Digital Discovery

ROYAL SOCIETY OF CHEMISTRY

PAPER

View Article Online
View Journal | View Issue

Check for updates

Cite this: *Digital Discovery*, 2024, **3**, 786

Predicting small molecules solubility on endpoint devices using deep ensemble neural networks

Mayk Caldas Ramos and Andrew D. White*

Received 3rd November 2023
Accepted 7th March 2024

DOI: 10.1039/d3dd00217a

rsc.li/digitaldiscovery

Aqueous solubility is a valuable yet challenging property to predict. Computing solubility using first-principles methods requires accounting for the competing effects of entropy and enthalpy, resulting in long computations for relatively poor accuracy. Data-driven approaches, such as deep learning, offer improved accuracy and computational efficiency but typically lack uncertainty quantification. Additionally, ease of use remains a concern for any computational technique, resulting in the sustained popularity of group-based contribution methods. In this work, we addressed these problems with a deep learning model with predictive uncertainty that runs on a static website (without a server). This approach moves computing needs onto the website visitor without requiring installation, removing the need to pay for and maintain servers. Our model achieves satisfactory results in solubility prediction. Furthermore, we demonstrate how to create molecular property prediction models that balance uncertainty and ease of use. The code is available at https://github.com/ur-whitelab/mol.dev, and the model is useable at https://mol.dev.

1 Introduction

Aqueous solubility measures the maximum quantity of matter that can be dissolved in a given volume of water. It depends on several conditions, such as temperature, pressure, pH, and the physicochemical properties of the compound being solvated.[1] The solubility of molecules is essential in many chemistry-related fields, including drug development,[2–5] protein design,[6] chemical[7,8] and separation[9] processes. In drug development, for instance, compounds with biological activity may not have enough bioavailability due to inadequate aqueous solubility.

Solubility prediction is essential, driving the development of various methods, from physics-based approaches—including first principles,[10,11] semi-empirical equations,[12–14] molecular dynamics (MD),[15–18] and quantum computations[19]—to empirical methods like quantitative structure–property relationship (QSPR)[20–23] and multiple linear regression (MLR).[24,25] Despite their sophistication, physics-based models often present complexity that limits accessibility to advanced users[26] and do not guarantee higher accuracy than empirical methods.[27] Data-driven models emerge as efficient alternatives, capable of outperforming physics-based models.[26] However, achieving accurate and reliable solubility predictions remains a significant challenge.[26,28]

To address the persistent issues of systematic bias and non-reproducibility in aqueous solubility datasets, Llinàs *et al.*[29,30] introduced two solubility challenges featuring consistent data. The first challenge red participants based on the root mean square error (RMSE) and the accuracy within a $\pm 0.5 \log S$ error range.[31] The second challenge revealed that despite the freedom in method selection, all entries relied on QSPR or machine learning (ML) techniques,[32] yet did not achieve a notable improvement over the first challenge.[31] These challenges highlighted the importance of data quality over model selection for accurate solubility predictions.[32] Sorkun *et al.*[28] further emphasized this by demonstrating how data quality assessments on subsets of the AqSolDB[1] significantly impacted model performance.[28]

McDonagh *et al.*[27] demonstrated that cheminformatic methods surpass first principle theoretical calculations in calculating solubilization free energies, highlighting the superior accuracy of cheminformatics and the efficacy of Random Forest models, evidenced by an RMSE of 0.93 using Llinàs' first dataset.[29] Data-driven approaches, particularly feature-based models, have contributed to accurate aqueous solubility prediction. Delaney[25] used MLR to develop a model called Estimated SOLubility (ESOL) adjusted on a 2874 small organic molecules dataset with an average absolute error (AAE) of 0.83. Comparable performance has been achieved using various methods including MLR,[24] Gaussian processes,[33] undirected graph recurrent neural networks (UG-RNN),[34] deep neural networks (DNN),[35] and random forests (RF).[36,37]

Recently, transformers[38] models have been applied to compute solubility of small molecules.[39–43] Francoeur and Koes[41] developed the SolTranNet, a transformers model trained on AqSolDB[1] solubility data. Notably, this architecture results in an RMSE of only 0.278 when trained and evaluated on the

Chemical Engineer Department, University of Rochester, Rochester, NY 14642, USA. E-mail: andrew.white@rochester.edu

original ESOL[25] dataset using random split. Nevertheless, it shows an RMSE of 2.99 when trained using the AqSolDB[1] and evaluated using ESOL. It suggests that the molecules present in ESOL may have low variability, meaning that samples in the test set are similar to samples in the training set. Hence, models trained on the ESOL training set performed excellently when evaluated on the ESOL test set.

Solubility models should ideally combine accuracy with ease of access. Thus, a common idea is to use web servers to provide easier public access. However, web servers demand continuous financial and time investments for maintenance, leading to the eventual disappearance of some, despite having institutional or government backing.[44] For instance, eight out of 89 web server tools featured in the 2020 *Nucleic Acids Research* special web server issue were offline by the end of 2022.[45] Moreover, computational demands can be significant, with tools like RoseTTAFold[46] and ATB[47] requiring hours to days for job completion, thus creating potential delays due to long queues and wait times.[48]

An alternative approach is to perform the computation directly on the user's device, removing the need for the server's maintenance and cost. This method allows hosting the website as a static file on platforms such as GitHub, with potential archiving on the Internet Archive.† We explored this approach in Ansari and White[49] for bioinformatics. Our web application implements a deep ensemble[50] recurrent neural network (RNN) capable of extracting data directly from molecular string representations, such as SMILES[51] or SELFIES,[52] which can be easily quickly accessed.[53,54]

The primary difficulty lies in the application's dependence on the device's capabilities, which is crucial for smartphones with limited resources. Balancing performance in low-resource settings, the use of transformer models[38] becomes impractical due to their large size, incompatible with smartphone memory and prolonged inference times. Additionally, our model implements a deep ensemble to calibrate uncertainties, making the application of transformers even more unfeasible. In contrast, using descriptors is an easy way to convey physical information to the model and, consequently, enables smaller models. However, descriptor computation is time-intensive. In our tests, using PaDEL to compute descriptors for all molecules in AqSolDB took roughly ∼20 hours. Furthermore, feature-based model development requires specialized knowledge for feature selection,[55] and is limited by the regions of the chemical space these descriptors cover.[56] Even application usage may need specialized data, as Kurotani *et al.*[37] illustrate. RNNs present an alternative for property extraction directly from string representations while allowing for adaptable computational resource management.

In this work, we developed a front-end application using a JavaScript (JS) implementation of TensorFlow framework.[57] Our application can be used to predict the solubility of small molecules with uncertainty. To calibrate the confidence of the prediction, our model implements a deep ensemble approach[50] which allows reporting model uncertainty when reporting the prediction. Our solution implements a deep ensemble of RNN models specially designed to achieve satisfactory performance while being able to run in an environment without strong computational resources. This application runs locally on the user's device and can be accessed at **https://mol.dev/**. Mol.dev does not save data input for predictions in any way.

2 Methods

2.1 Dataset

The data used for training the models were obtained from AqSolDB.[1] This database combined and curated data from 9 different aqueous solubility datasets. The main concern in using a large, curated database is to avoid problems with the generalizability of the model[58] and with the fidelity of the data.[59] AqSolDB consists of aqueous solubility ($\text{Log}\,S$) values for 9982 unique molecules extended with 17 topological and physicochemical 2D descriptors calculated by RDKit.[60]

We augmented AqSolDB to 96 625 molecules using SMILES randomization.[61,62] Each entry of AqSolDB was used to generate at most ten new unique randomized SMILES strings. Training the model on multiple representations of the same molecule improves its ability to learn the chemical space constraints of the training set, as demonstrated in previous studies.[61,62] Duplicates were removed.

After shuffling, the augmented dataset was split into 80%/20% for the training and test datasets, respectively. The curated datasets for the solubility challenges[29,32] were used as withheld validation data to evaluate the model's ability to predict solubility for unseen compounds. To refer to the validation datasets, we labeled the first solubility challenge dataset as "solubility challenge 1" and the two sets from the second solubility challenge as "solubility challenge 2_1" and "solubility challenge 2_2", respectively. Molecules in these three datasets were not found in train and test datasets.

2.2 Model architecture

Our model uses a deep ensemble approach as described by Lakshminarayanan *et al.*[50]. This technique was selected due to its ability to estimate prediction uncertainty, thus enhancing the predictive capability of our model. The uncertainty of a model can be divided into two sources: aleatoric uncertainty (AU) and epistemic uncertainty (EU).[63,64] These uncertainties quantify the intrinsic uncertainty inherent in data observations and the disagreement among model estimations, respectively.[65]

Given a model that outputs two values – $\hat{\mu}_m$ and $\hat{\sigma}_m$ – that characterize a normal distribution $\mathcal{N}(\widehat{\mu}_m, \widehat{\sigma}_m)$, a deep ensemble creates an ensemble of m models that can estimate prediction uncertainty. For a given data point $\vec{x}$, the estimates for the ensemble predictions are computed as follows:

$$\widehat{\mu}(\vec{x}) = \frac{1}{N}\sum_{m}\widehat{\mu}_m(\vec{x}) \tag{1}$$

† **https://archive.org/**

$$\hat{\sigma}_{\text{ale}}{}^{2}(\vec{x}) = \frac{1}{N}\sum_{m}\hat{\sigma}_{m}{}^{2}(\vec{x}), \quad \hat{\sigma}_{\text{epi}}{}^{2}(\vec{x}) = \frac{1}{N}\sum_{m}(\hat{\mu}(\vec{x}) - \hat{\mu}_{m}(\vec{x}))^{2} \tag{2}$$

where $\hat{\sigma}_{\text{ale}}{}^{2}$ is AU, $\hat{\sigma}_{\text{epi}}{}^{2}$ is EU, N is the ensemble size, and m indexes the models in the ensemble.

We used a deep neural network (DNN) implemented using Keras[66] and TensorFlow[67] to build the deep ensemble. Our DNN model uses Self-referencing embedded strings (SELFIES)[52] tokens as input. A pre-defined vocabulary was created by analyzing all training data. Each unique SELFIES group was assigned to a corresponding integer, yielding 273 distinct tokens. Simplified molecular-input line-entry system (SMILES)[51] or SELFIES[52] molecule representations are converted to tokens based on the pre-defined vocabulary. Fig. 1 illustrates the model architecture. The network can be divided into three sections: (i) embedding, (ii) bi-RNN, and (iii) fully connected NN.

The embedding layer converts a list of discrete tokens into a fixed-length vector space. Working on a continuous vector space has two main advantages: it uses a more compact representation, and semantically similar symbols can be described closely in vector space. Our embedding layer has an input dimension of 273 (vocabulary size) and an output dimension of 64.

Following the embedding layer, the data are fed into the bidirectional Recurrent Neural Network (RNN) layer. We used two RNN layers, each containing 64 units. The effects of using Gated Recurrent Unit (GRU) or Long Short-Term Memory (LSTM)[68] layers as the RNN layers were investigated (refer to Section 3.1). Using bi-RNN was motivated based on our previous work[49] in which LSTM helped improve the model's performance for predicting peptide properties using its sequences. More details regarding RNN, LSTM, and GRU layers can be found in ref. 69.

The output from the bi-LSTM stack undergoes normalization *via* Layer Normalization.[70] There is no agreement on why Layer Normalization improves the model's performance.[71–74] The absence of a comprehensive theoretical understanding of normalization effects hinders the evolution of novel regularization schemes.[75] Despite the limited understanding, Layer Normalization is employed due to its demonstrated effectiveness.[74]

After normalization, data is processed through three dense layers containing 32, 16, and 1 units, respectively. The 16-unit layer's output goes to two different 1-unit layers. One layer uses a linear function and the other uses a softplus function, producing $\hat{\mu}m$ and $\hat{\sigma}m$, respectively.

Negative log-likelihood loss l was used to train the model. It is defined as the probability of observing the label y given the input $\vec{x}$:

$$l(\vec{x}, y) = \frac{\log\left(\hat{\sigma}_{m}{}^{2}(\vec{x})\right)}{2} + \frac{(y - \hat{\mu}_{m}(\vec{x}))^{2}}{2\hat{\sigma}_{m}{}^{2}(\vec{x})} \tag{3}$$

During the training phase, dropout layers with 0.35 dropout rate were incorporated after the embedding and each dense layer to mitigate over-fitting.[76] Models were trained using the Adam[77] optimizer with a fixed learning rate of 0.0001 and default values for β_1 and β_2 (0.9 and 0.999, respectively).

Our model employs adversarial training, following the approach proposed by Lakshminarayanan *et al.*[50] to improve the robustness of our model predictions. Because the input for our model is a discrete sequence, we generate adversarial examples by modifying the embedded representation of the input data.

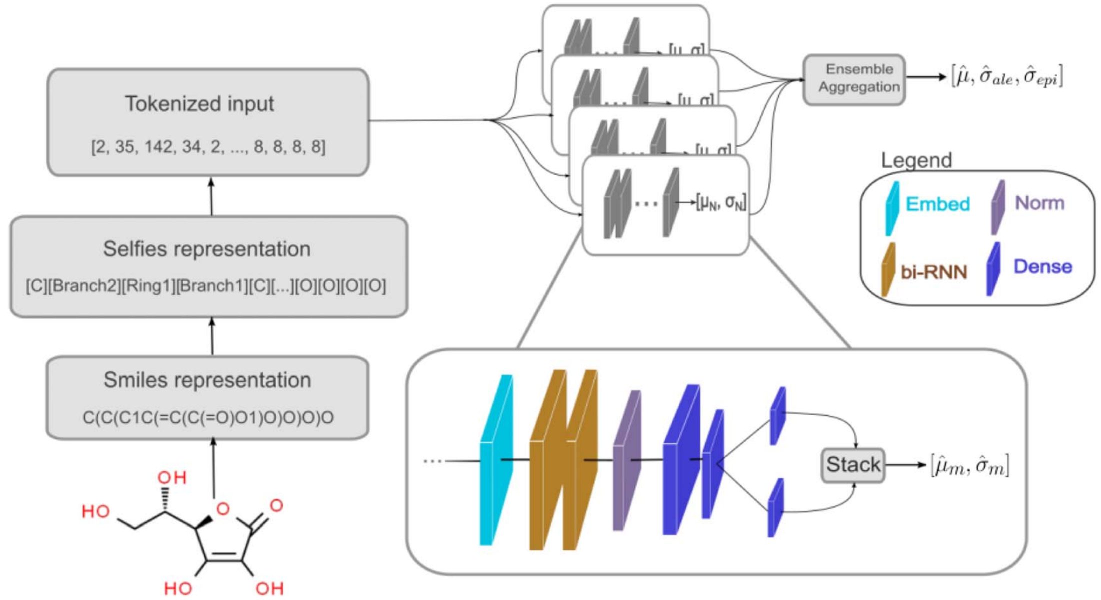

Fig. 1 Scheme of the deep learning DNN. The molecule is input using the SMILES or SELFIES representation. This representation is converted to a tokenized input based on a vocabulary obtained using the training dataset. A set of models represents the deep ensemble model. Each model consists of an embed layer, two bidirectional RNN (bi-RNN) layers, a normalization layer, and three fully connected layers being down-sized in three steps. Dropout layers are present after the embed and after each fully connected layer during training, but they were not represented in this scheme. Predictions of the models in the ensemble are then aggregated.

Each iteration in the training phase consists of first computing the loss using eqn (3) and a second step with a new input $\vec{x}'$ to smooth the model's prediction:

$$\vec{x}' = \vec{x} + \varepsilon \mathrm{sign}(\nabla_x l(\vec{x}, y)) \tag{4}$$

where ε is the strength of the adversarial perturbation.

Details of the model performance, limitations, training data, ethical considerations, and caveats are available as a model card[78] at **http://mol.dev/**.

3 Results

In order to evaluate the performance of our model using deep ensembles, two baseline models were created: (i) an XGBoost Random Forest (RF) model using the 17 descriptors available on AqSolDB plus 1809 molecular descriptors calculated by PaDELPy, a python wrapper for the PaDEL-descriptor[79] software, and (ii) a model with the same architecture used on our deep ensemble using RMSE as the loss function and no ensemble (referred to as DNN). RFs are the SOTA of solubility prediction. We used this baseline as a comparison to prove that our model is able to achieve SOTA performance using only molecular string representations. In addition, we evaluate the effects of (i) the bi-RNN layer (either GRU or LSTM), (ii) using an augmented dataset to train, (iii) the adversarial training, and (iv) the ensemble size in the model's performance. Table 1 shows the performance of each one of our trained models.

3.1 Gated layer

The most common RNN layers are the GRU and the LSTM. GRU layers use two gates, reset and update, to control the cell's internal state. On the other hand, LSTM layers use three gates: forget, input, and output, with the same objective. Available studies compare GRU and LSTM performances in RNNs for different applications, for instance: forecasting,[80] cryptocurrency,[81,82] wind speed,[83,84] condition of a paper press,[85] motive classification in thematic apperception tests[86] and music and raw speech.[87] Nevertheless, it is not clear which of those layers would perform better at a given task.

We trained models with four elements in the deep ensemble using GRU or LSTM. Metrics can be found in Table 1; for an explanation of the naming syntax used in this work, refer to Table 1 caption. Using LSTM resulted in a decrease in RMSE and MAE and an increase in the correlation coefficient, indicating better performance. For solubility challenges 1, 2_1, and 2_2, the $\text{kde4}^{\text{GRU}}_{\text{Aug}}$ model yielded RMSE values of 1.329, 1.354, and 1.626, respectively, while the $\text{kde4}^{\text{LSTM}}_{\text{Aug}}$ model achieved 1.273, 1.137, and 1.511, respectively. This trend was also observed for the models trained without data augmentation (see Table 1). Considering that LSTM performs better regarding this model and data, we will consider only bi-LSTM layers for further discussion. Those results are in accordance with our previous work[49] in which using LSTM helped improve the model's performance.

3.2 Data augmentation

Our model is not intrinsically invariant with respect to the SELFIES representation input. For instance, both "C(C(C1C(=C(C(=O)O1)O)O)O)O" and "O=C1OC(C(O)CO)C(O)=C1O" are valid SMILES representations for the ascorbic acid (see Fig. 1) that will be encoded for different SELFIES tokens. Hence, the model should learn to be invariant concerning changes in the string representation during training. It can be achieved by augmenting the dataset with SMILES randomization and training the model using different representations with the same label. Therefore, the model can learn relations in the chemical space instead of correlating the label with a specific representation.[61] With this aim, we evaluated the effects of

Table 1 Summary of the metrics for each trained model. We used the Root Mean Squared Error (RMSE(↓)), Mean Absolute Error (MAE(↓)), and Pearson correlation coefficient (*r*(↑)) to evaluate our models. The arrows indicate the direction of improvement. Deep ensemble models are referred to as "kde*N*", where *N* is the ensemble size. Baseline models using random forest (RF) and the DNN model employed for deep ensemble (DNN) are also displayed. DNN model was trained as described in section 2. The models in which data augmentation was used were subscribed with the flag Aug. A superscript indicates if the bidirectional layer implements a GRU or a LSTM layer. In addition, models trained not using adversarial perturbation are flagged with "-NoAdv". The columns show the results of each model evaluated on each solubility challenge dataset. 2_1 represents the tight dataset (set-1), while 2_2 represents the loose dataset (set-2) as described in the original paper (see ref. 30). *r* stands for the Pearson correlation coefficient. The best-performing model in each dataset is displayed in bold

	Solubility challenge 1			Solubility challenge 2_1			Solubility challenge 2_2		
Model	RMSE	MAE	*r*	RMSE	MAE	*r*	RMSE	MAE	*r*
RF	1.121	0.914	0.547	**0.950**	**0.727**	**0.725**	**1.205**	**1.002**	**0.840**
DNN	1.540	1.214	0.433	1.315	1.035	0.651	1.879	1.381	0.736
DNN_{Aug}	1.261	1.007	0.453	1.371	1.085	0.453	2.189	1.710	0.386
kde4^{GRU}	1.610	1.145	0.462	1.413	1.114	0.604	1.488	1.220	0.704
$\text{kde4}^{\text{LSTM}}$	1.554	1.191	0.507	1.469	1.188	0.650	1.523	1.161	0.706
kde4^{GRU}-NoAdv	1.729	1.348	0.525	1.483	1.235	0.622	1.954	1.599	0.517
$\text{kde4}^{\text{LSTM}}$-NoAdv	1.425	1.114	0.505	1.258	0.972	0.610	1.719	1.439	0.609
$\text{kde4}^{\text{GRU}}_{\text{Aug}}$	1.329	1.148	0.426	1.354	1.157	0.674	1.626	1.340	0.623
$\text{kde4}^{\text{LSTM}}_{\text{Aug}}$	1.273	0.984	0.473	1.137	0.932	0.639	1.511	1.128	0.717
$\text{kde8}^{\text{LSTM}}_{\text{Aug}}$	1.247	0.984	0.542	1.044	0.846	0.701	1.418	1.118	0.729
kde10AugLSTM-NoAdv	1.689	1.437	0.471	1.451	1.238	0.676	1.599	1.405	0.699
$\text{kde10}^{\text{LSTM}}_{\text{Aug}}$	**1.095**	**0.843**	**0.559**	0.983	0.793	0.724	1.263	1.051	0.792

augmenting the dataset by generating new randomized SMILES representations for each sample.

Augmenting the dataset had a significant impact on the metrics. It could be seen improvements of ∼0.5 in the RMSE when evaluating on challenge datasets 1 and 2_1, and a gain of ∼0.2 on 2_2 (see Table 1). Concerning the first two datasets, augmenting data improved every model used in this study. However, surprisingly, data augmentation led to a deprecation of the DNN model on the solubility challenge 2_2 dataset. This behavior was not further investigated.

3.3 Adversarial training

Using adversarial training improved performance in Lakshminarayanan *et al.*[50] studies. Hence, they suggested that it should be used in future applications of their deep learning algorithm. Thus, we tested the effects of adversarial perturbation on training models with ensemble sizes of 4 and 10.

Comparing kde4$^{\text{LSTM}}$-NoAdv and kde4$^{\text{LSTM}}$, using adversarial training decreases model performance. It can be seen in Table 1 that using adversarial perturbation increased the RMSE from 1.425 to 1.554 and 1.258 to 1.469 in solubility challenges dataset 1 and 2_1, respectively. However, the RMSE decreased from 1.719 to 1.523 in dataset 2_2. Using adversarial perturbation affected our kde4$^{\text{LSTM}}$'s performance by a change in RMSE of ±0.2.

The inconsistent performance improvement observed when using adversarial training was further investigated with models in which the dataset was augmented. Due to the lack of multiple string representations in the training dataset, it is known that kde4$^{\text{LSTM}}$ may have generalization problems. A generalization issue could direct the adversarial perturbation in a non-physical direction because the model does not have complete knowledge about the chemical representation space. This hypothesis is reinforced when we compare kde10$^{\text{LSTM}}_{\text{Aug}}$-NoAdv and kde10$^{\text{LSTM}}_{\text{Aug}}$. When using adversarial training on a model trained with an augmented dataset, the performance improvement is more evident (∼0.5) and consistent for all the test datasets.

3.4 Deep ensemble size

To investigate the effects of increasing the ensemble size, we trained models with an ensemble of 4, 8, and 10 models. Given the previous results, these models used LSTM as the bi-RNN layer and were trained on the augmented dataset. Specifically for the solubility challenge 2_2, the most complex set to predict, these models presented an RMSE of 1.511, 1.418, and 1.263, respectively. Therefore, increasing the ensemble size consistently improved performance. We also observed this improvement on the other datasets (see Table 1).

Besides the immediate improvement in RMSE, increasing the ensemble size also improves the uncertainty of the model. Fig. 2 shows the density distribution of the aleatoric variance and the epistemic variance (respectively related to AU and EU) for kde4$^{\text{LSTM}}_{\text{Aug}}$ (top 6 panels) and kde10$^{\text{LSTM}}_{\text{Aug}}$ (bottom six panels).

The increase in ensemble size led to a decrease in both uncertainties. AU distributions for the kde4$^{\text{LSTM}}_{\text{Aug}}$ are centered around 4 log S^2, displaying a long tail that extends to values as high as 20 log S^2 in the worst case (solubility challenge 2_2). A similar trend is observed in EU distributions. On the other hand, the kde10$^{\text{LSTM}}_{\text{Aug}}$ model results in narrower distributions. The mean of these distributions remains relatively unchanged, but a noticeable reduction in the extent of their tails can be observed. AU distribution ends in values around 10 log S^2.

4 Discussion

After extensively investigating the hyperparameter selection, we compared our model with available state-of-art models from the literature. Performance metrics on the solubility challenge datasets can be found in Table 2. Parity plots for our chosen models are presented in Fig. 3.

Comparing the performance of different models is a complex task, as performance metrics cannot be directly compared across models evaluated on distinct datasets. To address this issue, Panapitiya *et al.*[89] curated a large and diverse dataset to train models with various architectures and molecular representations. They also compared the performance of these models on datasets from the literature.[24,25,29,30,88,92–97] Although their models achieved an RMSE of ∼1.1 on their test set, using descriptors as molecular representations resulted in RMSE values ranging from 0.55 to ∼1.35 when applied to other datasets from the literature. According to their study, the solubility challenge datasets by Llinàs *et al.*[29,30] were found to be particularly challenging due to their more significant reproducibility error. Therefore, we focused on the Llinàs datasets to compare our performance with the literature.

Focusing on the solubility challenge 1 dataset,[29] kde10$^{\text{LSTM}}_{\text{Aug}}$ is only ∼0.2 RMSE units worse than the best model available in the literature.[34] The RMSE of the participants of the challenge was not reported.[31] The primary metric used to evaluate models was the percentage of predictions within an error of 0.5 Log S units (called ±0.5 log%). Computing the same metric, kde10$^{\text{LSTM}}_{\text{Aug}}$ has a percentage of correct prediction of 44.4%. This result would place our model among the 35% best participants. The participant with the best performance presented a ±0.5 log% of 60.7%.

The architecture of the models was not published in the findings of the first challenge.[31] Nevertheless, the findings for the second challenge[32] investigated the participants more thoroughly. Participants were asked to identify their models' architecture and descriptors used. The challenge is divided into two datasets. Set-1 contains Log S values with an average interlaboratory reproducibility of 0.17 Log S. Our kde10$^{\text{LSTM}}_{\text{Aug}}$ achieve an RMSE of 0.983 and a ±0.5 log% of 40.0% in this dataset. Therefore, our model performs better than 62% of the published RMSE values and 50% of the ±0.5 log%. In addition, the model with the best performance is an artificial neural network (ANN) that correctly predicted 61% (±0.5 log%) of the molecule's Log S using a combination of molecule descriptors and fingerprints. The second dataset (set-2) contains molecules whose solubility measurements are more challenging, reporting an average error in reproducibility of 0.62 Log S. The kde10$^{\text{LSTM}}_{\text{Aug}}$ achieves an RMSE of 1.263 and a ±0.5 log% of 23.3%. It performs better than 82% of the candidates when considering the RMSE. Surprisingly, ±0.5 log% does not follow this outstanding performance, which is more significant than only 32% regarding the literature, kde10$^{\text{LSTM}}_{\text{Aug}}$ has an RMSE only ∼0.1

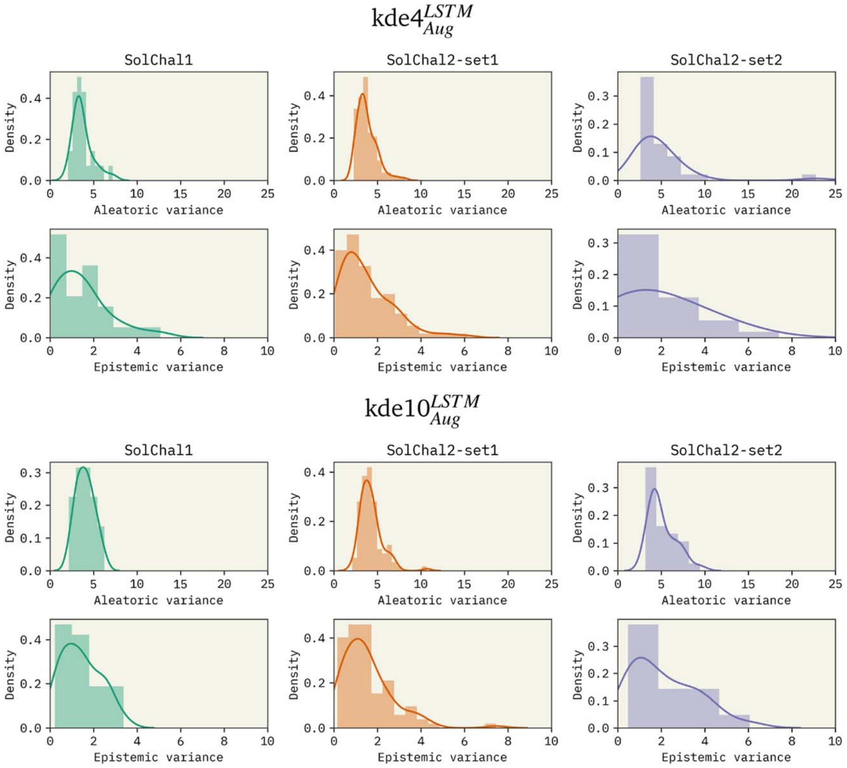

Fig. 2 Density distribution of the aleatoric (AU) and epistemic variances (EU) for the: (i) kde4$^{\mathrm{LSTM}}_{\mathrm{Aug}}$ (top six panels) and (ii) kde10$^{\mathrm{LSTM}}_{\mathrm{Aug}}$ (bottom six panels). Increasing ensemble size reduces the extent of the distribution's tail, decreasing uncertainty about predictions. However, the ensemble size does not noticeably affect the distribution center.

Table 2 Metrics for the best models found in the current study (upper section) and for other state-of-art models available in the literature (lower section). Values were taken from the cited references. Missing values stand for entries that the cited authors did not study. SolChal columns stand for the solubility challenges. 2_1 represents the tight dataset (set-1), while 2_2 represents the loose dataset (set-2) as described in the original paper (see ref. 30). The best-performing model in each dataset has its RMSE value in bold

	Solubility challenge 1			Solubility challenge 2_1			Solubility challenge 2_2		
Model	RMSE	MAE	*r*	RMSE	MAE	*r*	RMSE	MAE	*r*
RF	1.121	0.914	**0.950**	**0.727**	**1.205**	**1.002**			
DNN	1.540	1.214	1.315	1.035	1.879	1.381			
DNN$_{\mathrm{Aug}}$	1.261	1.007	1.371	1.085	2.189	1.710			
kde4$^{\mathrm{LSTM}}_{\mathrm{Aug}}$	1.273	0.984	1.137	0.932	1.511	1.128	1.397	1.131	
kde8$^{\mathrm{LSTM}}_{\mathrm{Aug}}$	1.247	0.984	1.044	0.846	1.418	1.118	1.676	1.339	
kde10$^{\mathrm{LSTM}}_{\mathrm{Aug}}$	**1.095**	**0.843**	0.983	0.793	1.263	1.051	**1.316**	**1.089**	
Linear regression[25]							0.75		
UG-RNN[34]	0.90	0.74							
RF w/CDF descriptors[27]	0.93								
RF w/Morgan fingerprints[36]		**0.64**							
Consensus[88]	**0.91**								
GNN[89]	∼1.10		**0.91**		**1.17**				
SolvBert[90]	0.925								
[a]SolTranNet[41]			1.004		1.295		2.99		
[b]SMILES-BERT[91]							0.47		
[b]MolBERT[40]							0.531		
[b]RT[42]							0.73		
[b]MolFormer[43]							**0.278**		

[a] Has overlap between training and test sets. [b] Pre-trained model was fine-tuned on ESOL.

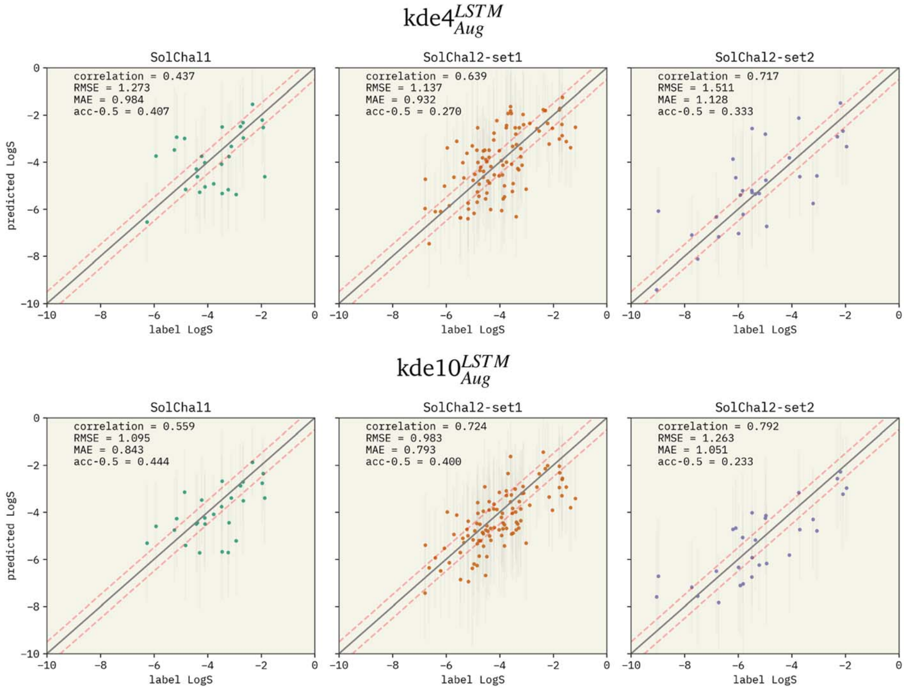

Fig. 3 Parity plots for two selected models being evaluated on the solubility challenge datasets: (i) kde4$^{\text{LSTM}}_{\text{Aug}}$ (top row), and (ii) kde10$^{\text{LSTM}}_{\text{Aug}}$ (bottom row). The left, middle, and right columns show the parity plots for solubility challenge 1,[29] 2-set1, and 2-set2,[30] respectively. Pearson correlation coefficient is displayed together with RMSE and MAE. "acc-0.5" stands for the ±0.5 log% metric. Red dashed lines show the limits for molecules considered a correct prediction when computing the ±0.5 log%. The correlation between predicted values and labels increases when more models are added to the ensemble. RMSE and MAE also follow this pattern. However, the ±0.5 log% decreases in set-2 of the second solubility challenge dataset (SolChal2-set2). While kde10$^{\text{LSTM}}_{\text{Aug}}$ improved the prediction of molecules that were being poorly predicted by kde4$^{\text{LSTM}}_{\text{Aug}}$, the prediction of molecules with smaller errors was not greatly improved.

higher than a GNN that used an extensive set of numeric and one-hot descriptors in their feature vector.[89] Our model performs better than a transformer model that uses SMILES-string and an adjacency matrix and inputs.[41] The performance of those models is available in Table 2.

Notably, all participants in the solubility challenge 2 submitted a kind of QSPR or descriptor-based ML model. Using descriptors provides an easy way to ensure model invariance concerning molecule representation and is more informative since they can be physical quantities. However, selecting appropriate descriptors is crucial for developing descriptor-based ML models. It often requires specialists with a strong intuition about the relevant physical and chemical properties for predicting the target quantity. Feature-based models are still being considered to be the SOTA of solubility prediction. Recently, studies investigating different descriptors and fingerprints were performed.[36,98] These studies showed that similarly to the impacts of data quality,[28] molecular representation also has a great impact on models' performance. Despite Tayyebi *et al.*[36] being able to achieve an MAE of 0.64 on solubility challenge 1 when using Morgan fingerprints (MF), Zagidullin *et al.*[98] reported poor performance when using MF. Our approach, on the other hand, is based on extracting information from simple string representations, a more straightforward raw data. Furthermore, we could achieve state-of-the-art performance while balancing the model size and complexity and using a raw input (a simple string). This simplified usage enables running the model on devices with limited computing power.

Lastly, transformer models have been used to address the issue of accurately predicting the solubility of small compounds. The typical workflow for transformers involves pre-training the model using a large dataset and subsequently fine-tuning it for a specific downstream task using a smaller dataset. Most existing models were either pre-trained on the ESOL[25] dataset or pre-trained on a larger dataset and fine-tuned using ESOL. Hence, the generalizability of those models cannot be verified. In a study by Francoeur and Koes,[41] they considered two versions of their model, SolTranNet. The first version of SolTranNet was trained with the ESOL dataset using random splits. This approach achieved an RMSE of 0.278. Subsequently, the deployed version of SolTranNet was trained with the AqSolDB.[1] When ESOL was used to evaluate their deployed version, the model presented an RMSE of 2.99. While our model achieved an RMSE of 1.316 on ESOL, outperforming the SolTranNet deployed version, it cannot be compared with other models trained on ESOL.

5 Conclusions

We used the JavaScript implementation of TensorFlow (tensorflowJS) to implement a deep ensemble recurrent neural network (RNN) that can accurately predict Log S values directly from SMILES or SELFIES string representations. This model is hosted on a static website and can be accessed at **https://mol.dev/**. The contributions of this work can be listed as follows: (1) we show that it is possible to use string representations to predict solubilities; (2) we show that using strings does not lead to an unacceptable decrease in performance, with models performing comparable to state-of-the-art (SOTA) models on Llinas *et al.* datasets; (3) our model is able to perform predictions with uncertainties, increasing the reliability and practical utility of the predictions; (4) we largely improve model ease of use by implementing a static website whose does not require domain-specific data or knowledge to be used.

Our based on a deep ensemble of recurrent neural networks (RNNs) model was trained using SMILES randomization for data augmentation on the AqSolDB dataset and validated using the solubility challenges by Llínas *et al.*[29,30] It directly processes molecular string representations, such as SMILES or SELFIES, to predict solubility without relying on pre-selected descriptors. This approach not only simplifies the prediction process but also enhances its applicability across a broader chemical space. In addition, we show that this deep ensemble RNN model could achieve similar performance compared to a random forest (RF) using PaDEL descriptors. RFs with descriptors were shown to perform relatively well in other datasets.

By carefully compromising between performance and complexity, we developed a model with acceptable performance and that is not computationally intensive. It enables us to host the model on a static website using TensorFlow JS. Our model was designed to operate on devices with limited computational resources, aiming to broaden the accessibility of advanced solubility prediction tools. This application can satisfactorily run on any device with limited computational resources, such as laptops and smartphones. This approach ensures wider applicability, catering to the needs of users without access to high-performance computing facilities, improving usability and flexibility, and decreasing implementation costs. We believe this is a considerable step in improving the usability of deep learning models and promoting such models to a broader scientific community.

Data availability

The code for training and evaluating the models discussed in this manuscript are available at: **https://github.com/ur-whitelab/mol.dev/tree/main/ml**. The code employed for this study was the code deployed in February 17. This study used publicly available data from AqSolDB dataset (**https://doi.org/10.1038/s41597-019-0151-1**), ESOL dataset (**https://doi.org/10.1021/ci034243x**), and the first (**https://doi.org/10.1021/ci800436c**) and second solubility challenges (**https://doi.org/10.1021/acs.jcim.0c00701**). These datasets are compiled and available at: **https://github.com/ur-whitelab/mol.dev/blob/main/ml/data.zip**.

Author contributions

M. C. R. implemented the deep learning model, performed the training and hyperparameters optimization, tested the model's performance, analyzed the results, and wrote this manuscript. A. D. W. idealized the project, proposed the model to be used, implemented the deep learning approach, and developed **http://mol.dev/**.

Conflicts of interest

There are no conflicts to declare.

Acknowledgements

The authors acknowledge the National Institute of General Medical Sciences of the National Institutes of Health (NIH) under award number R35GM137966. This research used the computational resources and structure provided by the Center for Integrated Research Computing (CIRC) at the University of Rochester.

Notes and references

1 M. C. Sorkun, A. Khetan and S. Er, *Sci. Data*, 2019, **6**, 143.

2 F. Dajas, *J. Ethnopharmacol.*, 2012, **143**, 383–396.

3 L. Di, P. V. Fish and T. Mano, *Drug Discovery Today*, 2012, **17**, 486–495.

4 R. Docherty, K. Pencheva and Y. A. Abramov, *J. Pharm. Pharmacol.*, 2015, **67**, 847–856.

5 J. A. Barrett, W. Yang, S. M. Skolnik, L. M. Belliveau and K. M. Patros, *Drug Discovery Today*, 2022, **27**, 1315–1325.

6 P. Sormanni, F. A. Aprile and M. Vendruscolo, *J. Mol. Biol.*, 2015, **427**, 478–490.

7 J. M. Herrero-Martínez, M. Sanmartin, M. Rosés, E. Bosch and C. Ràfols, *Electrophoresis*, 2005, **26**, 1886–1895.

8 L. J. Diorazio, D. R. J. Hose and N. K. Adlington, *Org. Process Res. Dev.*, 2016, **20**, 760–773.

9 E. Sheikholeslamzadeh and S. Rohani, *Ind. Eng. Chem. Res.*, 2012, **51**, 464–473.

10 S. H. Yalkowsky and S. C. Valvani, *J. Pharm. Sci.*, 1980, **69**, 912–922.

11 Y. Ran and S. H. Yalkowsky, *J. Chem. Inf. Comput. Sci.*, 2001, **41**, 354–357.

12 A. Fredenslund, R. L. Jones and J. M. Prausnitz, *AIChE J.*, 1975, **21**(6), 1086–1099.

13 D. S. Abrams and J. M. Prausnitz, *AIChE J.*, 1975, **21**, 116–128.

14 G. Maurer and J. M. Prausnitz, *Fluid Phase Equilib.*, 1978, **2**, 91–99.

15 K. Lüder, L. Lindfors, J. Westergren, S. Nordholm and R. Kjellander, *J. Phys. Chem.*, 2007, **111**(25), 7303–7311.

16 K. Lüder, L. Lindfors, J. Westergren, S. Nordholm and R. Kjellander, *J. Phys. Chem. B*, 2007, **111**(7), 1883–1892.

17 S. Boothroyd, A. Kerridge, A. Broo, D. Buttar and J. Anwar, *Phys. Chem. Chem. Phys.*, 2018, **20**, 20981–20987.

18 S. Boothroyd and J. Anwar, *J. Chem. Phys.*, 2019, **151**, 184113.

19 J. Tomasi, B. Mennucci and R. Cammi, *Chem. Rev.*, 2005, **105**, 2999–3093.

20 X. Yu, X. Wang, H. Wang, X. Li and J. Gao, *QSAR Comb. Sci.*, 2006, **25**, 156–161.

21 J. Ghasemi and S. Saaidpour, *Chem. Pharm. Bull.*, 2007, **55**(4), 669–674.

22 P. R. Duchowicz and E. A. Castro, *Int. J. Mol. Sci.*, 2009, **10**, 2558–2577.

23 B. Louis, J. Singh, B. Shaik, V. K. Agrawal and P. V. Khadikar, *Chem. Biol. Drug Des.*, 2009, **74**(2), 190–195.

24 J. Huuskonen, *J. Chem. Inf. Comput. Sci.*, 2000, **40**, 773–777.

25 J. S. Delaney, *J. Chem. Inf. Comput. Sci.*, 2004, **44**, 1000–1005.

26 R. E. Skyner, J. L. McDonagh, C. R. Groom, T. van Mourik and J. B. O. Mitchell, *Phys. Chem. Chem. Phys.*, 2015, **17**, 6174–6191.

27 J. L. McDonagh, N. Nath, L. De Ferrari, T. van Mourik and J. B. O. Mitchell, *J. Chem. Inf. Model.*, 2014, **54**, 844–856.

28 M. C. Sorkun, J. M. V. A. Koelman and S. Er, *iScience*, 2021, **24**, 101961.

29 A. Llinàs, R. C. Glen and J. M. Goodman, *J. Chem. Inf. Model.*, 2008, **48**, 1289–1303.

30 A. Llinas and A. Avdeef, *J. Chem. Inf. Model.*, 2019, **59**, 3036–3040.

31 A. J. Hopfinger, E. X. Esposito, A. Llinàs, R. C. Glen and J. M. Goodman, *J. Chem. Inf. Model.*, 2009, **49**, 1–5.

32 A. Llinas, I. Oprisiu and A. Avdeef, *J. Chem. Inf. Model.*, 2020, **60**, 4791–4803.

33 A. Schwaighofer, T. Schroeter, S. Mika, J. Laub, A. ter Laak, D. Sülzle, U. Ganzer, N. Heinrich and K.-R. Müller, *J. Chem. Inf. Model.*, 2007, **47**, 407–424.

34 A. Lusci, G. Pollastri and P. Baldi, *J. Chem. Inf. Model.*, 2013, **53**, 1563–1575.

35 Z. Ye and D. Ouyang, *J. Cheminf.*, 2021, **13**, 98.

36 A. Tayyebi, A. S. Alshami, Z. Rabiei, X. Yu, N. Ismail, M. J. Talukder and J. Power, *J. Cheminf.*, 2023, **15**, 99.

37 A. Kurotani, T. Kakiuchi and J. Kikuchi, *ACS Omega*, 2021, **6**, 14278–14287.

38 A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, *arXiv*, 2017, preprint, arXiv:1706.03762v7, DOI: 10.48550/arXiv.1706.03762.

39 S. Wang, Y. Guo, Y. Wang, H. Sun and J. Huang, *Proceedings of the 10th ACM*, 2019.

40 B. Fabian, T. Edlich, H. Gaspar and M. Segler and Others, *arXiv*, 2020, preprint, arXiv:2011.13230v1, DOI: 10.48550/arXiv.2011.13230.

41 P. G. Francoeur and D. R. Koes, *J. Chem. Inf. Model.*, 2021, **61**, 2530–2536.

42 J. Born and M. Manica, *arXiv*, 2022, preprint, arXiv:2202.01338v3, DOI: 10.48550/arXiv.2202.01338.

43 J. Ross, B. Belgodere, V. Chenthamarakshan, I. Padhi, Y. Mroueh and P. Das, *Res. Sq.*, 2022, DOI: 10.21203/rs.3.rs-1570270/v1.

44 B. Zdrazil and R. Guha, *J. Med. Chem.*, 2017, **61**, 4688–4703.

45 D. Seelow, *Nucleic Acids Res.*, 2020, **48**, W1–W4.

46 M. Baek, F. DiMaio, I. Anishchenko, J. Dauparas, S. Ovchinnikov, G. R. Lee, J. Wang, Q. Cong, L. N. Kinch, R. D. Schaeffer, C. Millán, H. Park, C. Adams, C. R. Glassman, A. DeGiovanni, J. H. Pereira, A. V. Rodrigues, A. A. van Dijk, A. C. Ebrecht, D. J. Opperman, T. Sagmeister, C. Buhlheller, T. Pavkov-Keller, M. K. Rathinaswamy, U. Dalwadi, C. K. Yip, J. E. Burke, K. C. Garcia, N. V. Grishin, P. D. Adams, R. J. Read and D. Baker, *Science*, 2021, **373**, 871–876.

47 M. Stroet, B. Caron, K. M. Visscher, D. P. Geerke, A. K. Malde and A. E. Mark, *J. Chem. Theory Comput.*, 2018, **14**, 5834–5845.

48 D. G. A. Smith, D. Altarawy, L. A. Burns, M. Welborn, L. N. Naden, L. Ward, S. Ellis, B. P. Pritchard and T. D. Crawford, *Wiley Interdiscip. Rev.: Comput. Mol. Sci.*, 2021, **11**(2), DOI: 10.1002/wcms.1491.

49 M. Ansari and A. D. White, *J. Chem. Inf. Model.*, 2023, **63**, 2546–2553.

50 B. Lakshminarayanan, A. Pritzel and C. Blundell, *arXiv*, 2016, preprint, arXiv:1612.01474v3, DOI: 10.48550/arXiv.1612.01474.

51 D. Weininger, *J. Chem. Inf. Model.*, 1988, **28**, 31–36.

52 M. Krenn, Q. Ai, S. Barthel, N. Carson, A. Frei, N. C. Frey, P. Friederich, T. Gaudin, A. A. Gayle, K. M. Jablonka, R. F. Lameiro, D. Lemm, A. Lo, S. M. Moosavi, J. M. Nápoles-Duarte, A. Nigam, R. Pollice, K. Rajan, U. Schatzschneider, P. Schwaller, M. Skreta, B. Smit, F. Strieth-Kalthoff, C. Sun, G. Tom, G. Falk von Rudorff, A. Wang, A. D. White, A. Young, R. Yu and A. Aspuru-Guzik, *Patterns*, 2022, **3**, 100588.

53 S. Kim, P. A. Thiessen, T. Cheng, B. Yu and E. E. Bolton, *Nucleic Acids Res.*, 2018, **46**, W563–W570.

54 O. T. Schilter, T. Laino and P. Schwaller, *Appl. AI Lett.*, 2024, **5**(1), DOI: 10.1002/ail2.91.

55 J. A. Beltran, L. Aguilera-Mendoza and C. A. Brizuela, *BMC Genomics*, 2018, **19**, 672.

56 G. M. Maggiora, On outliers and activity cliffs why QSAR often disappoints, *J. Chem. Inf. Model.*, 2006, **46**(4), 1535.

57 D. Smilkov, N. Thorat, Y. Assogba, C. Nicholson, N. Kreeger, P. Yu, S. Cai, E. Nielsen, D. Soegel, S. Bileschi and Others, *Proc. Mach. Learn.*, 2019, **1**, 309–321.

58 J. Wang, T. Hou and X. Xu, *J. Chem. Inf. Model.*, 2009, **49**, 571–581.

59 J. Wang and T. Hou, *Comb. Chem. High Throughput Screening*, 2011, **14**, 328–338.

60 Landrum, *RELease 1.0*.

61 J. Arús-Pous, S. V. Johansson, O. Prykhodko, E. J. Bjerrum, C. Tyrchan, J.-L. Reymond, H. Chen and O. Engkvist, *J. Cheminf.*, 2019, **11**, 71.

62 P. Schwaller, A. C. Vaucher, T. Laino and J.-L. Reymond, *ChemRxiv*, 2020, preprint, DOI: 10.26434/chemrxiv.13286741.v1.

63 M. H. Shaker and E. Hüllermeier, *Advances in Intelligent Data Analysis XVIII*, 2020, pp. 444–456.

64 B. Ghoshal, A. Tucker, B. Sanghera and W. Lup Wong, *Comput. Intell.*, 2021, **37**, 701–734.

65 G. Scalia, C. A. Grambow, B. Pernici, Y.-P. Li and W. H. Green, *J. Chem. Inf. Model.*, 2020, **60**, 2697–2717.

66 F. Chollet and Others, *Keras: The Python Deep Learning library*, 2018.

67 M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin and Others, *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015.

68 S. Hochreiter and J. Schmidhuber, *Neural Comput.*, 1997, **9**, 1735–1780.

69 A. Zhang, Z. C. Lipton, M. Li and A. J. Smola, *Dive into Deep Learning*, Cambridge University Press, 2023.

70 J. L. Ba, J. R. Kiros and G. E. Hinton, *arXiv*, 2016, preprint, arXiv:1607.06450v1, DOI: **10.48550/arXiv.1607.06450**.

71 S. Ioffe and C. Szegedy, *Proceedings of the 32nd International Conference on Machine Learning*, Lille, France, 2015, pp. 448–456.

72 M. Awais, M. T. B. Iqbal and S.-H. Bae, *Revisiting Internal Covariate Shift for Batch Normalization*, 2021.

73 S. Santurkar, D. Tsipras, A. Ilyas and A. Madry, *Advances in Neural Information Processing Systems*, 2018.

74 J. Xu, X. Sun, Z. Zhang, G. Zhao and J. Lin, *arXiv*, 2019, preprint, arXiv:1911.07013v1, DOI: **10.48550/arXiv.1911.07013**.

75 Y. Tian and Y. Zhang, *Inf. Fusion*, 2022, **80**, 146–166.

76 Y. Gal and Z. Ghahramani, *Proceedings of The 33rd International Conference on Machine Learning*, New York, USA, 2016, pp. 1050–1059.

77 D. P. Kingma and J. Ba, *arXiv*, 2014, preprint, arXiv:1412.6980v9, DOI: **10.48550/arXiv.1412.6980**.

78 M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I. D. Raji and T. Gebru, *Proceedings of the Conference on Fairness, Accountability, and Transparency*, New York, NY, USA, 2019, pp. 220–229.

79 C. W. Yap, PaDEL-descriptor: an open source software to calculate molecular descriptors and fingerprints, *J. Comput. Chem.*, 2011, **32**(7), 1466–1474.

80 S. Gao, Y. Huang, S. Zhang, J. Han, G. Wang, M. Zhang and Q. Lin, *J. Hydrol.*, 2020, **589**, 125188.

81 J. Kim, S. Kim, H. Wimmer and H. Liu, *2021 IEEE/ACIS 6th International Conference on Big Data, Cloud Computing, and Data Science*, BCD, 2021, pp. 37–44.

82 A.-A. Encean and D. Zinca, *Cryptocurrency Price Prediction Using LSTM and GRU Networks*, 2022.

83 V. B. Kumar, V. Bharat Kumar, V. Mallikarjuna Nookesh, B. Satya Saketh, S. Syama and J. Ramprabhakar, *Wind Speed Prediction Using Deep Learning-LSTM and GRU*, 2021.

84 X. Liu, Z. Lin and Z. Feng, *Energy*, 2021, **227**, 120492.

85 B. C. Mateus, M. Mendes, J. T. Farinha, R. Assis and A. M. Cardoso, *Energies*, 2021, **14**, 6958.

86 N. Gruber and A. Jockisch, *Front. Artif. Intell.*, 2020, **3**, 40.

87 J. Chung, C. Gulcehre, K. Cho and Y. Bengio, *arXiv*, 2014, preprint, arXiv:1412.3555, DOI: **10.48550/arXiv.1412.3555**.

88 S. Boobier, A. Osbourn and J. B. O. Mitchell, *J. Cheminf.*, 2017, **9**, 63.

89 G. Panapitiya, M. Girard, A. Hollas, J. Sepulveda, V. Murugesan, W. Wang and E. Saldanha, *ACS Omega*, 2022, **7**, 15695–15710.

90 J. Yu, C. Zhang, Y. Cheng, Y.-F. Yang, Y.-B. She, F. Liu, W. Su and A. Su, *Digital Discovery*, 2023, **2**, 409–421.

91 H. Kim, J. Lee, S. Ahn and J. R. Lee, *Sci. Rep.*, 2021, **11**, 11028.

92 G. Klopman and H. Zhu, *J. Chem. Inf. Comput. Sci.*, 2001, **41**, 439–445.

93 T. J. Hou, K. Xia, W. Zhang and X. J. Xu, *J. Chem. Inf. Comput. Sci.*, 2004, **44**, 266–275.

94 J. Wang, G. Krudy, T. Hou, W. Zhang, G. Holland and X. Xu, *J. Chem. Inf. Model.*, 2007, **47**, 1395–1404.

95 S. Boobier, D. R. J. Hose, A. J. Blacker and B. N. Nguyen, *Nat. Commun.*, 2020, **11**, 5753.

96 B. Tang, S. T. Kramer, M. Fang, Y. Qiu, Z. Wu and D. Xu, *J. Cheminf.*, 2020, **12**, 15.

97 Q. Cui, S. Lu, B. Ni, X. Zeng, Y. Tan, Y. D. Chen and H. Zhao, *Front. Oncol.*, 2020, **10**, 121.

98 B. Zagidullin, Z. Wang, Y. Guan, E. Pitkänen and J. Tang, *Briefings Bioinf.*, 2021, **22**(6), DOI: **10.1093/bib/bbab291**.