the RDKit WebBook Documentation[2]. The descriptor calculations were made by converting each SMILES string (representing each molecular entry), running on top of Python version 3.9[3] and using RDKit package, version 2025.03.4[4]. The Chemical information Type is presented, and the chemical species are sorted in the database file in the repository. The code used to generate descriptor values are in the online repository. The database uses as set of 106 for encoding values for each substrate, catalyst, catalyst ligand and oxidant (106 descriptors x 4 types). The descriptors encode electronic, structural and van der Waals surface area (VSA) features.

Table S1

List of RDKit Descriptors used in ML model development (*n* = 106 descriptors).

Descriptor	Chemical information Type	Meaning
BalabanJ (1)	Structural	Distance sum of the two end-vertex for each edge. Connectivity of a molecule based on its graph structure (BalabanJ index has been proven to be relevant to network branching).
BertzCT (1)	Structural	Complexity index, considering both the variety of kinds of bond connectivity's and atom types; information contents related to bond connectivity and atom type diversity.
0χ , 1χ (2)	Structural	This descriptor signifies a retention index derived directly from gradient retention times. Considers individual atoms and their valence (0) and pairs of directly connected atoms (1).
0χn – 4χn (5)	Structural	This descriptor signifies a retention index derived directly from gradient retention times (normalized indexes).
0χv – 4χv (5)	Structural	This descriptor signifies atomic valence connectivity index. Indices weighted by atomic valence

EState_VSA1 – EState_VSA11 (11)	VSA	MOE-type (QSAR model) descriptors using EState indices and surface area contributions. RDKit automatically calculates the EState index for each atom in the molecule, assigning each atom to a predefined EState range (bin). Sums the VSA of atoms within each bin (surface area grouped by EState range).
Hall Kier α (1)	Structural	Descriptor fingerprint that displays the difference between active and inactive molecules.
HeavyAtomCount (1)	Structural	The number of heavy atoms in the molecule.
HeavyAtomMolWt (1)	Structural	The average molecular weight of the molecule ignoring hydrogens.
κ_1 , κ_2 , κ_3 (3)	Structural	This descriptor signifies # κ shape index: $(n-1) \times 2 / m^2$
***Max and Min AbsEStateIndex* (2)**	Electronic	Maximum and minimum absolute E-State
***Max and Min AbsPartialCharge* (2)**	Electronic	Maximum and minimum absolute partial charge
***Max and Min EStateIndex* (2)**	Electronic	Maximum and minimum E-State
***Max and Min PartialCharge* (2)**	Electronic	Maximum and minimum partial charge
Mol logP (1)	Structural	Wildman-Crippen logP value.
MolMR (1)	Electronic	Wildman-Crippen molar refractivity.
MolWt (1)	Structural	The average molecular weight of the molecule.
NHOH Count (1)	Structural	The number of NHs or OHs.
NO Count (1)	Structural	The number of Nitrogens and Oxygens.
$n_{alicarb}$ (1)	Structural	The number of aliphatic carbocycles.
n_{alihet} (1)	Structural	The number of aliphatic heterocycles.
n_{alirig} (1)	Structural	The number of aliphatic rings.
$n_{arocarb}$ (1)	Structural	The number of aromatic carbocycles.

n_{arohet} **(1)**	Structural	The number of aromatic heterocycles.
n_{arorig} **(1)**	Structural	The number of aromatic rings.
n_{Ha} **(1)**	Structural	The number of Hydrogen Bond Acceptors.
n_{Hd} **(1)**	Structural	The number of Hydrogen Bond Donors.
n_{het} **(1)**	Structural	The number of Heteroatoms.
n_{radele} **(1)**	Structural	The number of radical electrons.
n_{rot} **(1)**	Structural	The number of rotatable bonds.
$n_{satcarb}$ **(1)**	Structural	The number of saturated carbocycles.
n_{sathet} **(1)**	Structural	The number of saturated heterocycles.
n_{satrig} **(1)**	Structural	The number of saturated rings.
n_{ele} **(1)**	Structural	The number of valence electrons.
PEOE_VSA1 – PEOE_VSA14 (14)	VSA	MOE-type (QSAR model) descriptors using partial charges estimated using the Gasteiger PEOE and surface area contributions (sum the VSA of atoms within a given partial charge range).
SMR_VSA1 – SMR_VSA10 (10)	VSA	MOE-type (QSAR model) descriptors using atomic contributions to molar refractivity and surface area contributions (sums the VSA of atoms in each bin).
SlogP_VSA1 – SlogP_VSA12 (12)	VSA	MOE-type (QSAR model) descriptors using logP (calculated using the Crippen method) contributions and surface area sum of surface areas of atoms whose logP contribution falls within a specific range).
TPSA (1)	VSA	The total polar surface area of a molecule based upon fragment calculations.
VSA_EState1 – VSA_EState10 (10)	VSA	MOE-type (QSAR model) descriptors using EState indices and surface area contributions. The sum of VSA contributions of atoms whose EState values fall into specific predefined ranges (surface area grouped by predefined VSA ranges).

Experimental descriptors used in ML model development (Table S2) retrieved from each bibliographical source. Specific solvent parameters were retrieved from the Minnesota Solvent Descriptor Database[5]. Each entry represents one descriptor.

Table S2. List of experimental descriptors used in machine learning model development (n = 18).

Descriptor	Chemical Information Type	Meaning
Temp_K (1)	Experimental	Reaction temperature in Kelvin.
Yield (1)	Experimental	Reaction yield.
EE (1)	Experimental	Reaction enantiomeric excess.
Configuration (1)	Experimental	Product geometrical configuration.
Batch Variation (1)	Experimental	Attributes the same value to reactions that have the same experimental procedure conducted under differing conditions.
is_synthetic (1)	Experimental	Attributes values of 0 or 1 if reaction is experimental or from augmented strategy.
Substrate_quant_mmol (1)	Experimental	Quantity of reaction substrate in mmol.
Catalyst_quant_mmol (1)	Experimental	Quantity of reaction catalyst in mmol.
Ligand_quant_mmol (1)	Experimental	Quantity of reaction catalyst ligand in mmol.
Oxidant_quant_mmol (1)	Experimental	Quantity of reaction oxidant in mmol.
Additive_quant_mL (1)	Experimental	Volume of reaction additive in mL.
Solution_vol_mL (1)	Experimental	Volume of reaction solution in mL.
Time_h (1)	Experimental	Reaction duration in hours.
Solvent (1)	Experimental	Reaction solvent.
Solv_opt_freq20 (1)	Experimental	Solvent index of refraction at optical frequencies at 293 K.
Solv_opt_freq25 (1)	Experimental	Solvent index of refraction at optical frequencies at 298 K.
Solv_Hbond_ac (1)	Experimental	Solvent Abraham's hydrogen bond acidity.
Solv_Hbond_bs (1)	Experimental	Solvent Abraham's hydrogen bond basicity.
Solv_surf_tens (1)	Experimental	Solvent surface tension.
Solv_diele_cst (1)	Experimental	Solvent dielectric constant at 298 K.

Annex S3 – Ligand Generative Descriptors Optimization Features

We present the workflow of the code used to generate descriptors for the potential new ESA reactions for each catalyst type. The scripts for algorithm development and hyperparameter optimization are in the article's online repository, in the "Scripts" folder. The full statistical results are available in the "Results" folder of the online repository.

1. Data Preparation

i) Load datasets:

- *Augmented_ESA_Vanadium_Database.csv* (main dataset with experimental and computed descriptors).
- *Database_6.csv* (used for defining target output columns).

ii) Handling missing values by replacing them with zeros and define columns to exclude from feature selection.

iii) Identify feature set *X_names* by excluding categorical/non-numerical descriptors and target variables.

2. Feature Engineering and Scaling:

i) Define input features (X), target variable (y = Yield) and standardize features using *StandardScaler* to ensure numerical stability.

3. Model Training

i). Initialize a *Random Forest Regressor* with optimized hyperparameters (n_estimators=350, max_depth=40, random_state=47).

ii) Train the model on the scaled feature matrix X_scaled to predict the experimental Yield.

4. Descriptor Constraints

i) Define constraints for feature validity:

Strictly positive descriptors (e.g., solution volume, temperature, reagent concentrations).

Non-negative descriptors (e.g., solvation properties like surface tension or aromaticity).

5. Alternative Descriptor Generation

i) For a given target yield and catalyst structure, subset the dataset to only entries matching the catalyst structure.

ii) Apply perturbation to existing scaled feature vectors to simulate alternative plausible reaction conditions.

iii) Predict the yield for each perturbed feature vector using the trained model and check if is within a tolerance window of the target yield.

iv). Inverse transform valid feature vectors back to the original scale, and enforce descriptor constraints, ensuring only chemically meaningful conditions are accepted.

6. Adaptive Sampling Strategy

i) Perturbations are iteratively applied up to a maximum number of attempts (10,000).

Note: If insufficient alternatives are found, the yield tolerance is dynamically increased to improve sampling success.

7. Batch Processing of Yield-Catalyst Pairs

i) Define a list of (target*_yield*, *original_cat_structure*) pairs across a range of yields (ten bins over the 0–100% range) and three catalyst types: $VOSO_4$, $VO(O\textit{i}Pr)_3$ and $VO(acac)_2$.

Note: For each pair, alternative generation routine to produce up to 1000 valid descriptors.

8. Aggregation and Output

i) Combine all valid alternative descriptors and ensure uniform column structure.

ii) Export the complete dataset to a CSV file named *Lig-Generated_Descriptors.csv*.

Best hyper-parameters for each algorithm for model optimization:

Random Forest Regressor = n_estimators= 150, min_samples_split= 3, min_samples_leaf= 1, max_features='sqrt', max_depth=None, bootstrap= False, random_state=47

Annex S4 – Database Sources for Generative Model Training

Two open-source datasets were used at different parts of the ML pipeline to enable the optimized descriptor sets to be translatable back into synthetically viable molecular structures.

1. ChEMBL 31 (Curated Subset)

i) Used as an initial training and benchmarking dataset for evaluating various generative model architectures.

ii) Contains over 20,000 curated chemical structures.

iii) Selected for its broad chemical diversity and well-annotated molecular representations.

iv) Serves as a foundational training set to assess generative performance before scaling to larger databases.

Access Links: full ChEMBL 31 dataset; curated 20k ChEMBL 31 dataset is available in the folder 'Datasets' in the article's online repository.

2. Mcule Commercial Database (MCD)

i) Used for final training of the selected generative model to ensure outputs are commercially viable.

ii) Comprises approximately 6 million molecules, curated to include only chemically valid, supplier-available compounds.

iii) Pre-processing includes structural standardization, removal of unstable compounds and validation of supplier availability.

iv) Enables the generative model to learn biases towards purchasable molecules.

The Mcule Commercial dataset is available in the folder (*In Stock Subset*) at https://mcule.com/database/. (*Mcule - Online drug discovery platform database)*

The RDKit descriptors mentioned in Annex S2 were calculated using the SMILES representation for each dataset. The code is also available in the article's online repository.

Annex S5 – Vocabulary & Model Architecture

1. Model Vocabulary Overview

The vocabulary code is a fundamental component to produce valid SMILES strings for the three generative models described above. It is a standard representation of the chemical information encoded in SMILES strings, to allow the model to generate valid structures. The vocabulary defines the set of characters and tokens, such as atomic symbols (C, O, N), bond types (-, =, #), branching symbols ((,)), and ring-closing identifiers (1, 2, etc.), that the model can use during training and generation. The vocabulary code parses all SMILES strings in the training dataset to identify every unique token. These are then assigned unique numerical indices, creating a mapping that the model can reference. Special tokens such as <PAD> (padding), <START> (sequence beginning), and <END> (sequence ending) are included. They guarantee consistent SMILES processing, while also simplifying validation. This tokenization and mapping are essential for generative architectures. For RNNs, the vocabulary aids token sequential prediction, while transformers look at the entire sequence at the same time with attention mechanisms. For the GPT, the vocabulary is critical for step-by-step prediction, generating one token at a time.

2. Model Architecture Overview

2.1. Recurrent Neural Network (RNN)

Figure S1 presents a recurrent neural network (RNN) for molecular generation. A vector of RDKit descriptors was used to initialize the hidden state and concatenated with each SMILES token. The RNN is made up of a series of LSTM or GRU cells to process the input sequentially. The RNN updates its hidden state and produces each character of the SMILES string. The <START> token denotes the starting sequence and the <END> token denotes the end of the sequence.

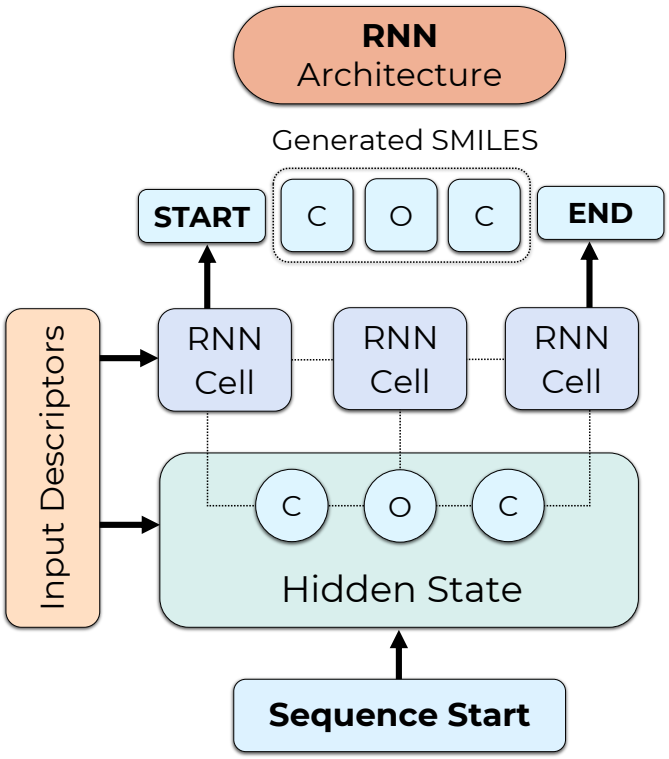

Figure S1. Schematic representation of the deployed RNN model architecture.

2.2. Transformer

Figure S2 illustrates a Transformer architecture developed for the generation of a SMILES from molecular descriptors. The descriptor vectors are projected to an embedding space and then fed together with positional encodings into an encoder. The decoder then autoregressively generates the SMILES string, token-by-token, while considering both previously generated tokens as well as the descriptor features encoded by the encoder. The use of multi-head self-attention further enables the model to capture long-range information, such as matched brackets or ring closures within a SMILES sequence.

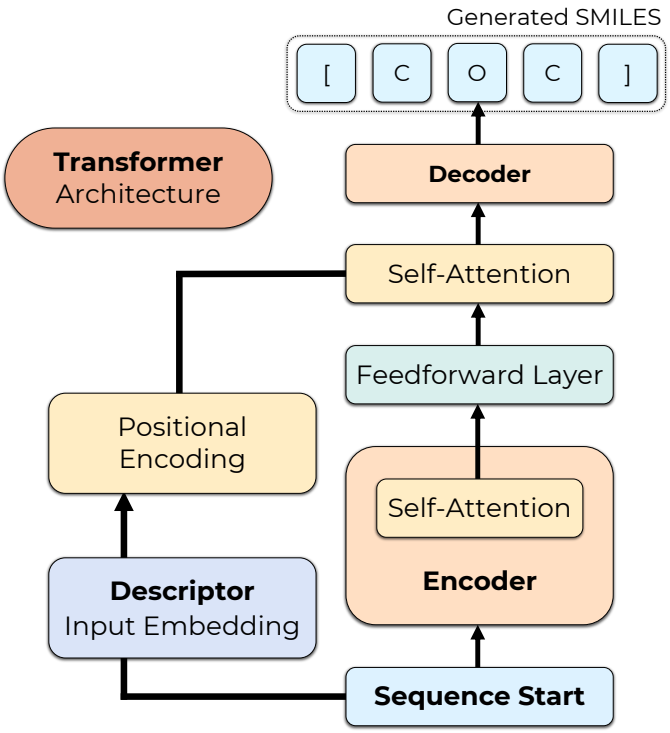

Figure S2. Schematic representation of the deployed Transformer model architecture.

2.3. Generative pre-trained transformer (GPT)

Figure S3 illustrates a GPT-style model trained to produce SMILES strings in an autoregressive way. The model is pretrained using large SMILES corpora and then fine-tuned with RDKit descriptor–SMILES pairs. Each input embedding is passed through a single Transformer decoder block with masked self-attention, with the vector of descriptors added as a special token. The model generates each SMILES token one at a time, conditioned on the previously-generated tokens and the descriptor context.

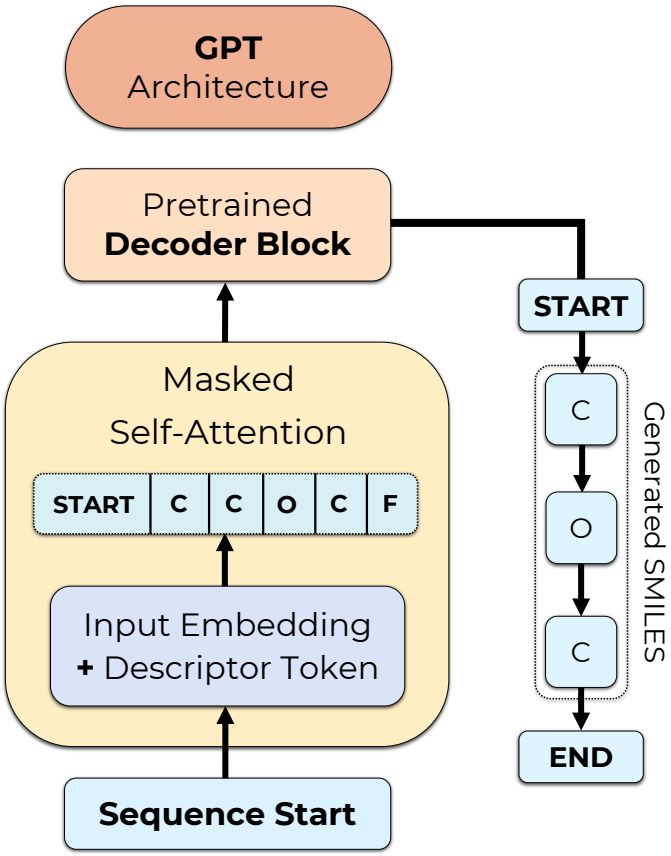

Figure S3. Schematic representation of the deployed GPT model architecture.

3. Model Architecture Training Results

3.1. Initial Testing (20k ChEMBL Dataset)

Learning efficiency was tested for each model architecture using a sample training set of 20,000 ChEMBL molecules, summarized by learning curves that track the token-level cross-entropy as a function of cumulative molecules seen during training (code provided in the article's online repository in the 'Scripts' folder). We used an epoch learning chart to evaluate the models, revealing their convergence behavior, and ability to minimize loss. The loss is computed over non-PAD tokens under teacher forcing, revealing their learning efficiency. The loss has no physical units as it is a dimensionless quantity because of the implementation of a natural logarithm. Each epoch corresponds to a full pass over the training set, so with 20,000 ChEMBL molecules, every epoch exposes the model to all 20k molecules once. This counts exposures rather than unique molecules, since the same molecules are revisited after shuffling each epoch. Figure S4 presents the training curves for each for generative model.

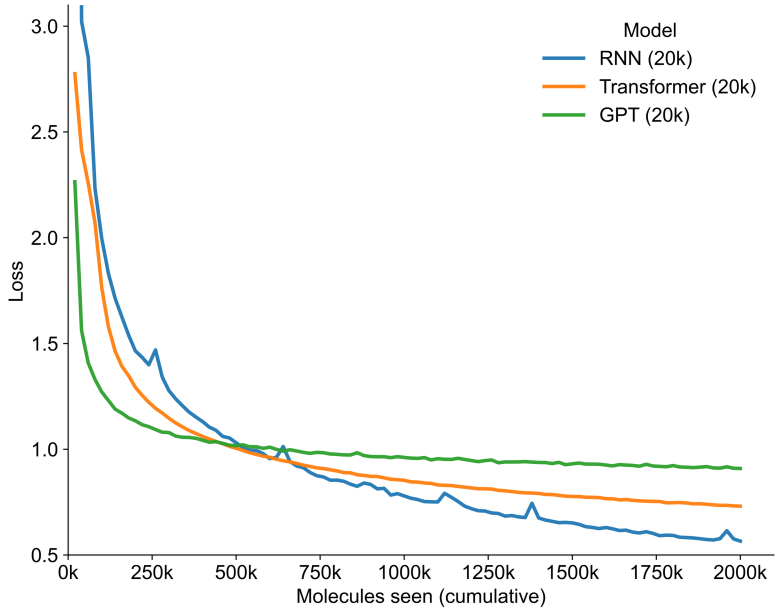

Figure S4. Training loss versus epoch for generative models (RNN, transformer, GPT) on 20,000 ChEMBL molecules sample set, across 100 epochs.

The learning curves present three learning stages during the 100 epochs. In the first stage (0–250k molecules), the GPT variant is most sample-efficient (loss about 2.3 to 1.3), followed by the Transformer (2.8 to 1.35) and the RNN (3.1 to 1.45). Between 400–600k molecules, the RNN closes the gap and overtakes the self-attention models (≈ 1.0 to 1.05 near 500k). Beyond ~1M molecules, improvements are gradual, as by 2.0M molecules the RNN attains the lowest loss (≈ 0.60), the Transformer is intermediate (≈ 0.78), and GPT plateaus (≈ 0.90).

3.2. Optimized Testing (6M Mcule Dataset)

After analyzing each model architecture generative results (presented in Figure 3 of the manuscript). We selected the Transformer architecture to be trained in the full 6 million molecules Mcule dataset. Figure S5 presents the learning curve for this model training scenario.

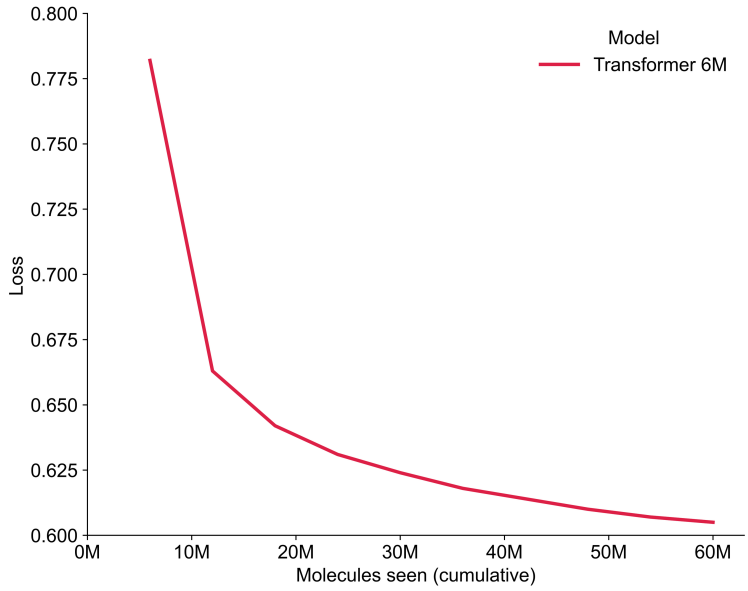

Figure S5. Training loss versus epoch for transformer generative model on the on the full 6 million Mcule dataset, across 10 epochs.

The results show that the performance of the transformer mode is higher when trained on the MCD dataset. The learning curve (Figure S5) reveals that the loss gradually decreases from about 0.800 to approximately 0.600. Although the training was limited to 10 epochs due to the exhaustion of computational resources, the consistent decline in loss indicates effective generalization.

3.3. Molecular Diversity Evaluation (Model trained on the 6M Mcule Dataset)

To quantify the structural diversity of generated ligands by the transformer model trained on the Mcule dataset, we filtered valid SMILES and deduplicating by canonical SMILES to ensure unique structures. We then computed Morgan fingerprints (radius = 2, 2048 bits), evaluating pairwise Tanimoto similarities without materializing the full matrix. It measures the mean pairwise similarity, where lower similarity implies higher diversity. In our model, structure diversity between the generated ligands was high, backed by a calculated Tanimoto similarity of 12.8 %. The script for evaluating the diversity of generated ligands is in the article's online repository, in the folder Scripts.

Annex S6 – RNN and Transformer (20k) Generation Analysis

Here we present the distribution of generated ligands across target yield bins for the model testing stage. Figures S6 and S7 depict the results for the RNN 20k and Transformer 20k models as they were the ones yielding acceptable validity results. These comparative plots illustrate how different generative architectures influence molecular diversity, based on training with a fixed sample of ChEMBL 31 20,000 compounds.

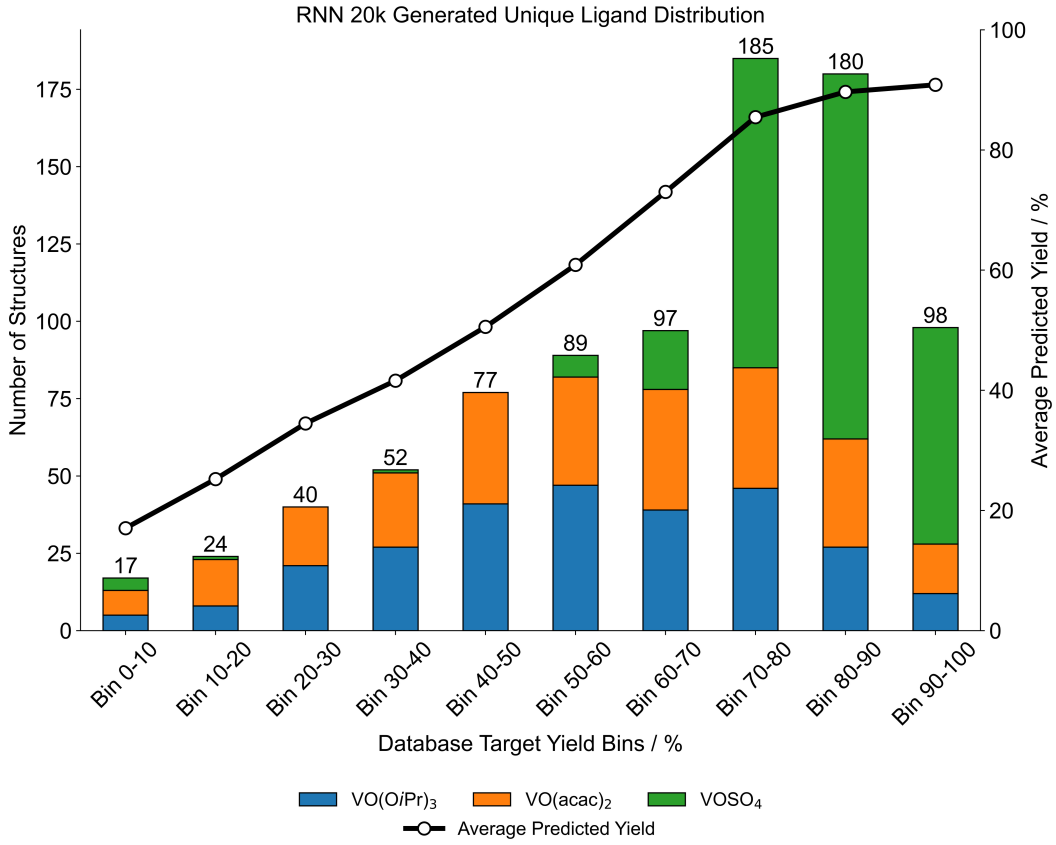

Figure S6. Distribution of unique ligands generated by the RNN 20k ChEMBL 31 model across target yield bins, with the number of structures generated for each catalyst type and their average predicted yield.

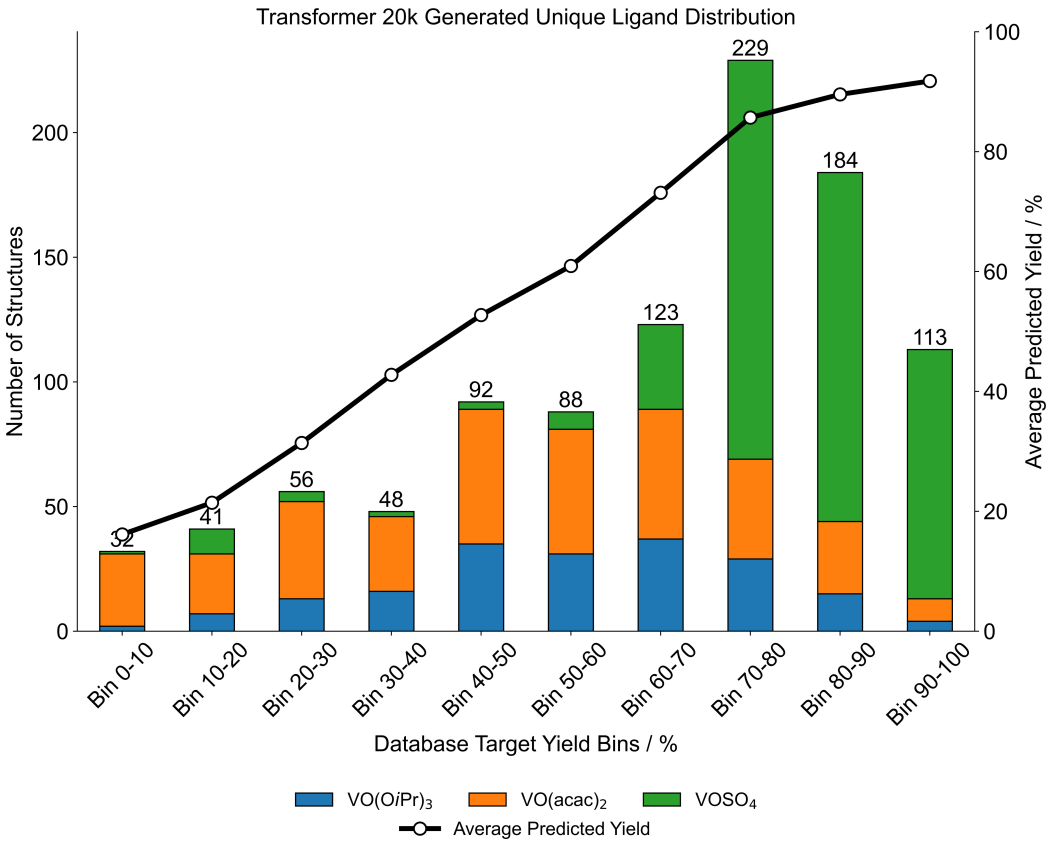

Figure S7. Distribution of unique ligands generated by the Transformer 20k ChEMBL 31 model across target yield bins, with the number of structures generated for each catalyst type and their average predicted yield.

Annex S7 – PCA and t-SNE Analysis of Generated Ligands

This annex presents the visualization of the descriptor space and ligand-catalyst relationships are key in highlighting clusters, outliers and the data dispersion. PCA can identify the key dimensions driving variance in the dataset, helping to uncover patterns, while t-SNE can complementary reveal finer-grained structures in the data, such as how ligands and catalysts cluster based on compatibility or similarity scores. This helps assessing whether the model's predictions are distributed across the descriptor space or concentrated in specific regions. Figures S8-S9 present the t-SNE plots and S10-S11 the PCA charts.

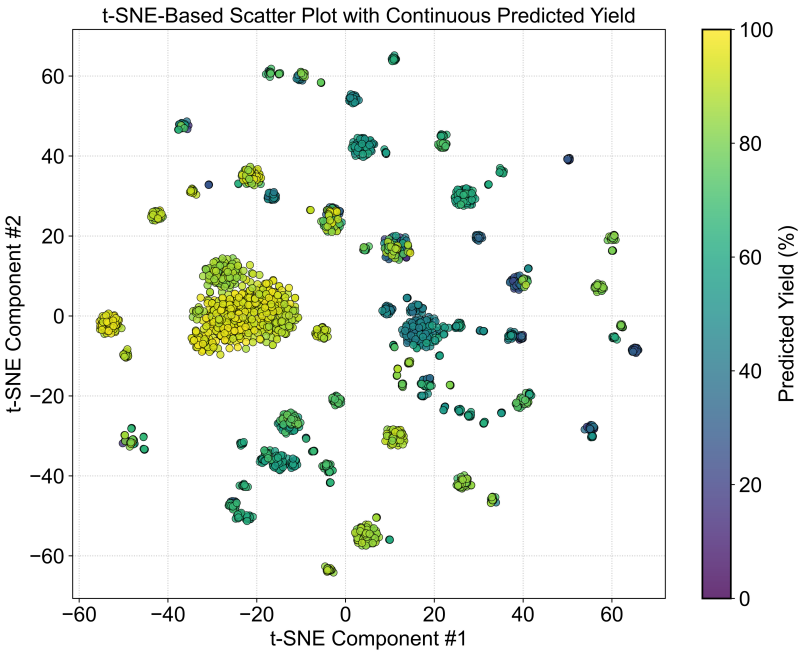

Figure S8. t-SNE visualization of generated ligand descriptor space for ESA vanadyl-based catalysts, showing ligand clustering with continuous predicted yield as a color gradient.

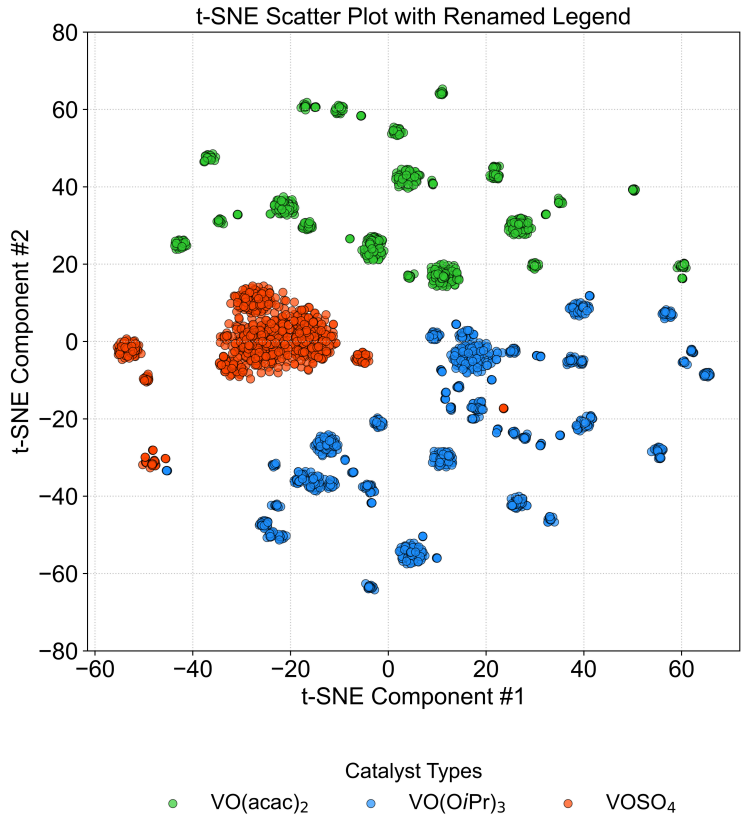

Figure S9. t-SNE visualization of generated ligand descriptor space for ESA vanadyl-based catalysts, showing ligand clustering by catalyst type.

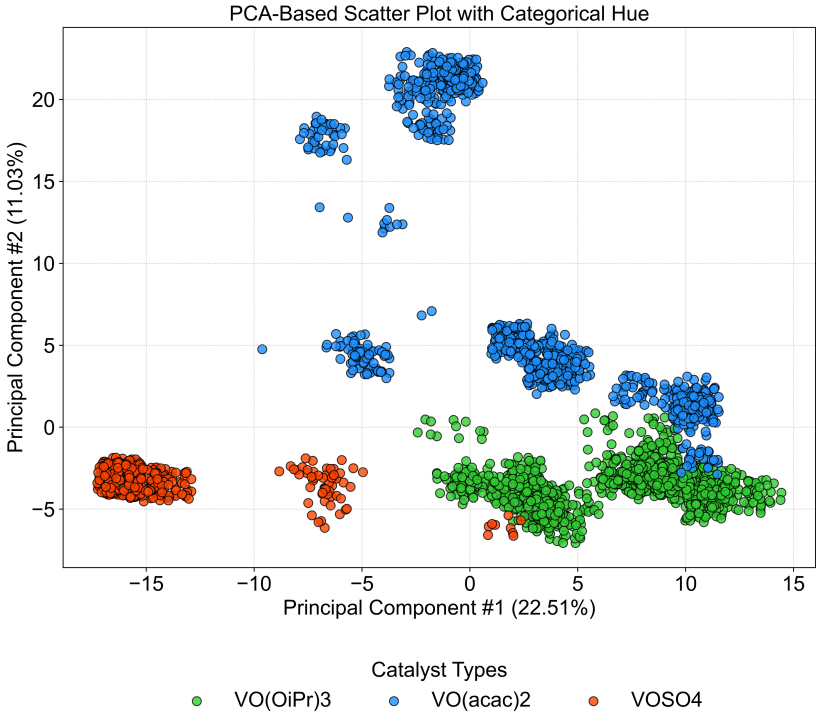

Figure S10. PCA of ligand descriptor space showing the distribution of ligands based on their descriptors, color-coded by catalyst type.

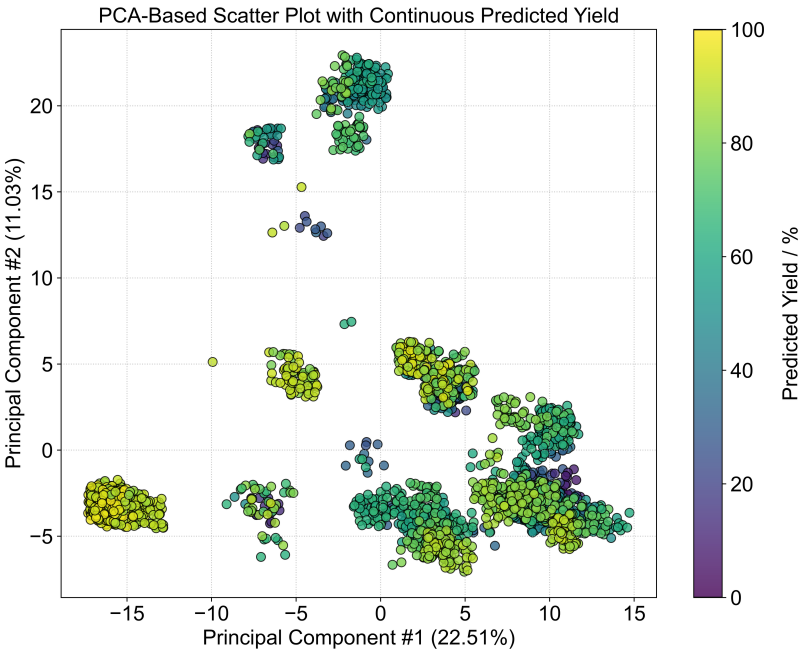

Figure S11. PCA of ligand descriptor space showing the distribution of ligands based on their descriptors, with predicted yield as a continuous color gradient.

Annex S8 – Synthetic Accessibility Scores for ESA Dataset and Generated Structures

This section outlines the complementary Synthetic Accessibility Scores (SAS) for the original experimental ESA dataset (Figure S12) and for the generated substrates for purchasable generated ligands (Figure S13). The code used to calculate the SAS is presented in the model's repository, using the workflow described by Schuffenhauer's method[6] in the following repository: https://github.com/rdkit/rdkit/tree/master/Contrib/SA_Score.

A precomputed fragment open-source score database was loaded, which assigns scores to molecular fragments based on their prevalence in known chemical libraries. A Morgan fingerprint was generated for each ligand, and each fragment score was totaled and normalized to create the base score. Additional penalties were applied to account for molecular complexity, stereochemistry (chiral centers), macrocyclic rings, spiro and bridgehead atoms. A sparsity penalty was added if the number of non-zero fingerprint elements exceeded the number of atoms. These components were combined into a raw SAS value, rescaled to a range of 1 (most accessible) to 10 (least accessible).

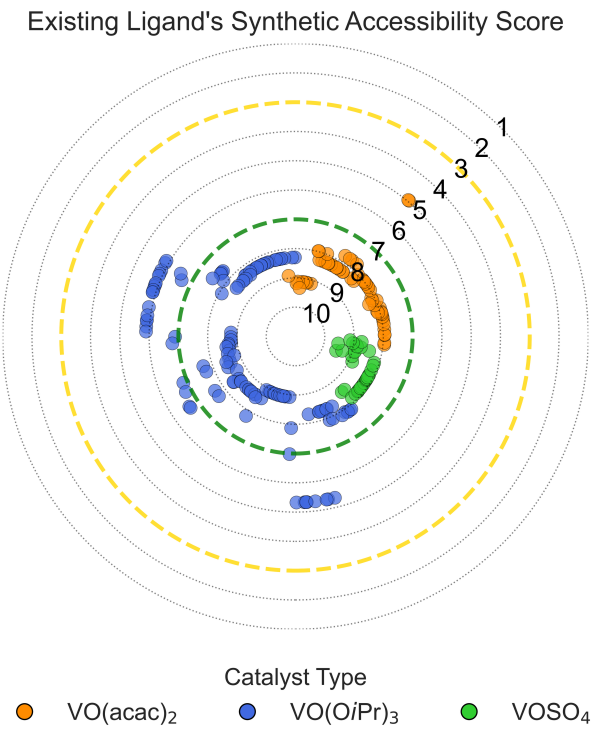

Figure S12. Synthetic Accessibility Score distribution of existing ESA ligands by catalyst type. The circular plot represents the synthetic feasibility of new ligands, with scores ranging from 10 (easiest to synthesize, center) to 1 (most challenging, outer regions). The green line represents the determined 'optimal' distribution of known ESA ligand reactions, while the yellow delimitates the 'moderate' area.

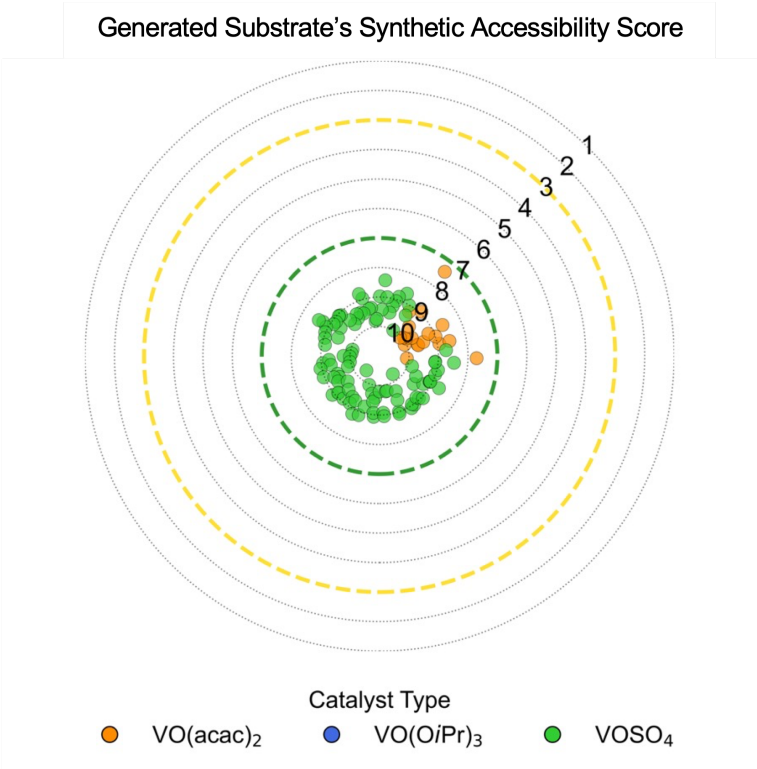

Figure S13. Synthetic Accessibility Score distribution of generated substrates for purchasable generated ligands by catalyst type. The circular plot represents the synthetic feasibility, with scores ranging from 10 (easiest to synthesize, center) to 1 (most challenging, outer regions).

Annex S9 – Sample Code for ML Model Design

The scripts for algorithm development are in the articles online repository, in the folder Scripts.

Scripts Overview

1. ESA Descriptor Generator

This script generates optimized experimental descriptors targeting specific yields for vanadium-based catalytic reactions. It uses an ML perturbation approach to simulate plausible reaction conditions. The outputs are used to inform downstream ligand structure generation.

1) Loading and preprocessing the ESA-Vanadium dataset.
2) Training a RandomForestRegressor to predict reaction yields.
3) Applying controlled random perturbations to descriptor vectors to generate new data points that match a target yield within a defined tolerance.
4) Filtering generated alternatives to ensure chemical plausibility using predefined descriptor constraints (e.g., positive values for temperature, concentration, etc.).
5) Iterating over a grid of catalyst structures and target yields to produce a comprehensive set of yield-sensitive descriptor alternatives.
6) Saving the final dataset (Lig-Generated_Descriptors.csv) with predicted yields and conditions for each generated entry.

2. Descriptor Generator ChEMBL 31 20k

Code calculates RDKit-based molecular descriptors for a dataset of ligands generated from the 20k ChEMBL-derived model. Key functionalities include:

1) Defining a curated list of RDKit molecular descriptors, including structural indices (e.g., Chi, Kappa), van der Waals surface area (VSA), and electronic properties (TPSA, EStateIndex).
2) Computing descriptors for each molecule, while handling invalid SMILES by returning NaNs.
3) Outputting the enriched dataset (chemlb-20k-rdkit.csv) to Google Drive for downstream model training, visualization, or ligand optimization.

3. Descriptor Generator Mcule Commercial Database 6M

Code calculates molecular descriptors for a large-scale commercial compound dataset from the Mcule Commercial Database (MCD), profiling purchasable molecules for use in generative model training. It includes a validation step to ensure SMILES quality.

1) Defining a curated list of molecular descriptors from RDKit, including topological indices (e.g., Chi, Kappa), surface area (VSA), and physicochemical properties (TPSA, BalabanJ).
2) Processing the dataset in chunks of 1 million SMILES, applying parallelized descriptor calculations using the swifter and dask libraries for efficiency.
3) Appending descriptor columns to the original data and progressively writing the results to disk (purchase-rdkit.csv) to manage memory use during computation.
4) Validating and cleaning the descriptor-annotated dataset by filtering only valid SMILES strings via RDKit parsing.

4. ESA RNN Model Training

This script implements a recurrent neural network (RNN) model that learns to translate numerical reaction descriptors into valid SMILES strings from the ChEMBL-derived dataset.

1) Defining a tokenization scheme for SMILES strings using regular expressions and building a vocabulary with special tokens (START, END, PAD).
2) Encoding SMILES strings into token sequences and pairing them with molecular descriptors for supervised learning.
3) Creating a MoleculeDataset class and corresponding DataLoader with proper sequence padding and batching.
4) Implementing a custom RNN model (DescriptorToSMILESModel) that takes descriptors as input and generates SMILES character-by-character using a GRU-based decoder.
5) Training the model over multiple epochs using a cross-entropy loss function, with early stopping triggered by a target loss threshold.
6) Saving the trained model (01-RNN-descriptor_to_smiles_model.pth) and vocabulary for future generation or fine-tuning.

<u>RNN Model Parameters</u>

Embedding Dimension: 276

Hidden Dimension (GRU): 148

RNN Type: GRU (Gated Recurrent Unit)

Vocabulary Size: Dynamic (len(vocabulary))

Descriptor Vector Size: descriptor_length = descriptors.shape[1]

Batch Size: 42

Loss Function: CrossEntropyLoss(ignore_index=pad_token_id)

Optimizer: Adam with learning rate 0.0001

Maximum Epochs: 100

Early Stopping: Stops if avg_loss ≤ 0.3

• Logging and exporting the training loss for each epoch to a CSV file (01-RNN-training_loss_epochs.csv) for performance tracking.

5. ESA Transformer Model Training

This code implements a Transformer-based architecture that maps numerical reaction descriptors to valid SMILES strings. It is used to traind on the ChEMBL 31 and MCD datasets.

1) Loading the descriptor-enriched SMILES dataset (ChEMBL 31 or MCD).
2) Building a custom SMILES tokenizer and vocabulary, including handling of special tokens (START, END, PAD) and rare patterns such as Cl, Br, and ring closures.
3) Encoding descriptor-SMILES pairs for training, using PyTorch Dataset and DataLoader classes with sequence padding.
4) Defining a Transformer decoder architecture, which maps learned descriptor embeddings to character-level SMILES sequences via attention layers and positional encodings.
5) Supporting both training and autoregressive generation modes with temperature-controlled sampling for diverse SMILES outputs.
6) Training the model using cross-entropy loss, with logging of per-epoch loss and early stopping.

<u>Transformer Model Parameters:</u>

Embedding Dimension: 24

Number of Attention Heads: 6

Number of Decoder Layers: 3

Positional Encoding: Sinusoidal, dynamically computed

Feedforward Layer Size: Implicit (default: same as embedding in nn.TransformerDecoderLayer)

Vocabulary Size; Dynamic (len(vocabulary))

Descriptor Vector Size: descriptor_length = descriptors.shape[1]

Batch Size: 47

Dropout: Not specified explicitly (uses PyTorch defaults for nn.TransformerDecoderLayer)

Loss Function: CrossEntropyLoss(ignore_index=pad_token_id)

Optimizer: Adam with lr = 0.001

Maximum Epochs: 100

Early Stopping: Stops if avg_loss <= 0.1

• Saving the trained model (02-Trn-descriptor_to_smiles_model.pth), vocabulary (02-Trn-vocabulary.pth), and loss tracking (02-Trn-training_loss_epochs.csv) for reproducibility.

6. ESA GPT Model Training

Code implements a GPT-style Transformer model to translate numerical reaction descriptors into SMILES strings. It leverages multi-head self-attention and positional encoding to capture long-range dependencies in molecular sequences.

1) Loading descriptor-annotated SMILES data.
2) Implementing a custom SMILES tokenizer and vocabulary builder, including support for special characters, ring structures, and positional information.
3) Preparing descriptor–SMILES pairs in PyTorch-compatible Dataset and DataLoader objects with dynamic padding.
4) Defining the VanillaGPTSMILES model, a Transformer-based architecture with: SMILES embeddings and sinusoidal positional encodings; Linear projection of descriptor vectors into token space; Stacked Transformer decoder blocks with masked attention
5) Training the model using cross-entropy loss and Adam optimization, with logging of epoch-wise loss and early stopping when reaching a desired loss threshold.

<u>GPT Model Parameters:</u>

Embedding Dimension: 36

Number of Attention Heads: 6

Number of Decoder Layers: 3

Feedforward Hidden Size (MLP dimension): 36

Dropout (Positional Encoding): 0.1

Dropout (Attention & MLP): 0.2

Vocabulary Size: dynamic (computed from tokenized SMILES; len(vocabulary))

Descriptor Vector Length: descriptors.shape[1] (depends on your dataset)

Batch Size: 47

Loss Function: CrossEntropyLoss(ignore_index=pad_token_id)

Optimizer: Adam with learning rate 0.001

Maximum Epochs: 100
Early Stopping Criterion: avg_loss <= 0.1

• Saving the trained model (03-GPT_decriptor_to_smiles_model.pth), vocabulary (03-GPT-vocabulary.pth), and training logs (03-GPT-training_loss_epochs.csv).

7. ESA RNN Generative SMILES Model

This script performs SMILES generation from optimized reaction descriptors using a previously trained RNN model and evaluates chemical validity:

1) Loading trained RNN model (01-RNN-descriptor_to_smiles_model.pth), its corresponding vocabulary (01-RNN-vocabulary.pth), and Test descriptors generated from prior model steps (Lig-Generated_Descriptors.csv)
2) Reconstructing the SMILES tokenizer and vocabulary to support sequence decoding.
3) Initializing and loading the trained RNN model, including architecture parameters (embedding size, hidden size).
4) Feeding the test descriptor set through the model to generate SMILES strings using greedy decoding, one token at a time.
5) Checking the chemical validity of generated SMILES using RDKit (MolFromSmiles).
6) Summarizing generation performance, total outputs, number and percentage of valid structures.
7) Saving the final annotated dataset, including generated SMILES and validity flags, to 01-RNN-Generated_SMILES.csv.

8. ESA Transformer Generative SMILES Model

This code uses a trained Transformer decoder model to generate SMILES strings from optimized ligand descriptors and assess their chemical validity. It includes:

1) Loading trained Transformer model (ChEMBL or MCD), its vocabulary (ChEMBL or MCD), and the descriptor-based test set (Lig-Generated_Descriptors.csv)
2) Reconstructing the tokenizer and vocabulary structure with special token mappings.
3) Rebuilding the Transformer model with the original architecture parameters, including multi-head attention and positional encodings.
4) Applying greedy decoding to generate SMILES strings from each descriptor vector using the .sample() method.

5) Evaluating the chemical validity of generated structures using RDKit (Chem.MolFromSmiles).
6) Summarizing performance statistics: number and percentage of valid vs. invalid SMILES.
7) Saving the final output to 02-Trn-Generated_SMILES.csv.

9. ESA GPT Generative SMILES Model

This code generates SMILES strings from ligand descriptors using a trained GPT-style Transformer decoder and evaluates the chemical validity of the outputs.

1) Loading the trained GPT model (03-GPT_decriptor_to_smiles_model.pth), associated vocabulary (03-GPT-vocabulary.pth) and descriptor test data (Lig-Generated_Descriptors.csv)
2) Reconstructing the SMILES tokenizer and vocabulary, including positional token mapping and decoding functionality.
3) Defining the Transformer decoder architecture with descriptor-to-embedding projection, positional encoding and stacked decoder blocks with masked multi-head self-attention.
4) Generating SMILES using autoregressive decoding stopping upon reaching the end token.
5) Evaluating chemical validity of each generated SMILES string using RDKit parsing.
6) Summarizing generation outcomes in terms of valid/invalid structure counts and percentages.
7) Exporting results, including generated SMILES, validity flags.

10. Generated SMILES Evaluation

Comprehensive evaluation of SMILES strings generated by multiple descriptor-to-structure models. It computes validity, uniqueness, novelty, and descriptor-based similarity against reference datasets.

1) Loading SMILES from generated output files.
2) Assessing chemical validity, uniqueness (non-redundant valid SMILES), and novelty against two reference libraries: ChEMBL31 or MCD and ESA experimental library.
3) Computing RDKit descriptors for valid generated molecules using topological, electronic, and spatial properties (e.g., Chi, TPSA, VSA).
4) Calculating cosine similarity between the generated molecule's descriptors and their corresponding optimized descriptor targets.
5) Aggregating results across four generated SMILES datasets.
6) Summarizing evaluation metrics.

11. Ligand SMILES Filtering and Curation

This script performs systematic curation, canonicalization, and validation of Ligands SMILES strings generated via descriptor-to-structure pipelines. It applies a multi-step filtering protocol to ensure chemical plausibility, structural consistency, and coordination compatibility of candidate ligands for vanadium binding.

1) Canonicalizing all SMILES using RDKit to ensure standardized representations prior to analysis and filtering.
2) Performing detailed molecular validation with RDKit, including sanitization error checks, valence correctness evaluation, kekulization feasibility and atom count and charge assessment
3) Filtering invalid molecules based on:Molecular size (minimum number of atoms), excessive character repetition (syntactic artifacts), presence of multiple disconnected structures, overall neutrality (formal charge = 0) and inclusion of vanadium-compatible donor motifs (via SMARTS-based substructure screening – full SMARTS list is in the code).
4) Integrating RDKit-based red flag diagnostics into a structured DataFrame.
5) De-duplicating filtered entries based on both SMILES and their associated categorical labels.

12. ESA Ligand Feature Importance Comparison

This script conducts comparative feature importance analysis across multiple SMILES-generated datasets by leveraging the optimized Random Forest regression models trained to predict catalytic reaction yields. It includes:

1) Loading and preprocessing multiple curated SMILES datasets:
2) Standardizing column references across datasets to identify.
3) Subsetting data by four catalyst identities: $VO(acac)_2$, $VOSO_4$, $VO(O\textit{i}Pr)_3$ and All_Catalysts (aggregated across all entries).
4) Training a Random Forest model for each catalyst condition in each dataset to predict yield from ligand descriptors and compute descriptor feature importance values.
5) Aggregating and exporting all feature importance scores in a structured format.
6) Performing pairwise comparisons of feature importance against the ESA reference for each catalyst, calculating the difference in importance values: Δ(External – ESA).

13. ESA Feature Correlation Analysis

This script performs a comparative analysis of feature importance profiles across descriptor-to-structure generative models using Spearman rank correlation and standard deviation.

1) Loading precomputed feature importance scores, previously generated.
2) Dataset normalization, grouping descriptor columns by dataset origin.
3) Global correlation benchmarking across catalysts:
4) For each catalyst ($VO(acac)_2$, $VOSO_4$, $VO(OiPr)_3$ and All_Catalysts), the script computes:
5) Spearman correlation coefficient ($\rho \times 100$) between ESA and each generated dataset.
6) Mean absolute difference in importance scores (Δ).
7) Number of common descriptors used in each comparison.

14. ESA Ligand & Catalyst Compatibility

This script evaluates descriptor-level compatibility and chemical similarity between ligands and vanadium-based catalysts, using experimental data and generated ligands. It includes:

• Parsing descriptor-annotated datasets, dynamically matching descriptor columns for ligand–catalyst pairs (e.g., _Lig and _Cat suffixes), enabling scalable cross-comparison across matched chemical features.

• Compatibility scoring:

i) Computes a Gaussian-weighted similarity between ligand and catalyst descriptors.

ii) Supports alternative methods such as scaled difference and sigmoid-based comparison (modular implementation).

iii) Aggregates scores to produce a single compatibility metric for each pair.

• Structural similarity with cosine similarity between ligand descriptor vectors. It also calculates this independently of the compatibility score

• Final combined score: weighted combination: 70% descriptor compatibility + 30% cosine similarity

• Two application modes:

i) New ligands vs. experimental catalysts (cross-validation of generative pipeline)

ii) Known ligands vs. known catalysts (internal benchmarking of ESA dataset)

15. ESA Synthetic Accessibility Score Calculation

This script calculates the Synthetic Accessibility Score (SAS) for a set of ligands, based on their SMILES representations. It is also used to assess the SAS for the existing ESA ligands and the generated substrates, assessing synthetic feasibility via de novo molecular design.

• Loading ligand SMILES from a CSV file.

• Computing Morgan fingerprints for each molecule using RDKit, with a radius of 2 to capture local atom environments.
• Retrieving fragment-based synthetic accessibility scores from a precompiled fragment penalty table (fpscores.pkl.gz) and combining them with structure-based penalties.
• Structural complexity penalties include:
i) Atom count size penalty
ii) Stereochemical complexity (number of chiral centers)
iii) Spiro and bridgehead atom penalties
iv) Macrocyclic ring penalty
v) Fingerprint density penalty (related to fragment diversity *vs* size)

• Final SA Score Transformation: Raw scores are scaled and normalized to range from 1 (very easy to synthesize) to 10 (very difficult)
• Batch Processing and Export results.

16. New Substrate Generation

Script generates substrate SMILES from descriptor vectors using a trained transformer decoder and evaluates structural validity and descriptor similarity (same decoded for ligand SMILES generation).

• Model Inference

Loads pre-trained descriptor-to-SMILES model and vocabulary.
Inputs substrate descriptors (_Sub).
Generates SMILES via decoding from descriptor embeddings.

• SMILES Generation & Validation

Converts token sequences to SMILES strings.
Assesses chemical validity using RDKit.
Cleans and formats output SMILES.

• Similarity Analysis

Calculates RDKit descriptors for generated SMILES.
Computes cosine similarity to original descriptors.

References

(1) Ferraz-Caetano, J.; Teixeira, F.; Cordeiro, M. N. D. S. Navigating epoxidation complexity: building a data science toolbox to design vanadium catalysts. *New Journal of Chemistry* **2024**, *48* (12), 5097-5100, DOI: https://doi.org/10.1039/D3NJ05784D.

(2) *RDKit: Open-source cheminformatics - Descritptor Guide Online Webbook*. https://www.rdkit.org/docs/GettingStartedInPython.html#list-of-available-descriptors (accessed July 1st, 2025).

(3) *Python Software Foundation - Python Language Reference, version 3.9.8*. http://www.python.org (accessed *July 1st, 2025).*

(4) Landrum, G. *RDKit: Open-source cheminformatics* 2025_03_4 (Q1 2025) Release - June 2025. http://www.rdkit.org/ (accessed July 1st, 2025).

(5) Marenich, A. V. K., Casey P; Thompson, Jason D; Hawkins, Gregory D; Chambers, Candee C; Giesen, David J; Winget, Paul; Cramer, Christopher J; Truhlar, Donald G. *Minnesota Solvation Database (MNSOL) version 2012. Retrieved from the Data Repository for the University of Minnesota*. 2020. DOI: https://doi.org/10.13020/3eks-j059.

(6) Ertl, P.; Schuffenhauer, A. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of Cheminformatics* **2009**, *1* (1), 8. DOI: https://doi.org/10.1186/1758-2946-1-8.