Evaluation of Deep Learning Architecture for Aqueous Solubility Prediction

Gihan Panapitiya,* Michael Girard, Aaron Hollas, Jonathan Sepulveda, Vijayakumar Murugesan, Wei Wang, and Emily Saldanha*

ABSTRACT:

Determining the aqueous solubility of molecules using the largest currently available solubility data set, we implement deep learning-based models to predict solubility from the molecular structure and explore several different molecular representations including molecular descriptors, simplified molecular-input line-entry system strings, molecular graphs, and three dimensional atomic coordinates using four different neural network architecture fully connected neural networks, recurrent neural networks, graph neural networks (GNNs), and SchNet.

We find that models using molecular descriptors achieve the best performance, with GNN models also achieving good performance. We perform extensive error analysis to understand the molecular properties that influence model performance.

Data:

1. Download data from https://figshare.com/s/542fb80e65742746603c and save it as data.csv in the ./data folder
2. Generate Pybel coordinates and MDM features by running create_data.py at the ./data folder
3. To train MDM, GNN, SMI and SCH models run train.py at ./mdm, ./gnn, ./smi and ./sch folders respectively.
4. To make predictions use predict.ipynb files at each model folders.

Best models used to obtain the results in the paper are included in each model folders (mdm_paper.h5, gnn_paper.pt, smi_paper.h5, sch_paper.pt)

The final data set consists of 17,149 molecules with sizes ranging from 1 to 273 atoms and with molecular masses ranging from 16 to 1819.The measured aqueous solubilities of these molecules range from 3.4 × 10−18 to 45.5 mol/L.

Data preprocessing is a vast yet vital step while building a model, this preprocessing has various steps, First downloading the data set from the link above. And save it in the '/content/data' folder. Now run the create data while uploading the 'fragments' which would be used for the schnet dataset.

This code ‘Create data’ takes about 3 hrs to run, giving us the data set for graphnet, smiles, mdm and schnet.

- The dataset for GNN is saved in the ‘/content/data/train.pkl.gz’,‘/content/data/test.pkl.gz’, ‘/content/data/val.pkl.gz’.
- The dataset for MDM is saved in the ‘/content/data/train’,‘/content/data/test’, ‘/content/data/val’.
- The dataset for SCHNet is saved in the ‘/content/pybel’.
- The dataset for MDM is saved in the ‘/content/data/train’,‘/content/data/test’, ‘/content/data/val’.

Procedure:

The code for the study has been uploaded in the github, now we need to run them accordingly to get the results.

1) Run the pip install module, there are some libraries which require installs, which would take about 20 mins to run.
2) Then the Data module, which takes more time about 3 hrs to run, and creates all the data for the models.
3) Now we run our models. First GNN, Going through the code and training the model.
4) Now we save the model and predict using the saved model.
5) This step does for all the models.
6) Then we run the MDM module, then SCHnet, and Smiles.

The results have been shown below, the r2 scores and the parity plots.

Results:

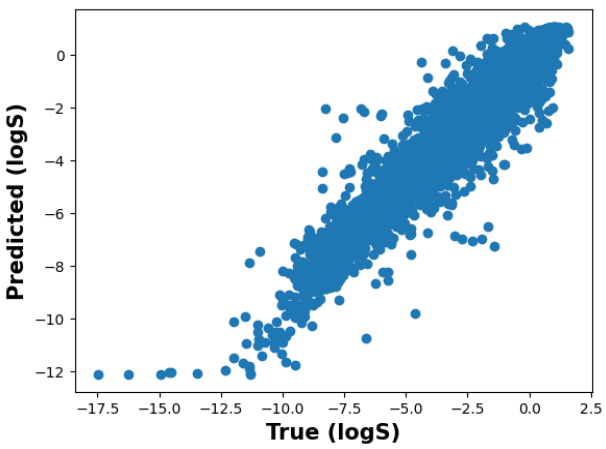
Smiles

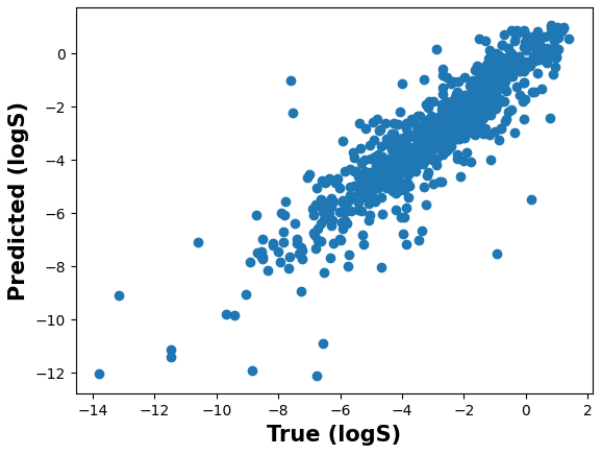
The model architecture is shown below. The training had 70 epochs and stopped due to early stopping. And ran for 10 mins.

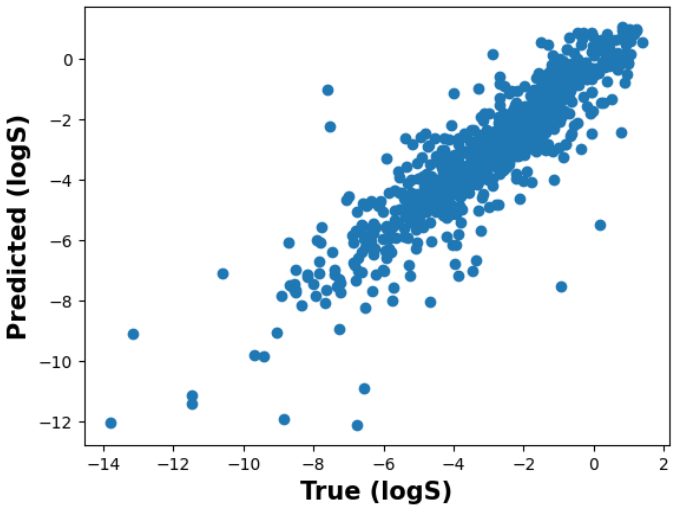
```
model = Sequential()
model.add(Embedding(max_features, int(args['ed']), input_length=maxlen))
model.add(LSTM( int(args['ls1']), return_sequences=True))
model.add(Dropout( args['d1'] ) )
model.add(Bidirectional(LSTM( int(args['ls1']) )))
model.add(Dropout( args['d2'] ))
model.add(Dense(int( args['h1'] )))
model.add(Activation( act[args['a1']] ))

model.add(Dense( int(args['h3']) ))
model.add(Activation( act[args['a3']] ))
model.add(Dropout( args['d3'] ))


model.add(Dense(1, activation = 'linear'))
adam = keras.optimizers.Adam( lr = config.lr )
```

Train results
r2: 0.9136
sp: 0.9506
rmse: 0.6468
mae: 0.4581

Val results
r2: 0.7943
sp: 0.8962
rmse: 1.0104
mae: 0.6812







Test results
r2: 0.7539
sp: 0.8680
rmse: 1.0883
mae: 0.7044

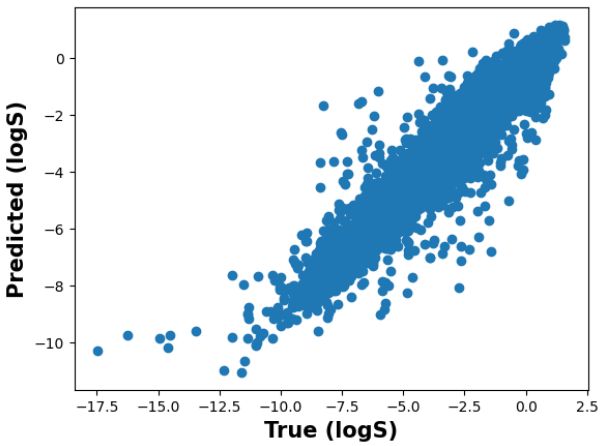
MDM

The model architecture is shown below. The training had 55 epochs and stopped due to early stopping. And ran for 1 mins.

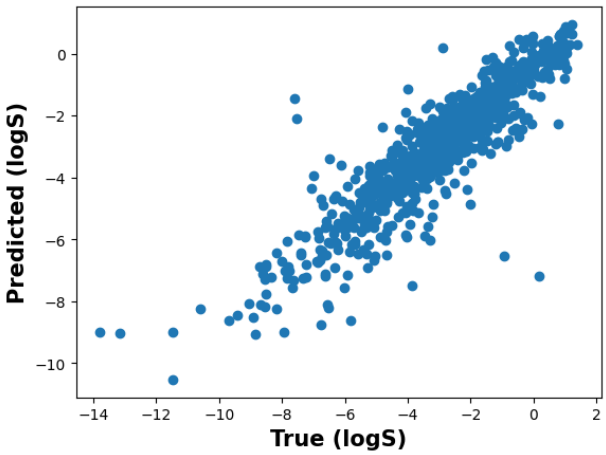
We used RDKit to identify molecular fragments attached to benzene-like structures in our data set. From the resulting fragments, we selected the 52 most common fragments in addition to seven other functional groups commonly found in chemical compounds. Combining the 2D descriptors, 3D descriptors, and fragment counts, there are 839 molecular descriptors used as features.

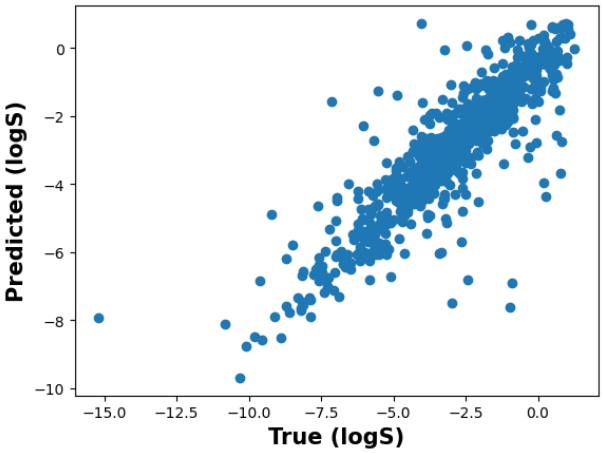
```
model = Sequential()
model.add(Dense(int(args['h1']), input_shape = (input_shape,) ))
model.add(Activation( act[args['a1']] ))
model.add(Dropout( args['d1'] ))
model.add(Dense(  int(args['h2'])  ))
model.add(Activation( act[args['a2']] ))
model.add(Dropout( args['d2'] ))

model.add(Dense(1, activation = 'linear'))
rmsprop = keras.optimizers.RMSprop( lr = config.lr )
opt = rmsprop

model.compile(loss='mean_squared_error', optimizer=opt)
```

train results
r2: 0.9011
sp: 0.9486
Rmse: 0.6918
mae: 0.4784



val results
r2: 0.8186
sp: 0.9058
rmse: 0.9487
mae: 0.6396



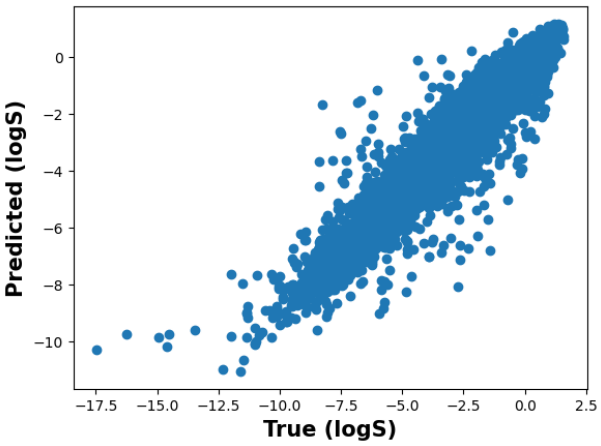


test results
r2: 0.7792
sp: 0.8852
rmse: 1.0309
mae: 0.6573

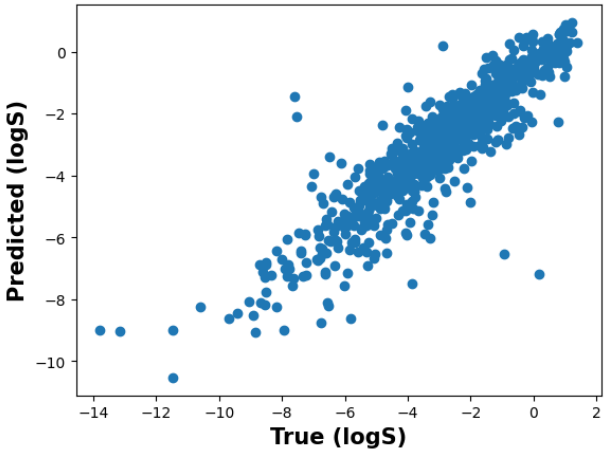
GNN

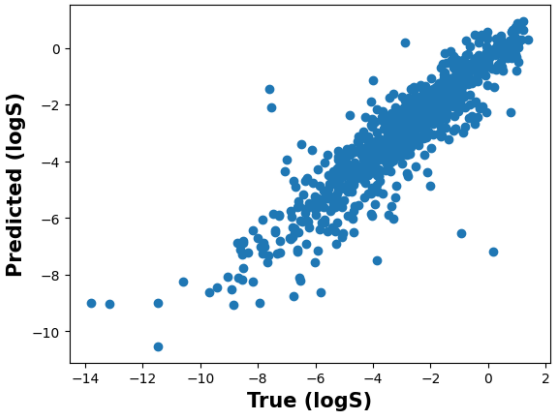
The model architecture is shown in the code. The training had 40 epochs and stopped due to early stopping. And ran for 66 mins.

Train results
r2: 0.7852
Rmse: 0.9233
sp: 1.0062
mae: 0.6532



Valid results
r2: 0.7408
rmse: 0.8813
sp: 0.9218
mae: 0.6633



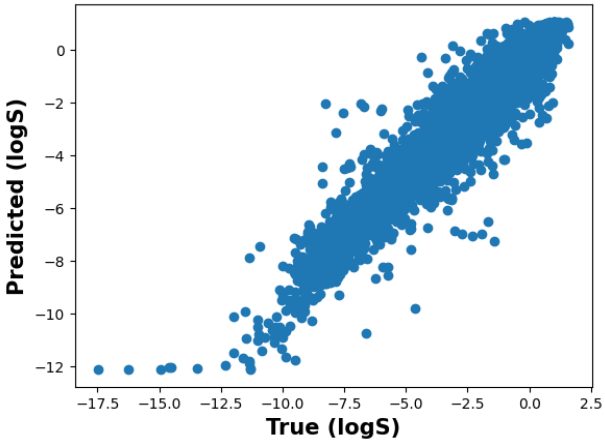


test results
r2: 0.7792
sp: 0.8652
rmse: 1.0729
mae: 0.7273

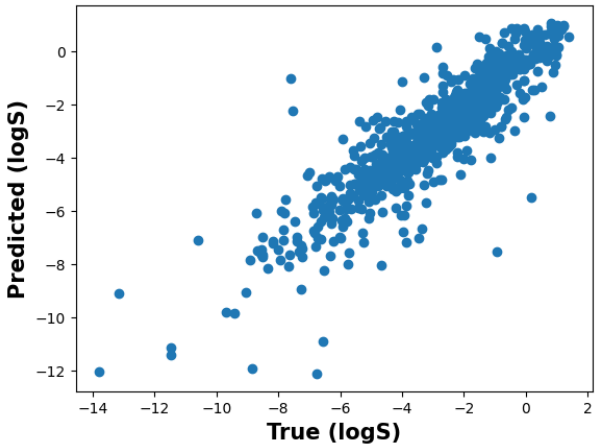
SCHNet:

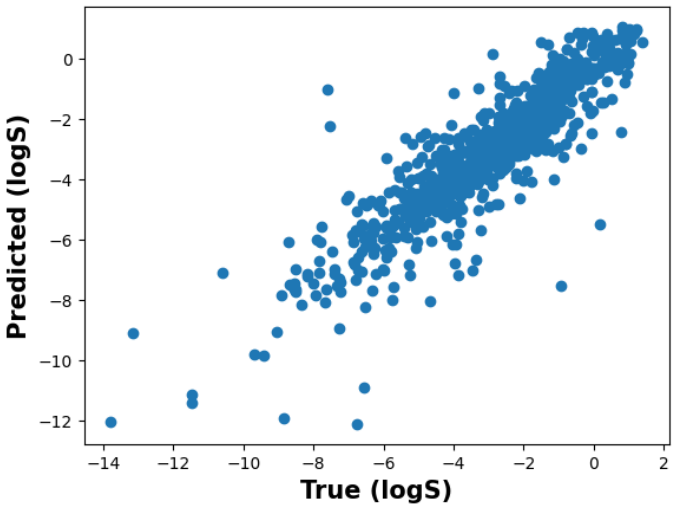
The model architecture is shown in the code, it includes schnetpack. The training had 50 epochs and stopped due to early stopping. And ran for 10 mins.

Train results
r2: 0.8636
sp: 0.9506
rmse: 1.0468
mae: 0.8581



Val results
r2: 0.8943
sp: 0.8962
rmse: 1.0104
mae: 0.8812





Test results
R2: 0.6882
sp: 0.8380
rmse: 1.2283
mae: 0.8844

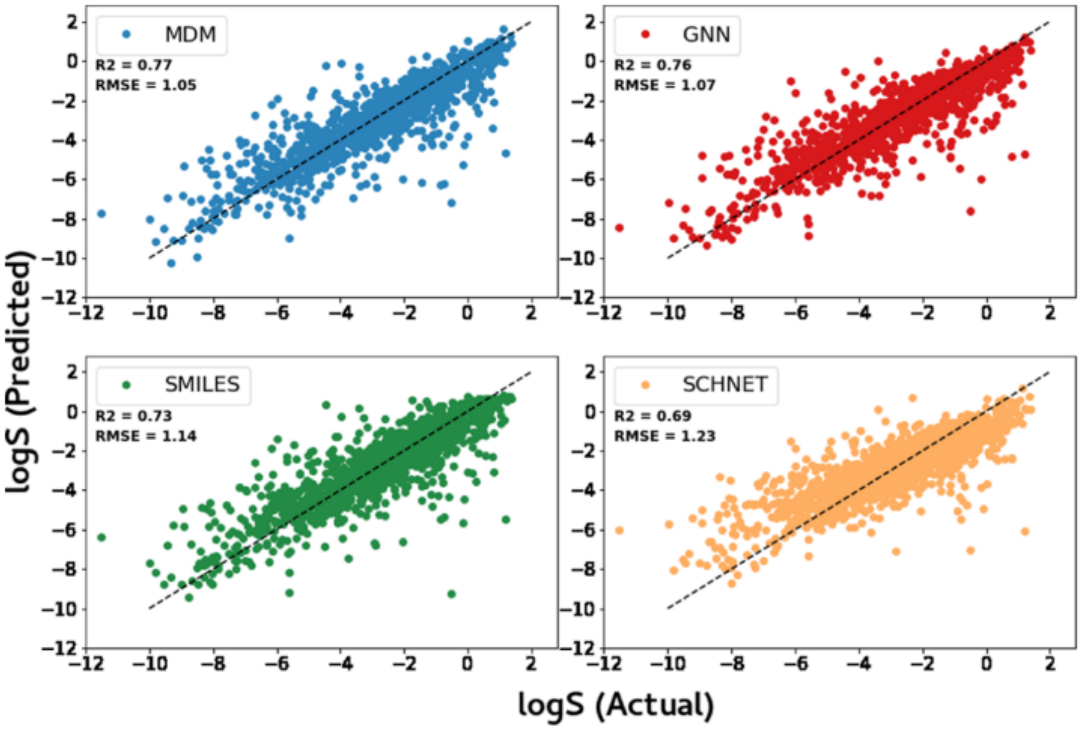
From The Research Paper:

Exactly from as show above regarding the time complexity, The average time (in seconds) required to generate input representations for a single molecule for the MDM, SMILES, GNN, and SchNet models are 0.52, 0.0004, 0.0029, and 0.03,

Table 2. Evaluation Results for the Four Models on the Test Set

model	R^2	Spearman	RMSE (log S)	MAE (log S)
MDM	**0.7719**	**0.8787**	**1.0513**	**0.6887**
GNN	0.7628	0.8708	1.0722	0.7256
SMILES	0.7337	0.8603	1.1360	0.7609
SCHNET	0.6883	0.8337	1.2291	0.8892

We see that it closely resembles the one from the research paper.



Conclusion:

- Overall, we found the best performing approach leveraged a set of derived molecular features.
- Of the three models that rely on raw molecular structure information, we find the GNN model achieves the highest performance.
- This error analysis leads to several key findings. Models with differing data representations and architectures make highly correlated errors, showing that they are learning similar structure− property relationships.