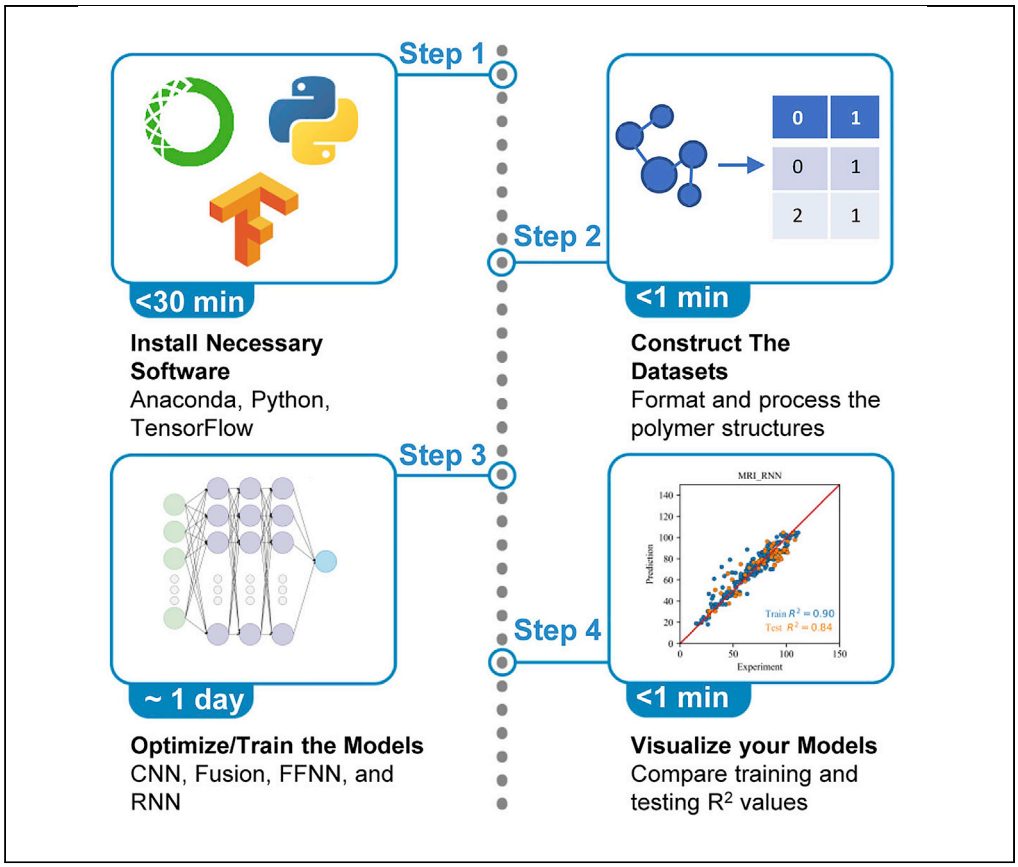
STAR Protocols

CellPress
OPEN ACCESS

Protocol

Unified machine learning protocol for copolymer structure-property predictions



Structure-property relationships are extremely valuable when predicting the properties of polymers. This protocol demonstrates a step-by-step approach, based on multiple machine learning (ML) architectures, which is capable of processing copolymer types such as alternating, random, block, and gradient copolymers. We detail steps for necessary software installation and construction of datasets. We further describe training and optimization steps for four neural network models and subsequent model visualization and comparison using training and test values.

Publisher's note: Undertaking any experimental protocol requires adherence to local institutional guidelines for laboratory safety and ethics.

Lei Tao, Tom Arbaugh, John Byrnes, Vikas Varshney, Ying Li

yli2562@wisc.edu

Highlights

Detailed steps for building machine learning model for copolymers

Consideration of both chemical composition and sequence distribution of copolymers

Analysis of different copolymers types using four machine learning models

Differentiation of sequence patterns of random, block, and gradient copolymers

Tao et al., STAR Protocols 3, 101875
December 16, 2022 © 2022 The Author(s).
https://doi.org/10.1016/j.xpro.2022.101875

STAR Protocols

CellPress
OPEN ACCESS

Protocol

Unified machine learning protocol for copolymer structure-property predictions

Lei Tao,[1,6,7] Tom Arbaugh,[2,6] John Byrnes,[3] Vikas Varshney,[4] and Ying Li[1,5,8,*]

[1]Department of Mechanical Engineering, University of Connecticut, Storrs, CT 06269, USA

[2]Department of Physics, Wesleyan University, Middletown, CT 06459, USA

[3]SRI International, San Diego, CA 92131, USA

[4]Materials and Manufacturing Directorate, Air Force Research Laboratory, Wright-Patterson Air Force Base, Dayton, OH 45433, USA

[5]Department of Mechanical Engineering, University of Wisconsin-Madison, Madison, WI 53706-1572, USA

[6]These authors contributed equally

[7]Technical contact: lei.tao@uconn.edu

[8]Lead contact

*Correspondence: yli2562@wisc.edu
https://doi.org/10.1016/j.xpro.2022.101875

SUMMARY

Structure-property relationships are extremely valuable when predicting the properties of polymers. This protocol demonstrates a step-by-step approach, based on multiple machine learning (ML) architectures, which is capable of processing copolymer types such as alternating, random, block, and gradient copolymers. We detail steps for necessary software installation and construction of datasets. We further describe training and optimization steps for four neural network models and subsequent model visualization and comparison using training and test values.

For complete details on the use and execution of this protocol, please refer to Tao et al. (2022).[1]

BEFORE YOU BEGIN

This section includes the introduction of the fundamentals of machine learning model involved, the minimal hardware requirements, and the installation procedures.

Regarding the machine learning model inputs, first the simplified molecular-input line-entry system (SMILES) is used to represent each monomer of each copolymer. Then, a feature vector is obtained for the repeat unit of each polymer using the Morgan fingerprint with radius 3.[2] The number of occurrences of each substructure found in the monomer is labeled with a real number in the feature vector.[3,4] This feature vector represents the composition information of copolymers. Stacking the feature vectors of the different monomers in a specific order represents the sequence distribution of a copolymer. The feature vector or the resultant feature matrix serves as the numerical input that is readable by ML models.

Based on different datasets, the target properties of copolymers investigated in this study included optoelectronic properties,[5] ^{19}F nuclear magnetic resonance (NMR) signal-to-noise ratio (SNR),[6] and glass transition temperature T_g.[7,8] The protocol here demonstrates the application of the ML model for ^{19}F NMR SNR predictions. The prediction of other properties can be processed in a similar way. While there is no limitation of the properties that can be processed by the machine learning models, the performance can be highly problem-dependent. In general it is recommended to check a target property in two aspects: a) if the targeted property relates to the polymer's monomer-level

This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

fingerprints that are used in this study; and b) if there is enough data available to develop the model. The fidelity of the proposed model towards a new property will require a thorough investigation of comparing different models via validation/verification on unseen test data. When the new property is somewhat similar to these cases, these machine learning models could still be used through a transfer learning strategy. In addition, the size of the dataset needed for reliable machine learning is also problem-dependent. Some properties may require a large dataset while others may not.

In terms of the machine learning algorithm, four ML models are applied to the copolymer dataset to correlate the molecular numerical input to the property output, including a feed-forward neural network (FFNN) model, a convolutional neural network (CNN) model, a recurrent neural network (RNN) model, and a combined FFNN/RNN (Fusion) model. For FFNN, the feature vector of a copolymer F_{AB} is calculated as the molar-weighted summation of each monomer's feature vector: $F_{AB} = F_A m_A + F_B m_B$. F_A and F_B are feature vectors for monomers A and B, respectively. m_A and m_B are molar-weighted ratios of monomers A and B, respectively. This representation considers the copolymer's composition information, but ignores the information of copolymers' sequence distribution. For the other three ML models, by stacking the feature vectors of two monomers into a feature matrix, the sequence distribution of copolymers is considered explicitly.

Repository download and environment installation

Timing: <30 min

1. Use a computer hardware that supports the installation of anaconda python 3.7.

   Note: The computer used in this protocol is a DELL Precision 3650 Tower Workstation with the following hardware: 11th Generation Intel Core i7-11700 (16 MB Cache, 8 Core, 2.5 GHz–4.9 GHz), 32 GB (2 × 16 GB) DDR4 UDIMM non-ECC Memory, and Nvidia RTX A4000 (16 GB, 4 DP), 256 GB PCIe NVMe Class 40 M.2 SSD. This specification is recommended but not required. GPU is not necessary but recommended. The download and installations times may vary based on specific computing resources.

2. Clone the GitHub repository which includes the training scripts and datasets.
3. Install Anaconda3 and the necessary packages.
   a. Find anaconda installation instructions here.
   b. Activate anaconda, navigate to the directory of the repository and create an environment with the required packages using:

```
> conda create -name copolymer -file requirements.txt -channel conda-forge
```

   c. Check the new environment using:

```
> conda env list
```

   d. Activate the environment using:

```
> conda activate copolymer
```

   e. Check whether all packages in requirements.txt are installed.

```
> conda list
```

If not, please use the following command to install the missing package.

```
> conda install package-name or > pip install package-name
```

f. Add conda environment into Jupyter notebook.

```
> python -m ipykernel install -user -name (environment name) -display-name (environment name)
```

g. Start Jupyter Notebook using:

```
> jupyter notebook
```

h. Set the kernel to use the created environment to run the Jupiter Notebook file.

⚠ CRITICAL: RDKit is only supported with python versions <3.8.

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
Dataset 2	GitHub	https://github.com/figotj/Copolymer
Software and algorithms		
Anaconda 3	Anaconda Inc.	https://www.anaconda.com/products/distribution
Python version 3.7	Python Software Foundation	https://www.python.org
Tensorflow 2.3.0	Open-source software	https://tensorflow.org
RDKit	Open-source software	https://rdkit.org
Model Codes	GitHub	https://github.com/figotj/Copolymer

STEP-BY-STEP METHOD DETAILS

All original code can be found at https://github.com/figotj/Copolymer. There are four .py files to be executed to build and train the ML models. To showcase the workflow of the model, a Jupyter Notebook file demonstrating the main steps of the model building can be found at https://github.com/figotj/Copolymer/blob/main/Protocol_Copolymer_19F%20MRI.ipynb.

Import dependencies

Timing: <1 min

1. Import the necessary python packages as well as setting TensorFlow to use available GPU hardware.

Note: The dependencies contained within these cells are included in the requirements.txt file contained in the repository and are necessary to train the following models.

```
>import pandas as pd

>import numpy as np

>import pickle

>from rdkit import Chem
```

```
>from rdkit.Chem import AllChem
>from rdkit.Chem import Descriptors
>from rdkit.Chem import rdMolDescriptors
>from rdkit.Chem.Draw import IPythonConsole
>from rdkit.Chem import Draw
>from rdkit.Chem.Draw import rdMolDraw2D
>from keras.layers import Input, Dense
>from keras.models import Model
>from keras.utils import plot_model
>from keras.layers.merge import Concatenate
>from tensorflow.keras.models import Sequential, save_model,
>    load_model
>from tensorflow.keras.layers import Dense, Flatten, LSTM,
>    Embedding, Bidirectional, TimeDistributed, Reshape
>from tensorflow.keras.optimizers import Adam
>from tensorflow import keras
>from tensorflow.keras.layers import Conv2D, MaxPooling2D,
>    Conv1D, MaxPooling1D, Dropout
>import tensorflow as tf
>from sklearn.model_selection import train_test_split
>from sklearn.metrics import r2_score, mean_squared_error,
>    mean_absolute_error
>import seaborn as sns
>from sklearn.model_selection import train_test_split
>import matplotlib.pyplot as plt
>import random
>import time
```

Construct datasets

Timing: <1 min

2. Load, visualize, and pre-process the dataset from the original publication.

Note: The dataset contains the Nuclear Magnetic Resonance Signal-to-Noise Ratio (NMR SNR) which will serve as the target property for training and predictions.

```
> DF_MRI = pd.read_excel(open('./datasets/Dataset 2.xlsx', 'rb'), sheet_name='Data organized fluoro-monomer')
```

	TFEA	HexaFOEA	NonaFOEA	PEGA	HEA	MSEA	19F NMR Signal-to-Noise Ratioa	Weight % Fluorine	Molecular weight (Mn)b	Dispersity (Đ)b	Smiles
0	0.0	0.0	0.1	0.9	0.0	0.0	20	0.036740	-	-	C.O(C(=O)C(C[*])[*])CCOC(C(F)(F)F)(C(F)(F)F)C(...
1	0.0	0.0	0.2	0.3	0.0	0.5	47	0.117168	7100	1.25	C.O(C(=O)C(C[*])[*])CCOC(C(F)(F)F)(C(F)(F)F)C(...
2	0.0	0.0	0.2	0.3	0.5	0.0	61	0.127174	8100	1.16	C.O(C(=O)C(C[*])[*])CCOC(C(F)(F)F)(C(F)(F)F)C(...
3	0.0	0.0	0.2	0.4	0.0	0.4	46	0.105659	7800	1.26	C.O(C(=O)C(C[*])[*])CCOC(C(F)(F)F)(C(F)(F)F)C(...
4	0.0	0.0	0.2	0.4	0.4	0.0	51	0.112017	-	-	C.O(C(=O)C(C[*])[*])CCOC(C(F)(F)F)(C(F)(F)F)C(...
...	...	...	...	...	...	...	...	...	...	...	...
408	0.6	0.0	0.0	0.3	0.1	0.0	50	0.137867	4600	1.37	C.C(OC(=O)C(C[*])[*])C(F)(F)F.O(C(=O)C(C[*])[*...
409	0.6	0.0	0.0	0.4	0.0	0.0	31	0.120230	-	-	C.C(OC(=O)C(C[*])[*])C(F)(F)F.O(C(=O)C(C[*])[*...
410	0.6	0.0	0.0	0.4	0.0	0.0	47	0.120230	-	-	C.C(OC(=O)C(C[*])[*])C(F)(F)F.O(C(=O)C(C[*])[*...
411	0.6	0.0	0.0	0.4	0.0	0.0	52	0.120230	-	-	C.C(OC(=O)C(C[*])[*])C(F)(F)F.O(C(=O)C(C[*])[*...
414	0.7	0.0	0.0	0.3	0.0	0.0	55	0.158419	-	-	C.C(OC(=O)C(C[*])[*])C(F)(F)F.O(C(=O)C(C[*])[*...

271 rows × 11 columns

Figure 1. Copolymer datasets containing the SMILES and molar ration of every monomer in each copolymer

3. Construct a table containing the SMILES and molar ratio of every monomer in each copolymer.

Note: Figure 1 shows the summarized table for the copolymer dataset of 271 copolymer data points. The chemical structure of each monomer is shown in Figure 2.

Train feed-forward neural network (FFNN)

Timing: <1 min

4. Prepare the input for the training of the FFNN model.
   a. Calculate the feature vector of a copolymer F_{AB} as the molar-weighted summation of each monomer's feature vector: $F_{AB} = F_A m_A + F_B m_B$.

Note: There are 271 copolymers. The feature vector for one monomer has a length of 80, and a molar-weighted summation of two feature vectors also has a length of 80. Therefore, the size of the input matrix is 271 × 80, as shown in Figure 3.

5. Build FFNN model using the TensorFlow package and train the model on the copolymer dataset.
   a. Employ Keras RandomSearch to optimize hyperparameters such as the number of layers, the number of neurons in each layer, and the learning rate.
   b. Add 2 hidden layers with 24 neurons in the first layer and 64 neurons in the second layer. Use the 'ReLU' activation function for all neurons. The model architecture is illustrated in Figure 4.
   c. Train the model for 100 epochs with batch sizes of 128.

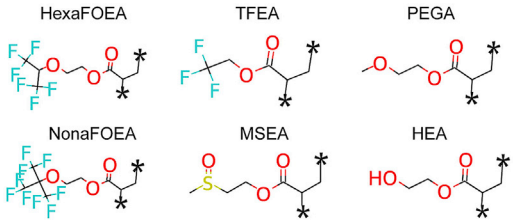


Figure 2. The chemical structure of each monomer using a SMILES to molecules function

d. Output the training R^2, MAE, and RMSE as well as the test R^2, MAE, and RMSE after the training is completed.

```
>model = keras.models.Sequential()
>model.add(Dense(units = 24, input_dim =
>    x_train.shape[1],activation='relu'))
>model.add(Dense(units = 64, activation='relu'))
>model.add(Dense(units = 1))
>model.compile(optimizer=keras.optimizers.Adam(
>    learning_rate=0.001), loss="mean_squared_error",
>    metrics=["mean_squared_error"])
>Model = model.fit(x_train, y_train, epochs = 1000,
>    batch_size = 128, validation_data = (x_test, y_test),
>    verbose=2)
```

6. Visualize results using matplotlib.

Note: The results visualization compares the predicted values from the FFNN model against the ground truth. A parity plot is the outcome of the model training/validation.

Train convolutional neural network (CNN)

Timing: <5 min

7. Prepare the input for the training of the CNN model.
   a. Stack feature vectors into a feature matrix based on the sequential distribution of the copolymer .
   b. Reshape the size of the feature matrix to be suitable for the CNN architecture.

Note: There are 271 copolymers. The feature vector for one monomer has a length of 80, and 100 feature vectors are stacked to represent a copolymer. The number of each monomer is set in the same proportion as their composition in the copolymer, e.g., stacking 55 F_A and 45 F_B if the molar ration of the copolymer A:B is 55:45. Finally the size of the input matrix is 271 × 100 × 80, as shown in Figure 5.

8. Build CNN model using the TensorFlow package and train the model on the copolymer dataset.
   a. Employ Keras RandomSearch to optimize hyperparameters such as the number of layers, the number of neurons in each layer, and the learning rate.

```
Mix_X.shape
```

```
(271, 80)
```

```
Mix_X
```

```
array([[1. , 0. , 0. , ..., 0. , 0.9, 0. ],
       [1. , 0.5, 0. , ..., 0. , 0.3, 0. ],
       [1. , 0. , 0. , ..., 0. , 0.3, 0. ],
       ...,
       [1. , 0. , 0. , ..., 0.6, 0.4, 0. ],
       [1. , 0. , 0. , ..., 0.6, 0.4, 0. ],
       [1. , 0. , 0. , ..., 0.7, 0.3, 0. ]])
```

Figure 3. The numerical input of copolymers for the CNN model

b. Add 3 2D Convolutional layers with 8 filters in each layer. Use the 'ReLU' activation functions and change kernel sizes. The model architecture is illustrated in Figure 6.
c. Apply a dropout rate of 0.3 during the training process which involves 200 epochs.
d. Output the training R^2, MAE, and RMSE as well as the test R^2, MAE, and RMSE after the training is completed.

```
>model = Sequential()
>model.add(Conv2D(8, (10, 10), activation='relu',
>    input_shape=X_train.shape[1:]))
>model.add(Conv2D(8, (4, 4), activation='relu'))
>model.add(Conv2D(8, (3, 3), activation='relu'))
>model.add(MaxPooling2D(pool_size=(2, 2)))
>model.add(Dropout(0.3))
>model.add(Flatten())
>model.add(Dense(1))
>optimizer=keras.optimizers.Adam(lr=0.005)
>model.compile(optimizer=optimizer,
>    loss='mean_absolute_error')
>Model=model.fit(x=X_train, y=y_train, epochs=200,
>    batch_size=64, validation_split=0.2)
```

9. Visualize results using matplotlib.

Note: The results visualization compares the predicted values from the CNN model against the ground truth. A parity plot is the outcome of the model training/validation.

Train recurrent neural network (RNN)

Timing: <2 min

10. Prepare the input for the training of the RNN model.

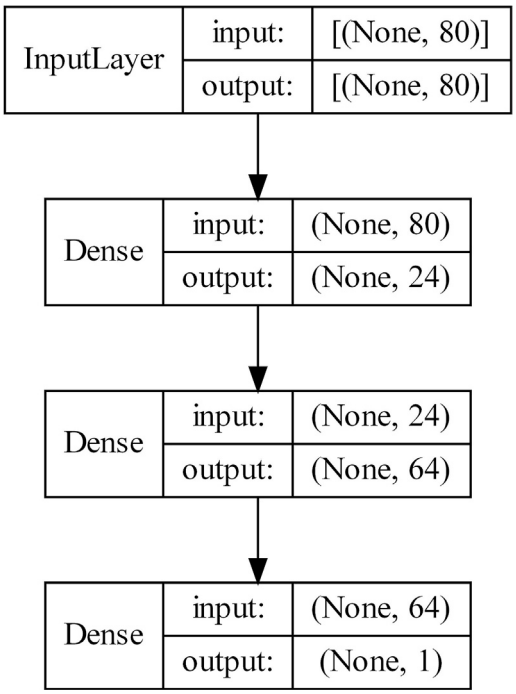


Figure 4. FFNN model's architecture

Note: Similar to CNN model, the feature vector for one monomer has a length of 80, and 100 feature vectors are stacked to represent a copolymer. The size of the input matrix is 271 × 100 × 80, as shown in Figure 5.

11. Build RNN model using the TensorFlow package and train the model on the copolymer dataset.
    a. Employ Keras RandomSearch to optimize hyperparameters such as the number of layers, the number of neurons in each layer, and the learning rate.
    b. Call the custom function getRNNmodel() to build units for the RNN model.
    c. Add 2 bidirectional long short-term memory (LSTM) layers with 20 neurons per layer, 1 time-distributed layer, and 1 reshape layer. The model architecture is illustrated in Figure 7.
    d. Train the model for 120 epochs on batch sizes of 4.
    e. Output the training R^2, MAE, and RMSE as well as the test R^2, MAE, and RMSE after the training is completed.

```
>def getRNNmodel(LSTMunits):
>    RNNmodel = Sequential()
>    RNNmodel.add(Bidirectional(LSTM(LSTMunits,
>    return_sequences=True), input_shape=(100, 80)))
>    RNNmodel.add(Bidirectional(LSTM(LSTMunits,
>    return_sequences=True)))
>    RNNmodel.add(TimeDistributed(Dense(int(LSTMunits/2),
>    activation="relu")))
>    RNNmodel.add(Reshape((int(LSTMunits/2*100),)))
>    RNNmodel.add(Dense(1))
>    return RNNmodel
```

```
>LSTMunits = 20
>RNNmodel = getRNNmodel(LSTMunits)
>RNNmodel.compile(loss='mse', optimizer='adam',
>    metrics=['mean_squared_error'])
>Model = RNNmodel.fit(X_train, y_train, validation_split=0.2,
>    epochs=200, batch_size=64)
```

12. Visualize results using matplotlib.

Note: The results visualization compares the predicted values from the RNN model against the ground truth. A parity plot is the outcome of the model training/validation.

Train fusion model

Timing: <2 min

```
Mix_X_100Block.shape
```

```
(271, 100, 80)
```

```
Mix_X_100Block
```

```
array([[[1, 0, 0, ..., 0, 1, 0],
        [1, 0, 0, ..., 0, 0, 0],
        [1, 0, 0, ..., 0, 1, 0],
        ...,
        [1, 0, 0, ..., 0, 1, 0],
        [1, 0, 0, ..., 0, 1, 0],
        [1, 0, 0, ..., 0, 1, 0]],

       [[1, 0, 0, ..., 0, 1, 0],
        [1, 0, 0, ..., 0, 0, 0],
        [1, 1, 0, ..., 0, 0, 0],
        ...,
        [1, 0, 0, ..., 0, 1, 0],
        [1, 1, 0, ..., 0, 0, 0],
        [1, 0, 0, ..., 0, 1, 0]],

       [[1, 0, 0, ..., 0, 1, 0],
        [1, 0, 0, ..., 0, 0, 0],
        [1, 0, 0, ..., 0, 0, 0],
        ...,
        [1, 0, 0, ..., 0, 1, 0],
        [1, 0, 0, ..., 0, 0, 0],
        [1, 0, 0, ..., 0, 1, 0]],

       ...,
```

Figure 5. The numerical input of copolymers for the CNN model

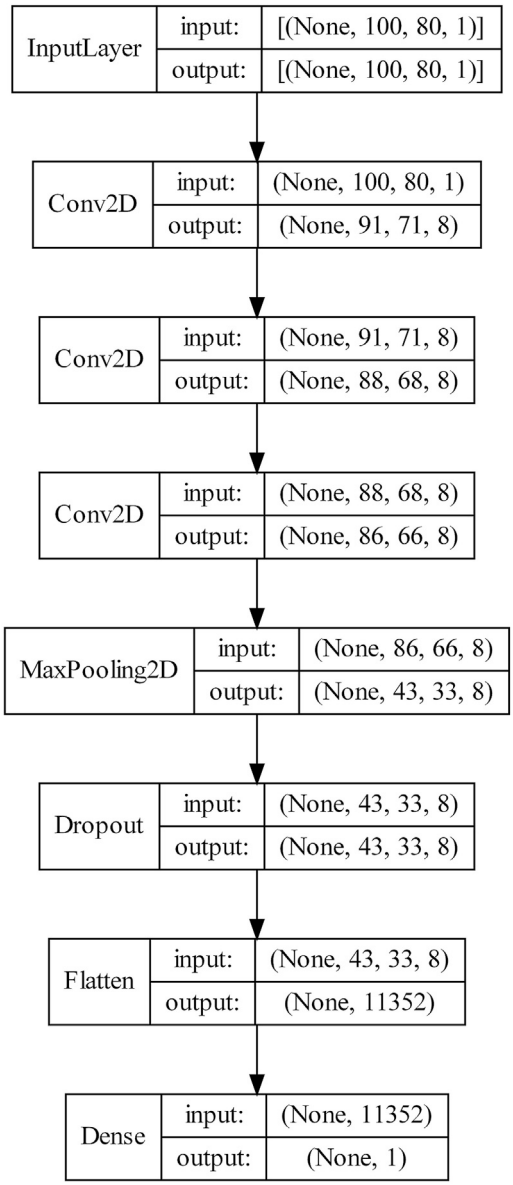


Figure 6. CNN model's architecture

13. Prepare the numerical input for the training of the Fusion model.
    a. Calculate the weighted sum of feature vectors to consider molecular composition in the FFNN component.
    b. Prepare a vector of 1/0 bit to represent the sequence distributions of copolymers in the RNN component.

Note: If using "1" for monomer "A" and "0" for monomer "B", then stacking 100 bits of 1/0 can represent the sequence distribution of copolymers.

14. Build the fusion model by combining the FFNN architecture and RNN architecture.
    a. Employ Keras RandomSearch to optimize hyperparameters such as the number of layers, the number of neurons in each layer, and the learning rate.

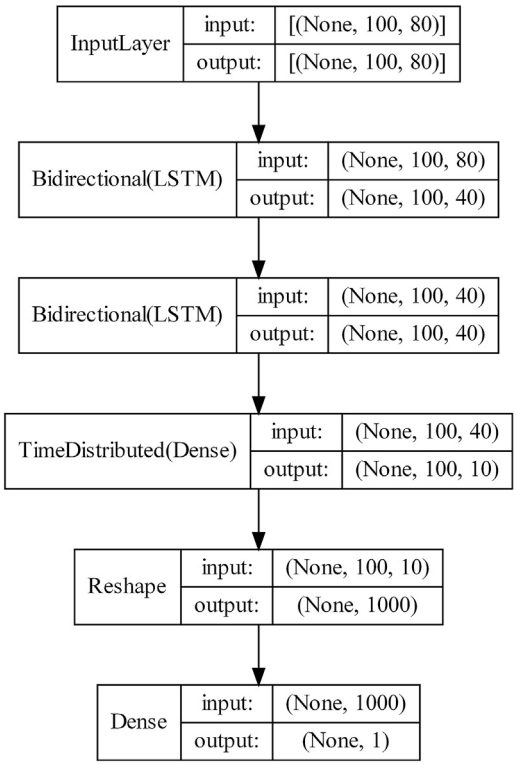


Figure 7. RNN model architecture

b. Add the Recurrent Neural Network (RNN) components of the Fusion model. Each unit has 2 bidirectional LSTM layers with 20 neurons each and 1 time-distributed layer.
c. Add the Feed Forward Neural Network (FFNN) component which contains 2 hidden layers with 8 neurons each with the 'ReLU' activation function.
d. Combine the two components using a concatenate layer and a final layer of 8 neurons with the 'ReLU' activation function. The model architecture is illustrated in Figure 8.
e. Train the model for 300 epochs on batch sizes of 32.
f. Output the training R^2, MAE, and RMSE as well as the test R^2, MAE, and RMSE after the training is completed.

```
>#define two sets of inputs
>inputA = Input(shape=(100,1))
>inputB = Input(shape=(80))
>#the first branch operates on the first input
>RNNmodel = Sequential()
>RNNmodel.add(Bidirectional(LSTM(LSTMunits,
>    return_sequences=True), input_shape=(100,1)))
>RNNmodel.add(Bidirectional(LSTM(LSTMunits,
>    return_sequences=True)))
>RNNmodel.add(TimeDistributed(Dense(int(LSTMunits/2),
>    activation="relu")))
```

```
>RNNmodel.add(Reshape((int(LSTMunits/2*100),)))
>
># the second branch opreates on the second input
>
>y = Dense(8, activation="relu")(inputB)
>
>y = Dense(8, activation="relu")(y)
>
>y = Model(inputs=inputB, outputs=y)
>
># combine the output of the two branches
>
>combined = Concatenate()([RNNmodel.output, y.output])
>
># apply a FC layer and then a regression prediction on the
>
># combined outputs
>
>z = Dense(8, activation="relu")(combined)
>
>z = Dense(1, activation="linear")(z)
>
># our model will accept the inputs of the two branches and
>
># then output a single value
>
>model = Model(inputs=[RNNmodel.input, y.input], outputs=z)
>
>model.compile(optimizer=keras.optimizers.Adam(lr=0.001),
>
>    loss="mean_squared_error",
>
>    metrics=["mean_squared_error"])
>
>Model = model.fit(x=[xtrain_A, xtrain_B], y=ytrain_B,
>
>    validation_data=([xtest_A, xtest_B], ytest_B),
>
>    epochs=300, batch_size=32, verbose=2)
```

15. Visualize results using matplotlib.

Note: The results visualization compares the predicted values from the Fusion model against the ground truth. A parity plot is the outcome of the model training/validation.

EXPECTED OUTCOMES

This protocol describes a step-by-step workflow to build four ML models to establish structure-property relationships for copolymers. Both monomer composition and sequence distribution are necessary information for ML models to learn the features of copolymers. Focusing on the different datasets, we build four ML model including FFNN, CNN, RNN, and Fusion model. The outcome of the protocol is a reliable ML model suitable for copolymer informatics. Different models' training loss and validation loss versus the number of epochs are shown in Figure 9. It is illustrated that the most difficult model to converge is the Fusion model considering the high peaks in its loss value curves. Since the Fusion model is a combination of FFNN and RNN models, it has a more complex architecture and contains more parameters to optimize during the gradient descent process. The two peaks indicate the complexity of the loss function and the difficulty of the convergence. One possible cause of the two peaks is there is a tradeoff between the convergence of its FFNN component and RNN component. Once the gradient descent leads to a more local minimum of the FFNN component, the loss of the RNN components may be increasing, and when it is corrected to a more local minimum of the RNN component, the loss of the FFNN components may start soaring. It is expected that after some adjustments, a balanced minimum for both FFNN and RNN components are obtained as the final global minimum.

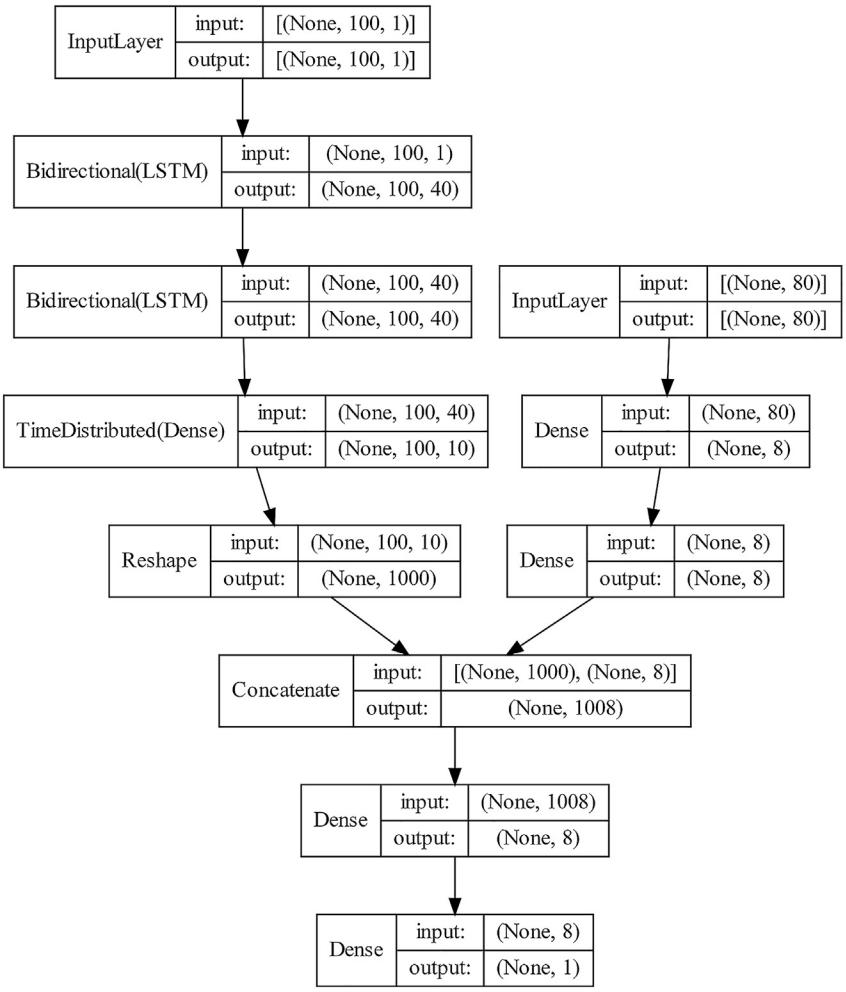


Figure 8. Fusion model's architecture

The parity plot comparing the ML prediction and experimental ground truth is demonstrated in Figure 10. Given a copolymer whose monomer composition and monomer sequence patterns (random, block, or gradient) are known, the ML model can provide reliable property predictions efficiently.

LIMITATIONS

The model architectures demonstrated in this protocol have been designed and tested for the polymer types and datasets referenced. Random, block, and gradient copolymers are investigated and tested, but branched and graft copolymers are not considered because of their complex chain architectures. As ML model performance is highly problem-dependent, further test is required in terms of the applicability of the investigated four ML models for other properties. This protocol demonstrates ML models for copolymers at monomer level by considering the monomer composition and monomer sequence patterns, but it doesn't encode microscale or macroscale level features of polymer such as average chain length, molecular weight, chain topology, crystallization, etc. As the performance of copolymers is determined by their features at different levels, the development of a multi-level ML model is preferred to better address the copolymer informatics challenges.

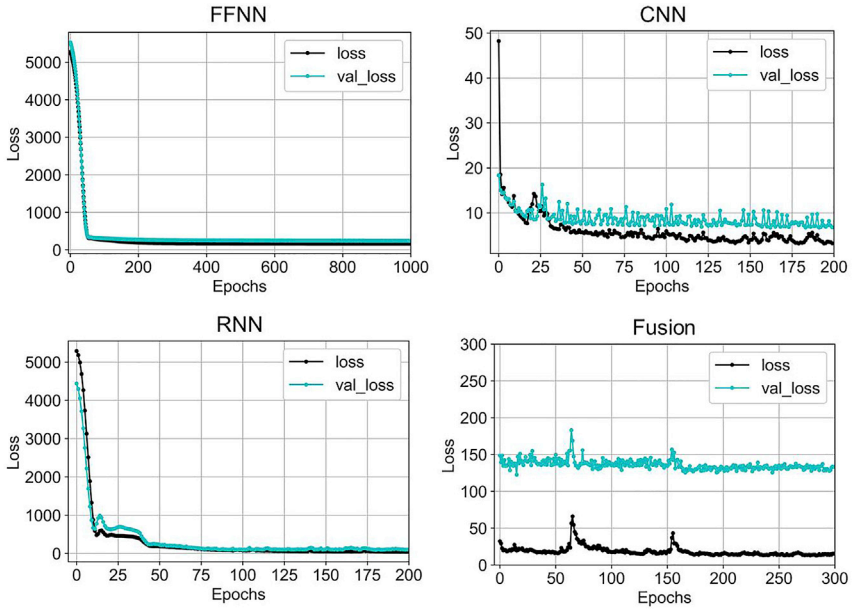


Figure 9. Training loss and validation loss of the ML models versus the number of epochs

TROUBLESHOOTING

Problem 1

Create environment using requirements.txt may output conflict or error because of different OS of computers (step Import Dependencies).

Potential solution

Create an environment and install each package manually.

- Download and install Anaconda https://www.anaconda.com/
- Create a new python 3.7 environment:

```
> conda create -n myenv python=3.7
```

- Pip install required packages (change package version may cause code errors):

```
>pip install RDkit
>pip install numpy==1.18.5
>pip install pandas==1.2.4
>pip install scipy=1.4.1
>pip install scikit-learn
>pip install matplotlib
>pip install -user keras==2.4.3
>pip install -user tensorflow==2.2.0
>pip install -user tensorflow-gpu==2.2.0
```

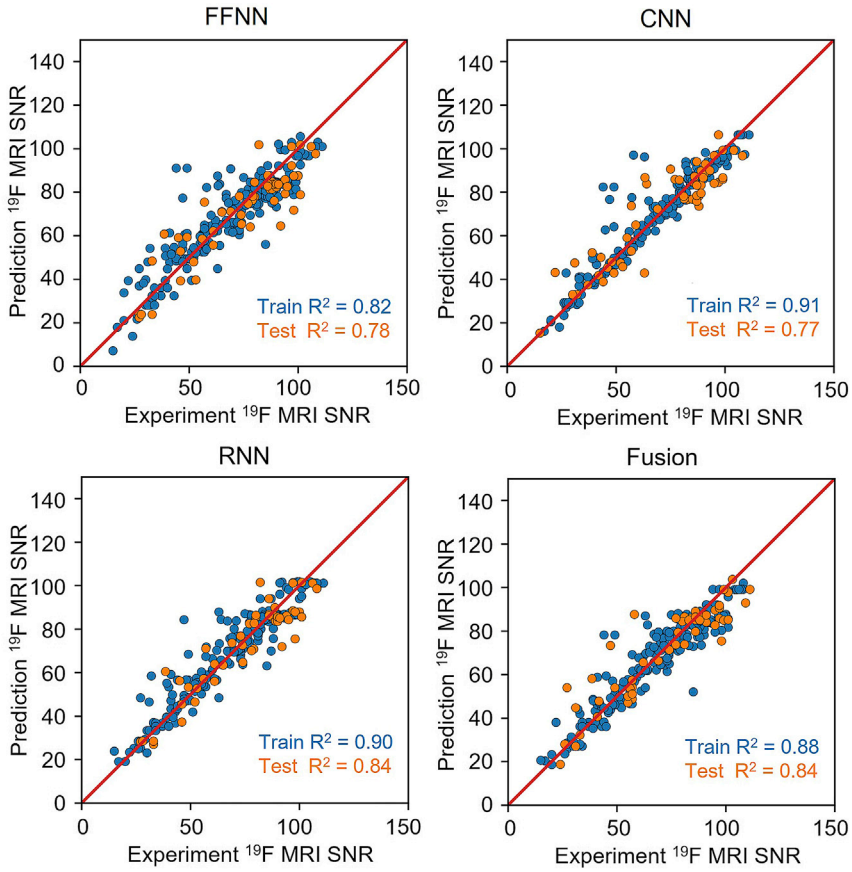


Figure 10. The parity plot of the four ML predicted SNR versus the experimental values

Problem 2

If plots appear only as text while running in Visual Studio code (step Construct Datasets).

Potential solution

Selecting the ellipsis next to the text, change the renderer to 'image/png'.

Problem 3

Looking for GPU but it shows no GPU error while calling the tensorflow GPU function (step Import Dependencies).

```
> physical_devices = tf.config.list_physical_devices('GPU')

> tf.config.list_physical_devices('GPU')
```

Potential solution

GPU is recommended but not required. Ignore and using the default CPU setup for the model training.

Problem 4

When processing the python dataframe, there may be warning message SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame (step Construct Datasets).

Potential solution

It is a chained assignment warning and can be turned off using.

```
> pd.set_option('mode.chained_assignment', None)
```

Problem 5

Load python pickle file error when python package version is not compatible. ValueError: unsupported pickle protocol: 5 (step Construct Datasets).

Potential solution

Pip install pickle5 in the anaconda environment.

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources and reagents should be directed to and will be fulfilled by the lead contact, Dr. Ying Li (yli2562@wisc.edu).

Materials availability

This study did not generate new unique reagents.

Data and code availability

This paper analyzes existing, publicly available data from publications and open website.

All original code has been deposited at https://github.com/figotj/Copolymer and archived at Zenodo (DOI) https://doi.org/10.5281/zenodo.7226849. They are publicly available as of the date of publication.

Any additional information required to reanalyze the data reported in this paper is available from the lead contact upon reasonable request.

ACKNOWLEDGMENTS

We gratefully acknowledge the financial support from the Air Force Office of Scientific Research through the Air Force's Young Investigator Research Program (FA9550-20-1-0183; Program Manager: Dr. Ming-Jen Pan), Air Force Research Laboratory/UES Inc. (FA8650-20-S-5008, PICASSO program), and the National Science Foundation (CMMI-1934829 and CAREER-2046751). Y.L. would also like to thank the support from 3M's Non-Tenured Faculty Award. This research also benefited in part from the computational resources and staff contributions by the Booth Engineering Center for Advanced Technology (BECAT) at the University of Connecticut. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the U.S. Department of Defense and National Science Foundation. The authors also acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin (Frontera project and the National Science Foundation award 1818253) for providing HPC resources that have contributed to the research results reported within this article.

AUTHOR CONTRIBUTIONS

Y.L. and J.B. conceived the idea. Y.L., J.B., and V.V. supervised the research. Y.L. and L.T. contributed to the design of the project and data analysis. L.T. collected and analyzed the data and established ML models. L.T. and T.A. wrote the first draft of the article, and all authors contributed equally to revising the article.

DECLARATION OF INTERESTS

The authors declare no competing interests.

REFERENCES

1. Tao, L., Byrnes, J., Varshney, V., and Li, Y. (2022). Machine learning strategies for the structure-property relationship of copolymers. iScience *25*, 104585.

2. Morgan, H.L. (1965). The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service. J. Chem. Doc. *5*, 107–113.

3. Tao, L., Varshney, V., and Li, Y. (2021). Benchmarking machine learning models for polymer informatics: an example of glass transition temperature. J. Chem. Inf. Model. *61*, 5395–5413.

4. Tao, L., Chen, G., and Li, Y. (2021). Machine learning discovery of high-temperature polymers. Patterns *2*, 100225.

5. Wilbraham, L., Sprick, R.S., Jelfs, K.E., and Zwijnenburg, M.A. (2019). Mapping binary copolymer property space with neural networks. Chem. Sci. *10*, 4973–4984.

6. Reis, M., Gusev, F., Taylor, N.G., Chung, S.H., Verber, M.D., Lee, Y.Z., Isayev, O., and Leibfarth, F.A. (2021). Machine-learning-guided discovery of 19F MRI agents enabled by automated copolymer synthesis. J. Am. Chem. Soc. *143*, 17677–17689.

7. Pilania, G., Iverson, C.N., Lookman, T., and Marrone, B.L. (2019). Machine-learning-based predictive modeling of glass transition temperatures: a case of polyhydroxyalkanoate homopolymers and copolymers. J. Chem. Inf. Model. *59*, 5013–5025.

8. Otsuka, S., Kuwajima, I., Hosoya, J., Xu, Y., and Yamazaki, M. (2011). PoLyInfo: Polymer Database for Polymeric Materials Design (IEEE)), pp. 22–29.