DeepAffinity: Interpretable Deep Learning of Compound-Protein Affinity through Unified Recurrent and Convolutional Neural Networks (Supplemental Data)

Mostafa Karimi[1,2], Di Wu[1], Zhangyang Wang[3] and Yang Shen[1,2,*]

[1] Department of Electrical and Computer Engineering,
[3] Department of Computer Science and Engineering, and
[2] TEES-AgriLife Center for Bioinformatics and Genomic Systems Engineering, Texas A&M University, College Station, 77843, USA.

[*] Contact: yshen@tamu.edu

1 Materials and Methods

1.1 Data curation

For the labeled data, we chose the half maximal inhibitory concentration (IC_{50}) as the main measure of binding affinity and extracted samples with known IC_{50} values from BindingDB; and we converted compound data from the structure-data file (SDF) format to canonical SMILES by the software openbabel (O'Boyle *et al.*, 2011). Moreover, as 95% of compounds and proteins with known IC_{50} values are of lengths up to 100 (SMILES characters) and 1500 (amino acids), respectively, we removed samples outside of the ranges as well as samples with incomplete information to determine protein sequences, questionable SDF files that openbabel couldn't convert into SMILES, or IC_{50} as a range instead of exact value (unless it is worse than a weak 30 μM or better than a tight 1 nM). In the end, we collected 413,635 samples from BindingDB, which are split into training, testing, and 3 generalization sets.

We used IC_{50} (and K_i and K_d as well) for a measure of binding affinity and its logarithm form ($pIC_{50} = -\log_{10} IC_{50}$) as the regression target. Prior to taking the logarithm, we truncated IC_{50} to the range of $[1 \times 10^{-9}\mathrm{M}, 3 \times 10^{-5}\mathrm{M}]$ considering the sensitive ranges of experimental methods for IC_{50} measurement. In other words, measurements stronger than 1 nM or weaker than 30 μM would be regarded 1 nM or 30 μM, respectively. Similarly, for both the K_i and K_d datasets, measurements were truncated to the range of $[10^{-10}\mathrm{M}, 10^{-4}\mathrm{M}]$ before being converted to pK_i and pK_d.

1.2 Protein representation

Groups	Amino-acid members	# of members	Background frequency
Nonpolar	Gly, Ala, Val, Leu, Ile, Met, Phe, Pro, Trp	9	0.505
Polar	Ser, Thr, Tyr, Cys, Gln, Asn	6	0.253
Acidic	Asp, Glu	2	0.111
Basic	Lys, Arg, His	3	0.131

Table S1: Grouping amino acids based nonpolar, polar, basic and acidic and their average percentage of existence in each secondary structure

Each SSE classified based on secondary structure category, solvent accessibility, physicochemical properties, and residue length. Specifically, an SSE is solvent exposed if at least 30% of residues are and buried otherwise; polar, non-polar, basic or acidic based on the highest odds (for each type, occurrence frequency in the SSE is normalized by background frequency seen in all protein sequences to remove the effect from group-size difference, See Table S1); short if length $L \leqslant 7$, medium if $7 < L \leqslant 15$, and long if $L > 15$.

Secondary Structure			Solvent Exposure		Property				Length		
Alpha	Beta	Coil	Not Exposed	Exposed	Non-polar	Polar	Acidic	Basic	Short	Medium	Long
A	B	C	N	E	G	T	D	K	S	M	L

Table S2: 4-tuple of letters in protein structural property sequence (SPS) for creating words

In this way, we defined 4 “alphabets” of 3, 2, 4 and 3 “letters”, respectively to characterize categories in SSE, solvent accessibility, physicochemical characteristics, and length (Table S2) and combined letters from the 4 alphabets in the order above to create 72 “words” (4-tuples) to describe SSEs. For example, the word “AEKM” means an α-type SSE that is solvent-exposed, overly basic, and of medium length.

We designed an algorithm to generate SSEs from predicted secondary structure classes based on two rules: the minimum size of α- or β SSEs is 5 and that for coil SSEs is 3; a group of less size is merged to neighboring SSEs that satisfy the minimum size, by sliding a window of size 7 down the sequence to "smooth" it. The rules are adopted to address uncertain or "orphan" groups of few members.

Algorithm 1 "Smoothing" Secondary Structure Sequence

```
Require: Secondary Structure of Protein Sequence
Ensure: Smoothed Secondary Structure Sequence
  Initialize MeaningfulMinLength as 3 for Coil and 5 for Sheet and Helix
  Initialize MaxGroupLength as 7
  for Each line in Secondary Structure do
    Initialize FinalSeq as Empty
    for Each letter in line do
      if Current = Next then
        Store Current in S
        Mark the length of continue same sequence as L
      else if $L > MeaningfulMinLength$ then
        Smooth all letters in S to last letter
        Store S to FinalSeq and clear S
        Continue
      else if Length of $S < MaxGroupLength$ then
        Store to S
        Continue
      else
        Smooth all letters to the most number letter in S
        Store S to FinalSeq and Clear S
        Continue
      end if
    end for
    return FinalSeq
    Clear FinalSeq
  end for
```

Algorithm 2 Secondary Structure Element (SSE) Sequence to Structure Property Sequence (SPS)

```
Require: Smoothed Secondary Structure and Exposedness Protein Sequence
Ensure: Structural Property Sequence (SPS).
  Initialize exposedness threshold $eThres$
  Initialize Polarity threshold $pThres$ based on the average of each types
  for Each $ss$ in Secondary Structure Element Sequences do
    Initialize $FinalSeq$ as Empty
    Read next Exposedness Protein Sequence to $acc$
    Read next Protein Sequence to $protein$
    for Each continuous same letter in $ss$ do
      Store corresponding letter to $G$
      if the number of 'e' in $ss > eThres$ then
        Store 'E' to $G$
      else
        Store 'N' to $G$
      end if
      if percentage of one type polarity in $protein > pThres$ then
        Store corresponding letter to $G$
      end if
      if Length of group $\leq 7$ then
        Store 'S' letter to $G$
      else if Length of group $\leq 15$ then
        Store 'M' letter to $G$
      else if Length of group $> 15$ then
        Store 'L' letter to $G$
      end if
      Store $G$ to $FinalSeq$
      Clear $S$
    end for
    return  $FinalSeq$
    Clear $FinalSeq$
  end for
```

1.3 Shallow machine learning models

Shallow machine learning models — ridge, lasso, and random forest (RF) — were used to compare learned protein/compound representations with baseline ones and implemented using scikit-learn (Pedregosa *et al.*, 2011). Hyper parameters include regularization constant $\alpha \in \{10^{-4}, 10^{-3}, \ldots, 10^{4}\}$ for lasso and ridge regression as well as the number of trees $\in \{50, 100, 200, 500\}$, the minimum sample per leaf $\in \{10, 50, 100\}$ and the maximum number of features for each tree $\in \{n, \sqrt{n}, \log_2 n\}$ for random forest where n denotes the number of features. These hyper parameters were optimized using 10-fold cross validation.

1.4 RNN for Unsupervised Learning

1.4.1 Introduction to GRU

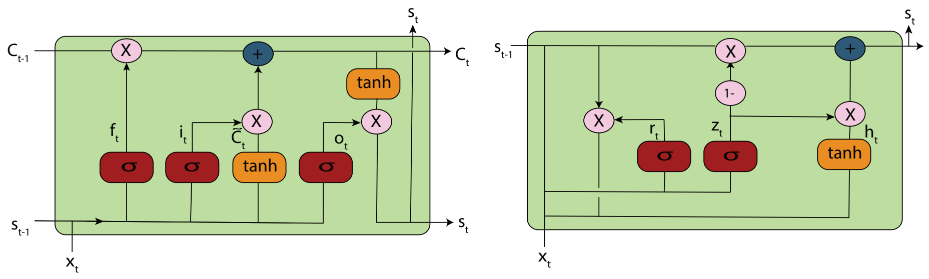


Figure S1: LSTM and GRU

As shown in Fig. S1, an LSTM unit consists of three "gates" — input (i_t), forget (f_t) and output (o_t) gates as well as two states — cell (also memory, C_t) and hidden states (also output, s_t). LSTM will iterate over the input sequence $(x_1, \cdots, x_t, \cdots, x_T)$ and calculate the output sequence of $(s_1, \cdots, s_t, \cdots, s_T)$ through:

$$\begin{aligned}
f_t &= \sigma(W_f \times [s_{t-1}, x_t] + b_f) \\
i_t &= \sigma(W_i \times [s_{t-1}, x_t] + b_i) \\
\tilde{C}_t &= \tanh(W_c \times [s_{t-1}, x_t] + b_c) \\
C_t &= f_t \times C_{t-1} + i_t \times \tilde{C}_t \\
o_t &= \sigma(W_o \times [s_{t-1}, x_t] + b_o) \\
s_t &= o_t \times \tanh(C_t)
\end{aligned} \tag{1}$$

GRU further simplifies LSTM without sacrificing the performance much. With much less parameters, it demands lighter computational time. A GRU unit consists of two gates — reset (r_t) and update (Z_t) gates as well as one state — hidden state (s_t). It calculates the output sequence of $(s_1, \cdots, s_t, \cdots, s_T)$ over the input sequence $(x_1, \cdots, x_t, \cdots, x_T)$ through:

$$\begin{aligned}
z_t &= \sigma(W_z \times [s_{t-1}, x_t] + b_z) \\
r_t &= \sigma(W_r \times [s_{t-1}, x_t] + b_r) \\
h_t &= \tanh(W_h \times [r_t \times s_{t-1}, x_t] + b_h) \\
s_t &= (1 - z_t) \times s_{t-1} + z_t \times h_t
\end{aligned} \tag{2}$$

1.4.2 GRU implementation

Our alphabets include 45 and 76 letters (including special symbols such as padding) for compound SMILES and protein SPS strings, respectively. Based on the statistics of 95% CPIs in BindingDB, we set the maximum lengths of

SMILES and SPS strings to be 100 and 152, respectively. Accordingly, we used 2 layers of GRU with both the latent dimension and the embedding layer (discrete letter to continuous vector) dimension being 128 for compounds and 256 for proteins. We used an initial learning rate of 0.5 with a decay rate of 0.99, a dropout rate of 0.2, and a batch size of 64. We also clipped gradients by their global norms. All neural network models for supervised learning were implemented based on TensorFlow (Abadi *et al.*, 2016) and TFLearn (Tang, 2016).

We further test four more seq2seq variants under the combination of the following options:

- **Bucketing** (Khomenko *et al.*, 2016) as an optimization trick to put sequences of similar lengths in the same bucket and padded accordingly during training. We used bucketing groups of {(30,30),(60,60),(90,90),(120,120),(152,152)} for SPS and {(20,20),(40,40),(60,60),(80,80),(100,100)} for SMILES.
- **Bidirectional GRU** that shares parameters to capture both forward and backward dependencies (Schuster and Paliwal, 1997).
- **Attention** mechanism (Bahdanau *et al.*, 2014) that allows encoders to "focus" in each encoding step on selected previous time points that are deemed important to predict target sequences.

Both bidirectional RNN and attention mechanism can help address computational challenges from long input sequences.

1.4.3 Attention mechanism for unsupervised learning

To address the challenge from long input sequences, the attention mechanism provides a way to "focus" for encoders. It only allows each encoding step to be affected by selected previous time points deemed important, thus saving its memory burden. Suppose that the maximum length of both the encoder and the decoder is L, the output of RNN (the input to the attention model) is $(s_1, \cdots, s_t, \cdots, s_L)$, the hidden state of the decoder $(h_1, \cdots, h_j, \cdots, h_L)$, and the output of attention model is C_j at each time step j of the decoder. The attention model is parametrized by two matrices, U_a and W_a, and a vector $\boldsymbol{v}_\mathrm{a}$ ('a' stands for attention) and formulated as:

$$\begin{aligned}
e_{j,t} &= \boldsymbol{v}_\mathrm{a} \tanh(U_\mathrm{a} h_{j-1} + W_\mathrm{a} s_t) \quad \forall j = 1, \cdots, L \quad \text{and} \quad \forall t = 1, \cdots, L \\
\alpha_{j,t} &= \frac{exp(e_{j,t})}{\sum_{k=1}^{L} exp(e_{j,k})} \quad \forall j = 1 \cdots L \quad and \quad t = 1 \cdots L \\
C_j &= \sum_{t=1}^{L} \alpha_{j,t} s_t \quad \forall j = 1 \cdots L
\end{aligned} \tag{3}$$

All seq2seq models were implemented based on the seq2seq code released by Google (Britz *et al.*, 2017).

1.5 Unified RNN-CNN for supervised learning

1.5.1 CNN

Convolutional neural networks (CNN) (Lawrence *et al.*, 1997) have made great success in modeling image data for computer vision. For either proteins or compounds, our CNN model consisted of a one-dimensional (1D) convolution layer followed by a max-pooling layer. The outputs of these layers for proteins and compounds were concatenated then fed to two fully connected (or dense) layers.

In particular, the 1D convolution layer had 64 various filters (all have length 8 and stride 4 for proteins and length 4 and stride 2 for compounds). The max-pooling layer with a filter of length 4 was to reduce the parameters for the next layers and introduce nonlinearity to the predictor. The 2 fully connected layers are with 200 and 50 neurons, respectively, activation function of leaky RELU (Maas *et al.*, 2013), Xavier initialization (Glorot and Bengio, 2010), and a drop-out rate of 20%.

1.5.2 Separate and unified RNN-CNN models

RNNs (encoders and attention models only) for unsupervised learning and CNNs for supervised learning are connected in series and trained separately or jointly (see Fig. S2).

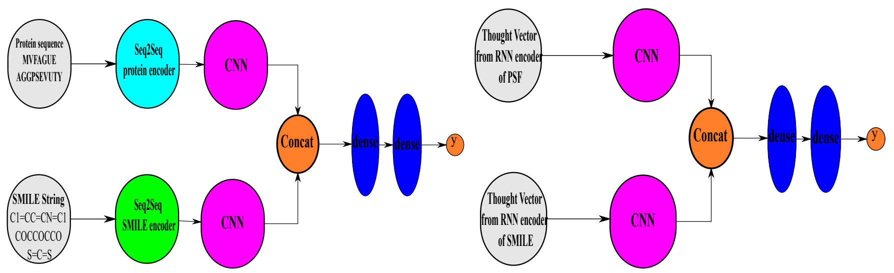

Protein sequence MVFAGUE AGGPSEVUTY
Seq2Seq protein encoder
CNN
SMILE String C1=CC=CN=C1 COCCOCCO S=C=S
Seq2Seq SMILE encoder
CNN
Concat
dense
dense
y
Thought Vector from RNN encoder of PSF
CNN
Thought Vector from RNN encoder of SMILE
CNN
Concat
dense
dense
y

Figure S2: Separate and unified RNN-CNN models.

In separate RNN-CNN models, a seq2seq model for proteins or compounds is trained once for context-specific representations and thought vectors from its encoder are fed as the input to train a subsequent CNN model. In unified RNN-CNN models, both RNN (just encoder + attention model) and CNN models are trained together to jointly learn representations and make predictions, which makes the representations further relevant to the specific task.

Considering that training the unified model involves a non-convex optimization problem in a much higher dimensional parameter space than training the CNN model alone, we used previously-learned GRU parameters as starting points to have a "warm start" for optimization.

Specifically, we first fixed parameters of RNN encoders and trained the attention models, CNN layers and fully connected layers over the labeled training set for 100 epochs with a learning rate of 10^{-3}. After the newly-trained layers are also warmed up (i.e., initialized), we jointly trained the unified RNN-CNN model for 200 epochs with a learning rate of 10^{-4}.

In other words, we fine-tuned protein or compound representations learned from seq2seq previously for compound-protein affinity prediction.

1.5.3 Attention mechanism for unified RNN-CNN models

We have also introduced protein and compound attention models in supervised learning to both improve predictive performances and enable model interpretability at the level of "letters" (SSEs in proteins and atoms in compounds). In the supervised model we just have the encoder and its attention α_t on each letter t for a given string $\mathbf{x}$. And the output of the attention model, C, will be the input to the subsequent 1D-CNN model. Suppose that the length of encoder is L and $(s_1, \cdots, s_t, \cdots, s_L)$ are the output of encoder. We parametrize the attention model of unified model with matrix U_{a} and the vector $\boldsymbol{v}_{\mathrm{a}}$. Then, The attention model is formulated as:

$$\begin{aligned} e_t &= \boldsymbol{v}_{\mathrm{a}} \tanh(W_{\mathrm{a}} s_t) \quad \forall t = 1, \cdots, L \\ \alpha_t &= \frac{exp(e_t)}{\sum_{k=1}^{L} exp(e_k)} \quad \forall t = 1 \cdots L \\ C &= \sum_{t=1}^{L} \alpha_t s_t \end{aligned} \tag{4}$$

The attention weights (scores) α_t suggest the importance of the t^{th} "letter" (secondary structure element in proteins and atom or connectivity in compounds) and thus predict the binding sites relevant to the predicted binding affinity.

All neural network models for supervised learning were implemented based on TensorFlow (Abadi *et al.*, 2016) and TFLearn (Tang, 2016).

1.6 Deep transfer learning

For deep transfer learning, we fine-tuned the first embedding layer (proteins or compounds) and the last two fully connected layers while fixing the RNN, attention and CNN layers of the model. Parameters for those fine-tuned layers were initialized at values in the original models and optimized over the new training data. Batch sizes were set at 2, 10, and 64 for ultra-low (1%), low (5%, 10%), and medium (30%, 50%) training coverage of each set, respectively.

We held out 30% of each labeled generalization set for testing and incrementally made available {1%,5%,10%,30%,50%} of each such set for training.

2 Prediction Results

2.1 Unsupervised learning for representation pre-training

	seq2seq	+bucketing	+fw/bw	+attention	+attention+fw/bw
Number of iterations	400K	400K	400K	400K	400K
Training error (perplexity)	7.13	7.11	2.00	3.10	**1.001**
Testing error (perplexity)	6.83	6.89	1.94	3.25	**1.0001**
Time (h)	39.33	48	62.28	84.13	81.36

Table S3: Performance comparison among 5 variants of seq2seq for compound representation based on perplexity under the limit of 4-day running time and 400K iterations.

	seq2seq	+bucketing	+fw/bw	+attention	+attention+fw/bw
Number of iterations	400K	400K	400K	153K	153K
Training error(perplexity)	40.85	40.52	16.77	1.007	**1.003**
Testing error(perplexity)	41.03	43.19	19.62	1.001	**1.001**
Time (h)	80.7	83.75	79.89	96	96

Table S4: Performance comparison among 5 variants of seq2seq for protein representations based on perplexity under the limit of 4-day running time and 400K iterations.

2.2 K_i prediction

	Baseline representations			Novel representations		
Error	Ridge	Lasso	RF	Ridge	Lasso	RF
Training	1.20 (0.55)	1.20 (0.55)	0.82 (0.83)	1.29 (0.44)	1.30 (0.43)	**0.68** (0.88)
Testing	1.23 (0.52)	1.23 (0.52)	0.98 (0.74)	1.30 (0.44)	1.31 (0.43)	**0.89** (0.78)
Estrogen receptors	1.99 (0.43)	2.00 (0.42)	2.09 (0.22)	1.95 (0.05)	1.94 (0.09)	**1.94** (0.10)
Ion channels	1.33 (0.1)	1.48 (0.17)	1.46 (0.10)	1.19 (0.07)	1.18 (0.10)	**1.02** (0.07)
GPCRs	1.03 (0.24)	1.02 (0.24)	0.99 (0.33)	0.66 (0.72)	**0.64** (0.75)	0.98 (0.56)
Time (core hours)	1.91	2.88	300.82	0.19	1.56	273.46
Memory (Gb)	3	3	3.2	3.2	3.2	2.7

Table S5: Comparing the baseline and the novel representations based on RMSE (Pearson correlation) for pK_i prediction.

	RF	Separate Model			Unified Model		
		single	param. ensemble	param.+NN ensemble	single	param. ensemble	param.+NN ensemble
Training	0.68 (0.88)	0.76 (0.84)	0.75 (0.85)	0.75 (0.86)	0.48 (0.94)	0.46 (0.94)	**0.45** (0.94)
Testing	0.89 (0.78)	0.94 (0.76)	0.93 (0.76)	0.92 (0.77)	0.82 (0.82)	0.81 (0.83)	**0.77** (0.84)
Estrogen receptors	1.94 (0.10)	2.03 (0.24)	1.87 (0.26)	**1.84** (0.26)	1.95 (0.2)	1.93 (0.22)	1.97 (0.10)
Ion channels	**1.02** (0.07)	1.39 (0.17)	1.45 (0.17)	1.32 (0.10)	1.36 (0.10)	1.35 (0.09)	1.26 (0.06)
GPCRs	0.98 (0.56)	1.32 (0.55)	1.07 (0.52)	1.13 (0.55)	0.90 (0.42)	0.89 (0.46)	**0.82** (0.55)

Table S6: Comparing random forest and variants of separate or unified RNN-CNN models based on RMSE (Pearson correlation) for pK_i prediction.

2.3 K_d prediction

	Baseline Representations			Novel Representations		
Error	Ridge	Lasso	RF	Ridge	Lasso	RF
Training	1.20 (0.68)	1.21 (0.67)	1.06 (0.77)	1.27 (0.64)	1.26 (0.64)	**0.91** (0.84)
Testing	1.24 (0.65)	1.25 (0.64)	1.16 (0.70)	1.29 (0.61)	1.30 (0.61)	**1.06** (0.76)
Estrogen receptors	2.54 (0.43)	2.46 (0.40)	2.84 (0.37)	**2.43** (0.68)	2.56 (0.47)	2.69 (0.03)
Ion channels	1.04 (0.56)	**1.00** (0.60)	1.23 (0.30)	1.49 (0.34)	1.67 (0.42)	1.45 (0.53)
Time (core hours)	0.14	0.22	7.16	0.01	0.1	12.66
Memory (Gb)	0.5	0.5	0.5	0.4	0.4	0.3

Table S7: Comparing the baseline and the novel representations based on RMSE (Pearson correlation) for pK_d prediction.

	RF	Separate Models			Unified Models		
		single	param. ensemble	param.+NN ensemble	single	param. ensemble	param.+NN ensemble
Training	0.91 (0.84)	0.67 (0.91)	0.65 (0.91)	0.65 (0.91)	0.58 (0.93)	0.57 (0.93)	**0.56** (0.93)
Testing	1.06 (0.76)	0.99 (0.79)	0.99 (0.80)	0.97 (0.80)	0.97 (0.8)	0.97 (0.80)	**0.92** (0.83)
Estrogen receptors	**2.69** (0.03)	2.80 (0.48)	2.80 (0.50)	3.00 (0.36)	2.93 (0.83)	2.97 (0.78)	3.07 (0.60)
Ion channels	1.45 (0.53)	1.47 (0.14)	1.47 (0.14)	1.32 (0.12)	1.29 (0.37)	1.29 (0.34)	**1.20** (0.30)

Table S8: Comparing random forest and variants of separate or unified RNN-CNN models based on RMSE (Pearson correlation) for pK_d prediction.

2.4 Generalization sets v.s. Training and testing sets

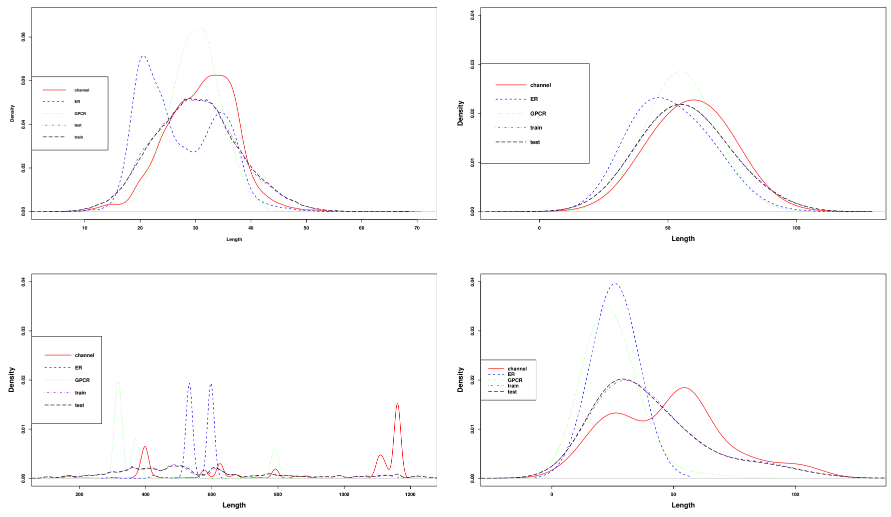


Figure S3: Comparing marginal distributions of various sets of compounds in size, i.e., the number of atoms (top left); compounds in SMILES length (top right); protein targets in the amino-acid length (bottom left); and protein targets in the SPS string length (bottom right).

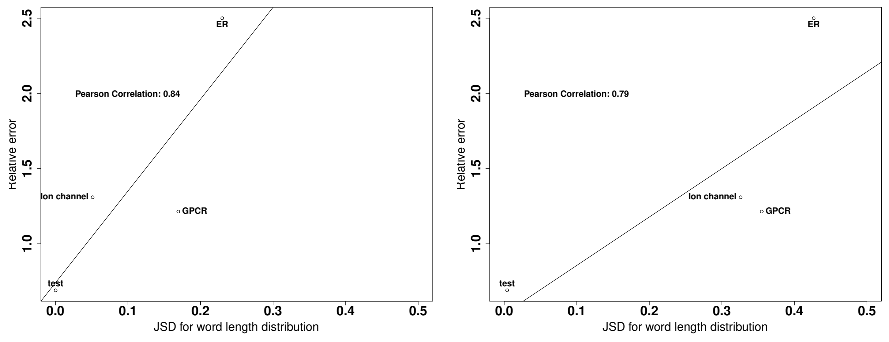
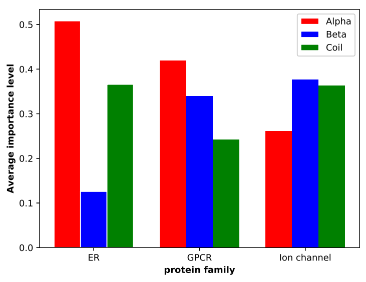


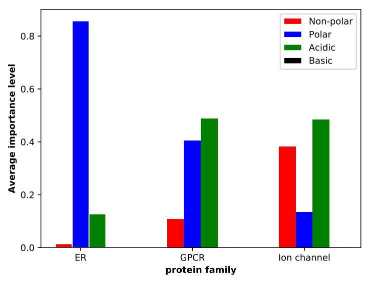
Figure S4: Relative errors to the training set (y axis) versus Jensen-Shannon distances from the training-set protein SPS word distribution (x axis: left) or word length distribution (x axis: right) for various sets of protein targets.

3 Where “attentions” are for generalization sets

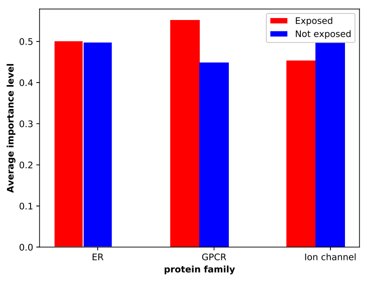
To interpret the unified RNN-CNN models’ predictive behaviors in the three generalization sets, we examined attention weights or scores (α_j’s for protein secondary structure element / SSE j) from attention models of the single model. Specifically, to assess which protein features are relevant to interacting each class of proteins, we presented the attention scores averaged over each “letter” for each of the four structural “alphabets” (SSE type, solvent accessibility, length, and amino-acid composition) in Fig. S5 as well as the average scores for each “word” as a 4−tuple of the letters in Fig. S6–8 (Supplementary Data). For estrogen receptors, α helices and coils as well as polar residues were much more focused on, which agrees with the fact that its helix 12 and neighboring loop are binding sites for cofactors and drugs. For GPCRs which represent a more diverse class, although the 7-helical transmembrane proteins’ secondary structures are mostly α helices, much attention was still paid to β strands and coils which are secondary structures of known extracellular drug binding site especially for class A GPCRs. Also, polar or acid residues were much more focused on than non-polar or basic residues. For ion channels, SSEs rich in nonpolar and acid residues were more focused on.



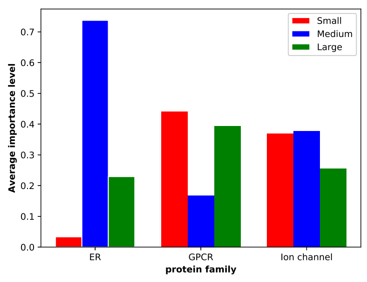
(a) Secondary structure element (SSE) type



(b) SSE amino-acid composition



(c) SSE solvent exposure



(d) SSE length

Figure S5: Attention scores averaged over various SSE (secondary structure element) properties for estrogen receptors, ion channels, and GPCRs.

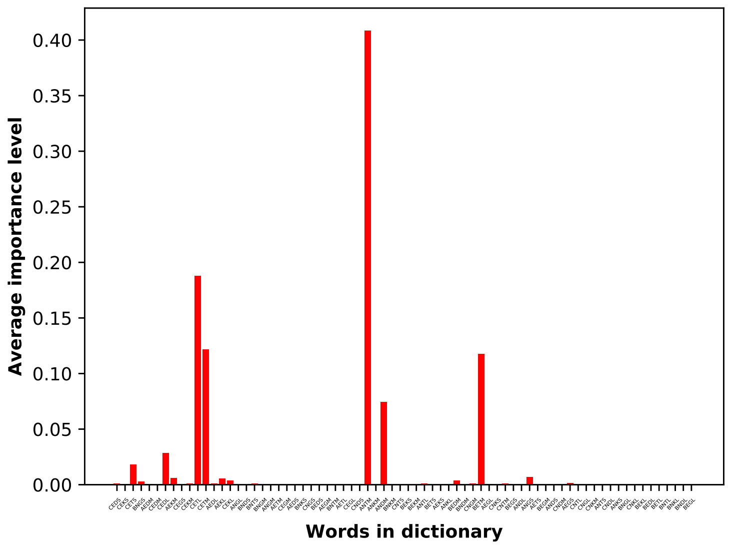


Figure S6: Average attention scores of SPS (structural property sequence) "words" for estrogen receptors.

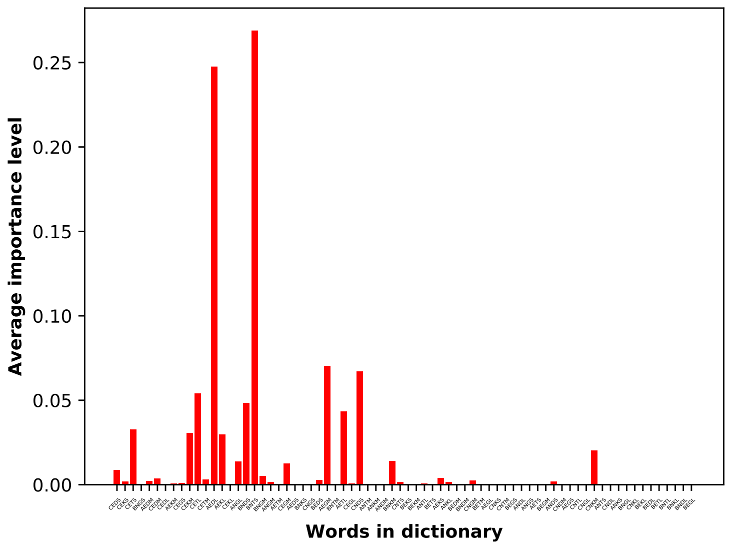


Figure S7: Average attention scores of SPS (structural property sequence) "words" for GPCRs.

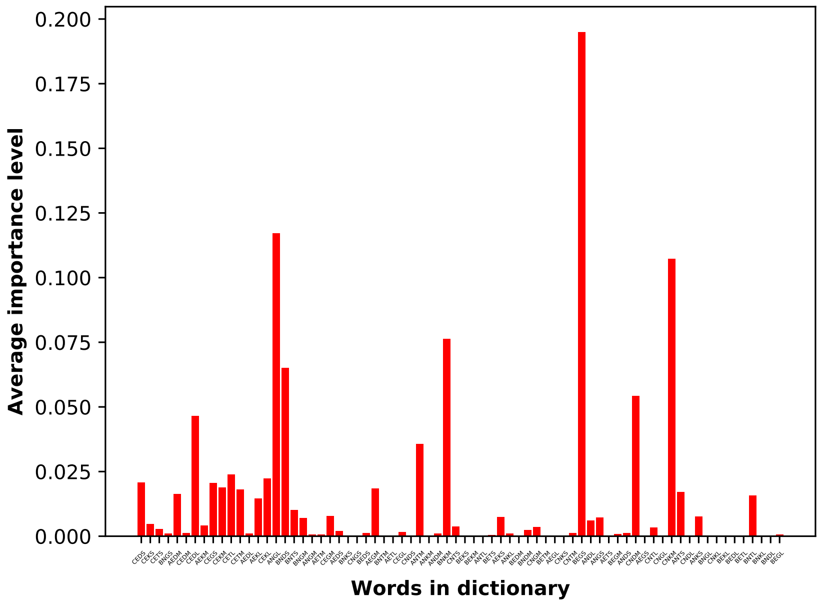


Figure S8: Average attention scores of SPS (structural property sequence) “words” for ion channels.

4 Attention weights for the unified model

4.1 Cyclooxygenas (COX) protein family

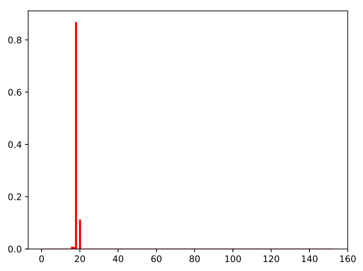
COX protein family represents an important class of drug targets for inflammatory diseases. These enzymes responsible for prostaglandin biosynthesis include COX-1 and COX-2 in human, both of which can be inhibited by nonsteroidal anti-inflammatory drugs (NSAIDs). We chose three common NSAIDs known for human COX-1/2 selectivity: celecoxib (pIC_{50} for COX-1: 4.09; COX-2: 5.17), ibuprofen (COX-1: 4.92, COX-2: 4.10) and rofecoxib (COX-1: ¡4; COX-2: 4.6) (Luo *et al.*, 2017). COX-1 and COX-2 both exist in our training set (2,097 COX-1 examples including 1,346 human COX-1 and 2,057 COX-2 examples including 1,924 human COX-2) and the three NSAIDs do not.

	Baseline rep. + RF			Novel rep. + RF			Novel rep. + DL		
	celecoxib	ibuprofen	rofecoxib	celecoxib	ibuprofen	rofecoxib	celecoxib	ibuprofen	rofecoxib
COX-1	8.89	8.55	8.76	8.45	8.26	8.07	**7.92**	**8.21**	7.53
COX-2	8.89	8.55	8.76	**8.83**	**8.53**	**9.16**	7.59	7.89	**7.88**

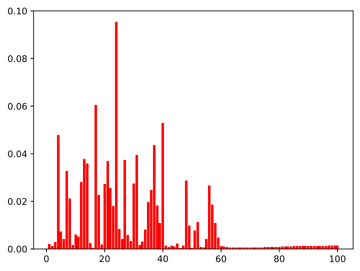
Table S9: Predicted pIC_{50} values and target specificity for three NSAIDs interacting with human COX-1 and COX-2.

From Table S9, we noticed that, using the baseline representations, random forest incorrectly predicted COX-1 and COX-2 to be equally favorable targets for each drug. This is because the two proteins are from the same family and their representations in Pfam domains are indistinguishable. Using the novel representations, random forest correctly predicted target selectivity for two of the three drugs (ibuprofen and rofexib), whereas our unified RNN-CNN models did so for all three.

We also present attention weights and distributions for each compound-protein pair (including COX-family, factor Xa and thrombin, and PTP family) in the following figures.

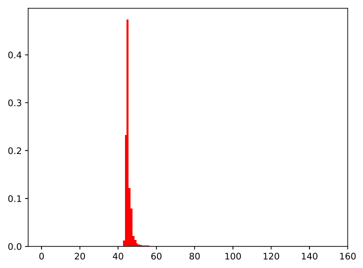


(a) COX1

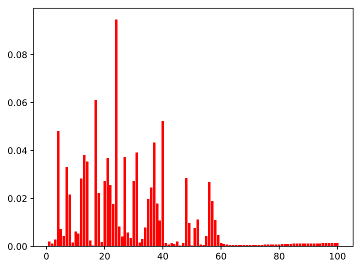


(b) celecoxib

Figure S9: COX1 celecoxib

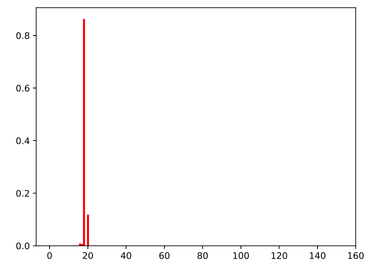


(a) COX2

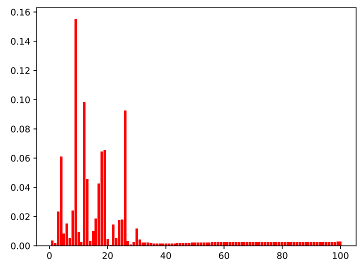


(b) celecoxib

Figure S10: COX2 celecoxib

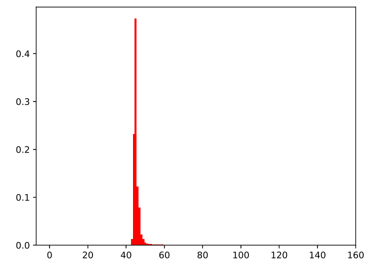


(a) COX1

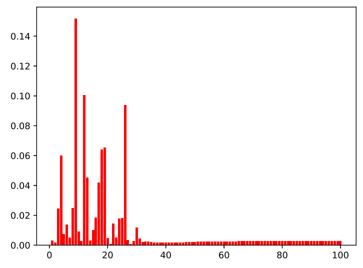


(b) ibuprofen

Figure S11: COX1 ibuprofen

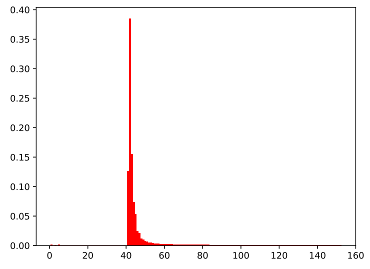


(a) COX2

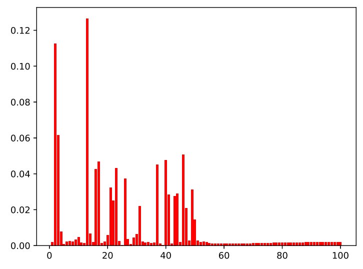


(b) ibuprofen

Figure S12: COX2 ibuprofen



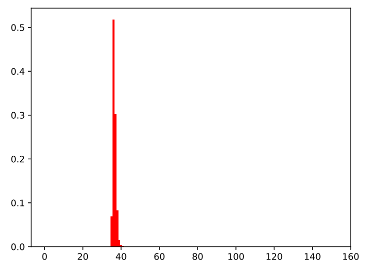
(a) COX2



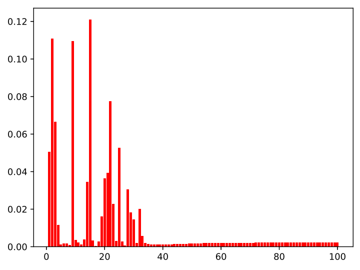
(b) rofecoxib

Figure S13: COX2 rofecoxib

4.2 PTP family



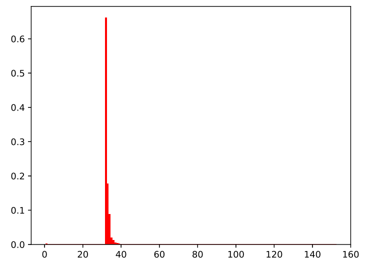
(a) PTP1B



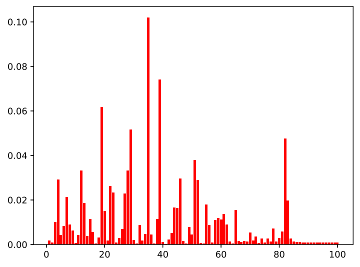
(b) comp1

Figure S14: PTP1B Comp1

4.3 Factor Xa vs. Thrombin



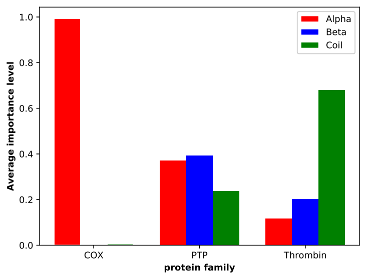
(a) factorXa



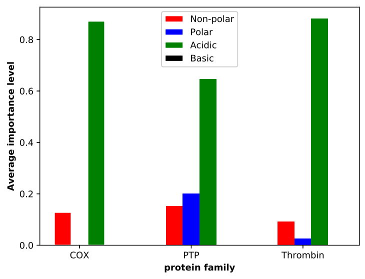
(b) DX9065

Figure S15: factorXa DX9065

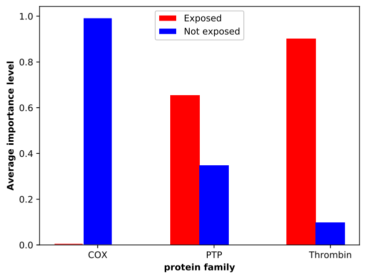
4.4 Marginal distributions of attentions



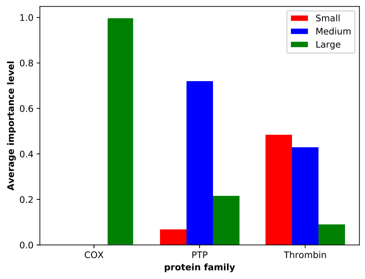
(a) secondary structure type



(b) secondary structure property



(c) secondary structure exposedness



(d) secondary structure length

Figure S16: attention distribution

4.5 Joint (Word) distributions of attentions

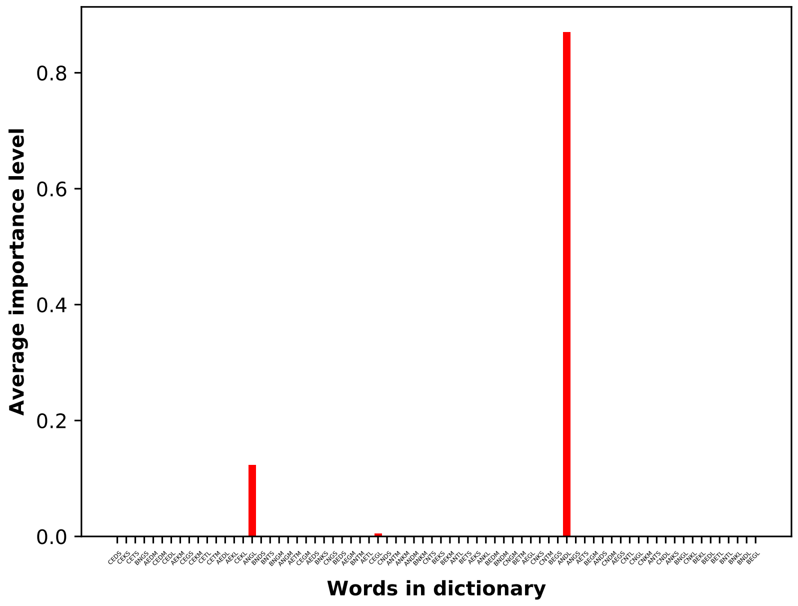


Figure S17: COX

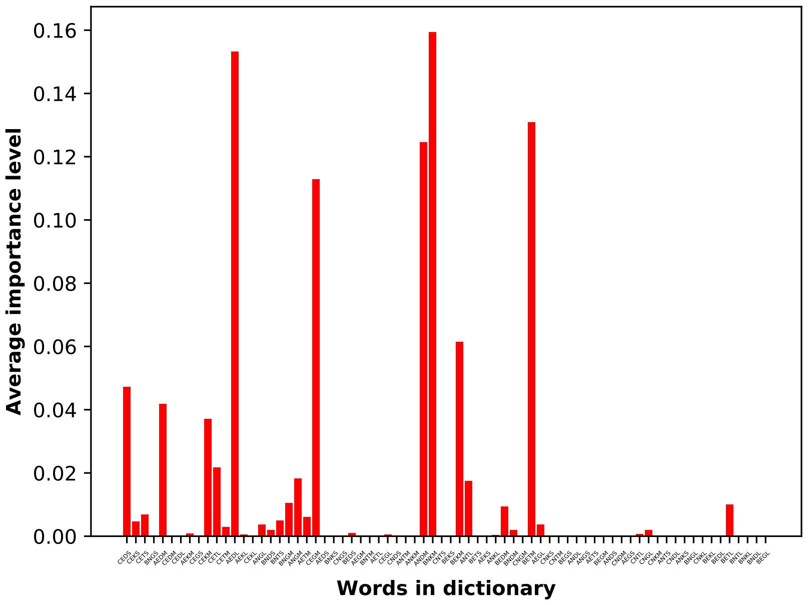


Figure S18: PTP

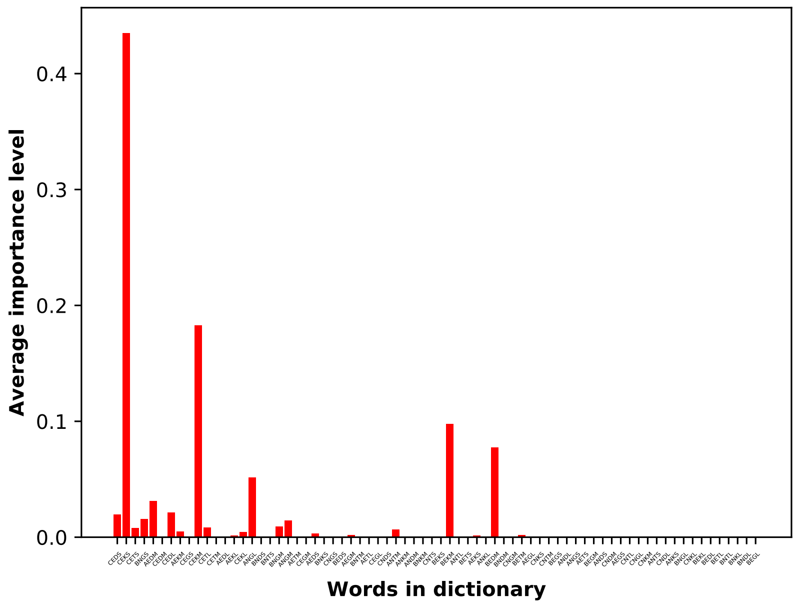


Figure S19: thrombin+factorXa

5 Visualization of attention weights on structural data of compound-protein interactions

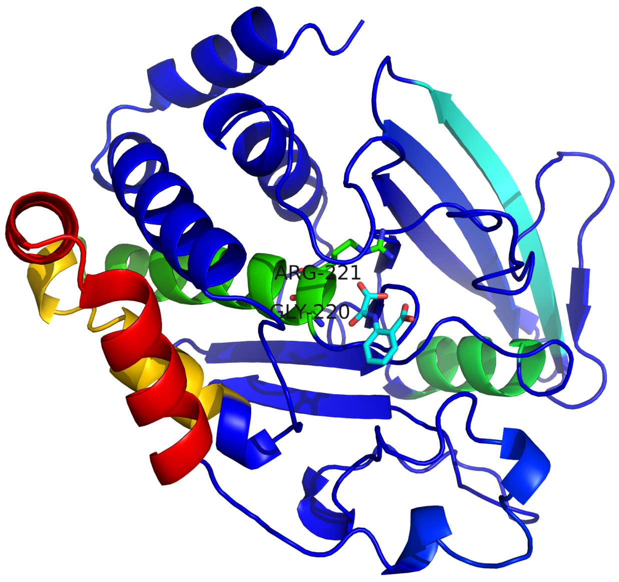


Figure S20: Interpreting deep model for PTP1B binding: 3D structure of PTP1B (cartoon) in complex with OBA (cyan sticks) (PDB ID:1C85) and color-coded by attention scores (warmer colors for higher values).

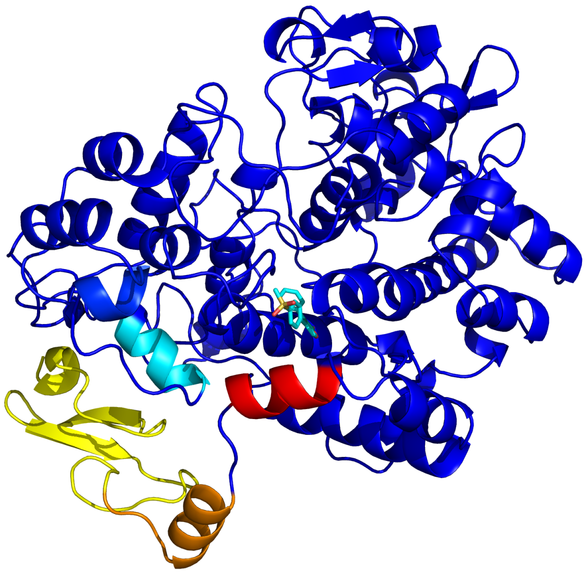


Figure S21: Interpreting deep model for COX2 binding: 3D structure of COX2 (cartoon) in complex with Rofecoxib (cyan sticks) (PDB ID:5KIR) and color-coded by attention scores (warmer colors for higher values).

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., *et al.* (2016). Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Britz, D., Goldie, A., Luong, T., and Le, Q. (2017). Massive Exploration of Neural Machine Translation Architectures. *ArXiv e-prints*.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Khomenko, V., Shyshkov, O., Radyvonenko, O., and Bokhan, K. (2016). Accelerating recurrent neural network training using sequence bucketing and multi-gpu data parallelization. In *Data Stream Mining & Processing (DSMP), IEEE First International Conference on*, pages 100–103. IEEE.

Lawrence, S., Giles, C. L., Tsoi, A. C., and Back, A. D. (1997). Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, **8**(1), 98–113.

Luo, Y., Zhao, X., Zhou, J., Yang, J., Zhang, Y., Kuang, W., Peng, J., Chen, L., and Zeng, J. (2017). A network integration approach for drug-target interaction prediction and computational drug repositioning from heterogeneous information. *Nature communications*, **8**(1), 573.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3.

O'Boyle, N. M., Banck, M., James, C. A., Morley, C., Vandermeersch, T., and Hutchison, G. R. (2011). Open babel: An open chemical toolbox. *Journal of cheminformatics*, **3**(1), 33.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, **12**, 2825–2830.

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, **45**(11), 2673–2681.

Tang, Y. (2016). Tf. learn: Tensorflow's high-level module for distributed machine learning. *arXiv preprint arXiv:1612.04251*.