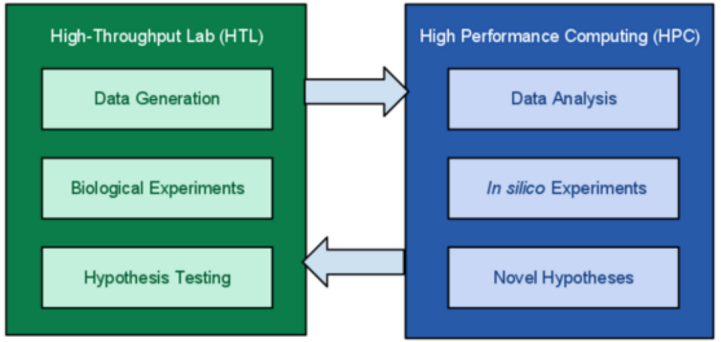
Follow along:
github.com/DOE-NCI-Pilot1/NIH.AI-Deep-Learning-Tutorial

Argonne NATIONAL LABORATORY

FROM IN-VITRO PANELS TO HIGH-THROUGHPUT VIRTUAL SCREENING



AUSTIN CLYDE
aclyde@{uchicago.edu, anl.gov}

Computational Science, ANL
Ph.D. student, University of Chicago

October 23, 2019
NIH.AI Workshop on Applications of Machine Learning for Next Generation Sequencing & Drug Data

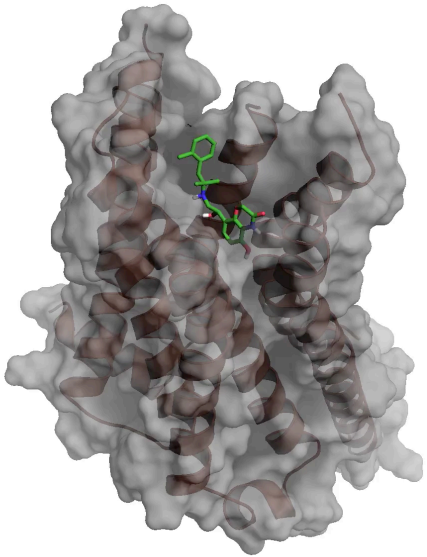
Everyone's familiar with the premise to computational docking

Good

- Very fast
  - Less than a second for multiple conformers and multiple alignments
  - Scoring function (embarrassingly parallel)
- Uniform scoring for various ligands

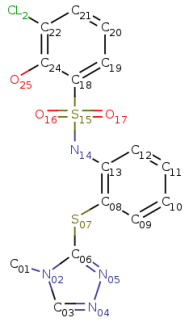
Not so good

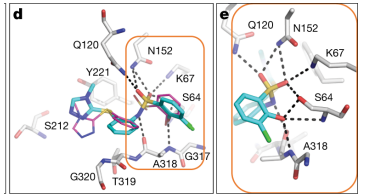
- The actual score is questionably related to free energy
- Not very accurate if your metric is how accurate the pose is

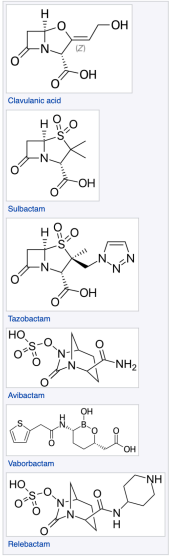


Need for high throughput virtual methods

- Assays on PubChem:
  - 2.1M
- Can buy today:
  - <10 Billion
- Enumerated
  - <100 Billion
- 10^{60} estimated drug like compounds.







LETTER

https://doi.org/10.1038/s41586-019-1540-5

Anthropogenic biases in chemical reaction data hinder exploratory inorganic synthesis

Xiwen Jia[1], Allyson Lynch[1], Yuheng Huang[1], Matthew Danielson[1], Immaculate Lang'at[1], Alexander Milder[1], Aaron E. Ruby[1], Hao Wang[1], Sorelle A. Friedler[2]*, Alexander J. Norquist[1]* & Joshua Schrier[1,3]*

"Machine-learning models that we train on a smaller randomized reaction dataset outperform models trained on larger human-selected reaction datasets, demonstrating the importance of identifying and addressing anthropogenic biases in scientific data."

ARTICLE

https://doi.org/10.1038/s41586-019-0917-9

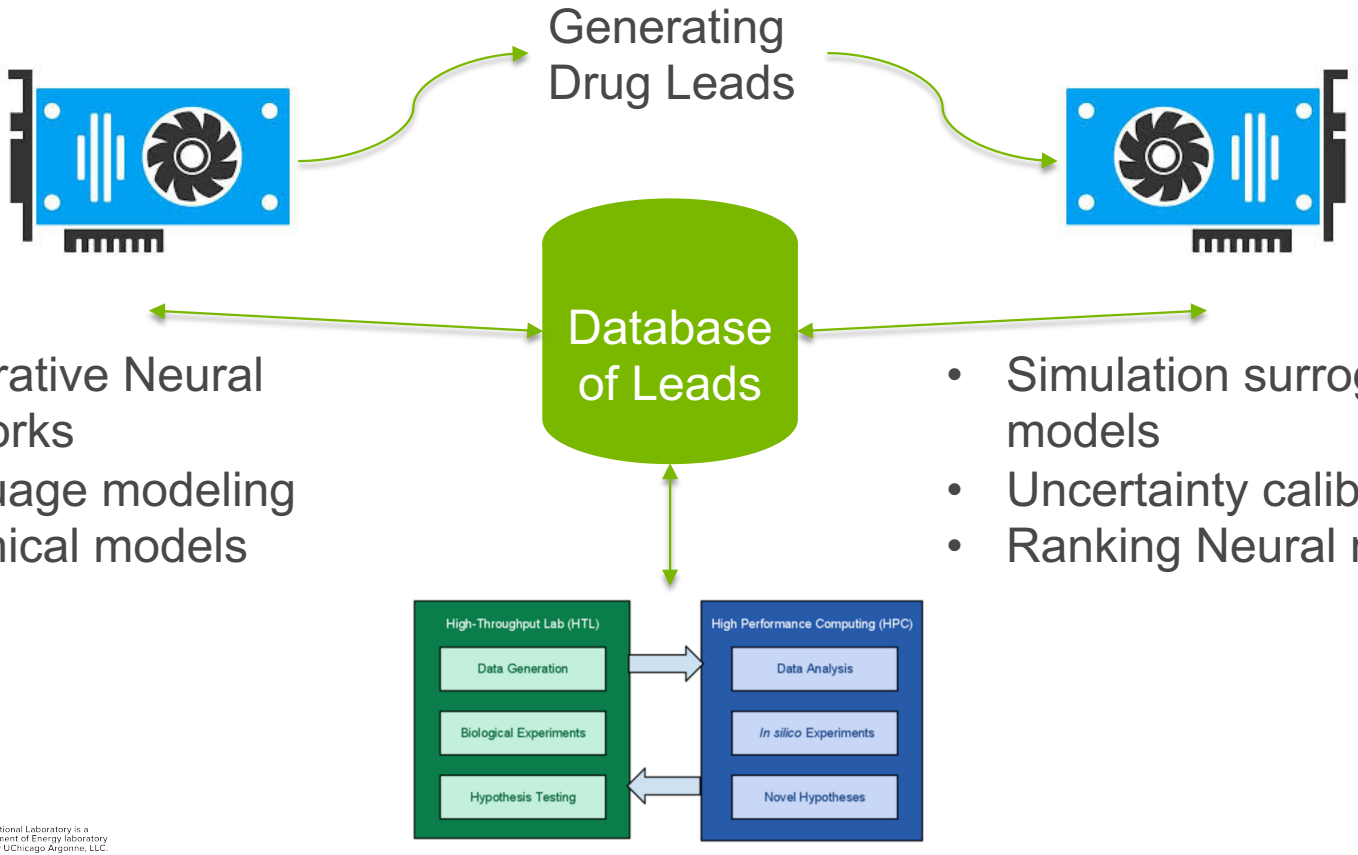
Ultra-large library docking for discovering new chemotypes

Jiankun Lyu[1,2,10], Sheng Wang[3,4,10], Trent E. Balius[1,10], Isha Singh[1,10], Anat Levit[1], Yurii S. Moroz[5,6], Matthew J. O'Meara[1], Tao Che[4], Enkhjargal Algaa[1], Kateryna Tolmachova[7], Andrey A. Tolmachev[7], Brian K. Shoichet[1]*, Bryan L. Roth[4,8,9]* & John J. Irwin[1]*

Of 81 new chemotypes discovered, 30 showed submicromolar activity, including a 180-pM subtype-selective agonist of the D4 dopamine receptor.

DRUG DISCOVERY

HIGH THROUGHPUT SCREENING



- Generative Neural Networks
- Language modeling
- Graphical models

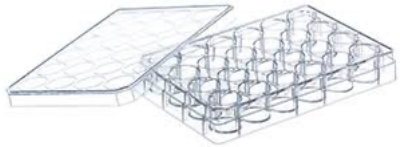
- Simulation surrogate models
- Uncertainty calibrated
- Ranking Neural networks

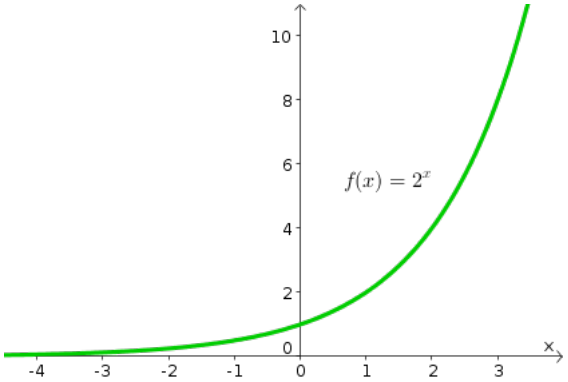
From panels to virtual screening

Imagine a simple panel for a phenotype:

$$\text{phenotype}_x = \text{Expert Analysis of } x$$

$$\text{phenotype}_x = Ax + b$$



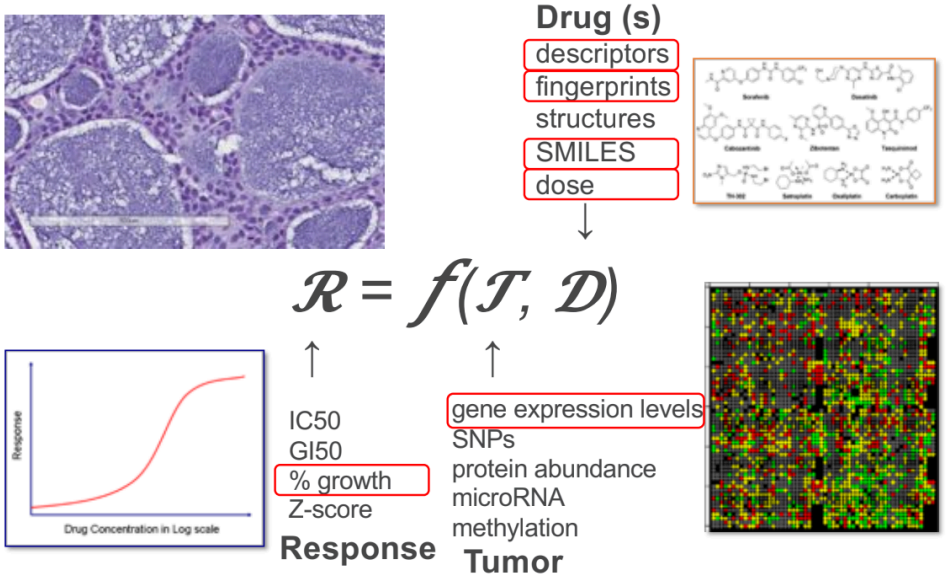


CANCER CELL LINE SCREENS

- In-vitro panels are samples from $\mathcal{R} = f_{\text{true}}(\mathcal{T}, \mathcal{D})$

- We aim to model this function continuously

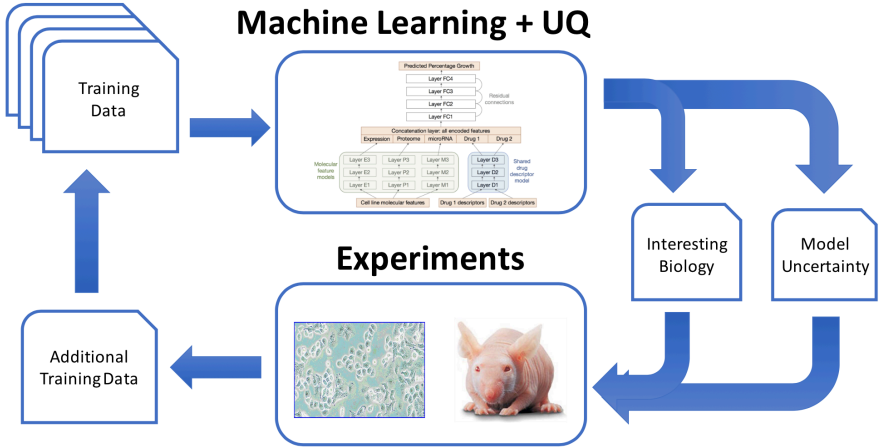
$$\widehat{\mathcal{R}} = \widehat{f}_\theta(\mathcal{T}, \mathcal{D})$$

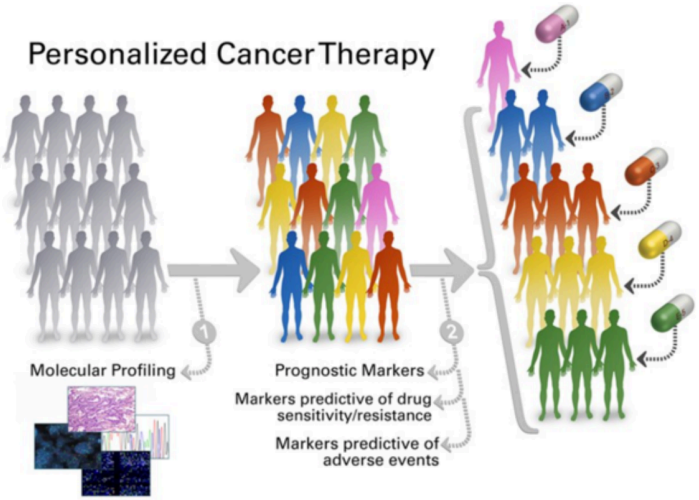


DEEP LEARNING & PRECISION MEDICINE

Understanding molecular structure and genetics

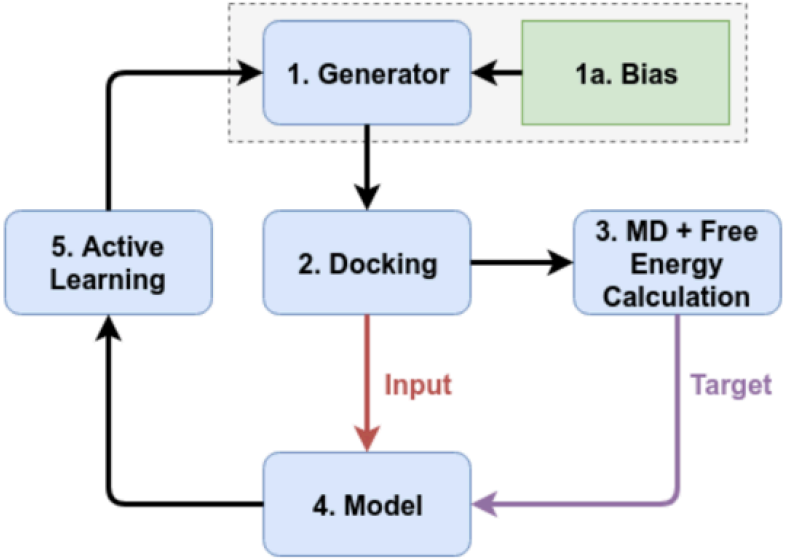
- Remember GWAS studies? We're approx. Chemical Space Wide-GWAS





(why?)

Hypothesis Generation:




```
for molecule in EnamineReal:
        for cell in NCI60.cells():
                if cell.type() == 'melanoma':
                        prediction = f(cell, molecule)
                        if prediction < THRESH:
                                lab.run(molecule, cell)
```

QUESTIONS?

FROM PANEL TO VIRTUAL SCREEN

Data is available at DOE-NCI-GITHUB

Target data:

- Growth response
- Binding Affinity
- Cell death
- Etc.

Machine learning algorithm:

- May have parameters ”theta”

$$\hat{\mathcal{R}} = \hat{f}_\theta(\mathcal{T}, \mathcal{D})$$

Tumor data:

- Cell Name
- Type
- RNA-seq
- SNPs

Drug Data:

- Drug name
- Molecular properties
- Fingerprints
- Formula
- SMILES

TARGET DATA

$$\widehat{\mathcal{R}} = \widehat{f}_\theta(\mathcal{T}, \mathcal{D})$$

	CCLE	NCI60	CTRP	GDSC	gCSI
Samples	11.670	3,780,148	395,263	225,480	6,455
Cells	504	59	887	1,075	409
Drugs	24	52,671	554	249	16
%	0.3%	82.7%	8.8%	5%	0.1%

	SOURCE	CELL	DRUG	STUDY	AUC	IC50	EC50	EC50se	R2fit	Einf	HS	AAC1	AUC1	DSS1
2724943	NCI60	NCI60.NCI-H522	NSC.632899	9011NS77	0.9423	NaN	4.704	0.6647	0.9944	0.5198	1.6880	0.0865	0.9135	0.0590
112232	CTRP	CTRP.HCC-1833	CTRP.321	323204	0.9972	3.545	3.545	0.0802	0.9783	0.0000	2.2700	0.0688	0.9312	0.0545
3031980	NCI60	NCI60.OVCAR-8	NSC.722829	0202NS56	0.9726	4.136	4.136	9460.0000	1.0000	0.0000	4.0000	0.0410	0.9590	0.0319
827560	NCI60	NCI60.A549	NSC.773177	1307NS27	0.9034	NaN	5.238	0.1082	0.9950	0.5318	2.0960	0.1450	0.8550	0.1127
3969399	NCI60	NCI60.SW-620	NSC.617287	9309SR64	0.9578	NaN	7.519	0.3027	0.8897	0.9280	0.9032	0.0607	0.9393	0.0000

FEATURE DATA

$$\widehat{\mathcal{R}} = \widehat{f}_\theta(\mathcal{T}, \mathcal{D})$$

1. Depending on ML method, feature data should be scaled
   i. Deep learning should be scaled to [0,1]
2. Missing feature SAMPLES are ok, but feature columns should be imputed or removed

Example RNA-seq Feature Frame

	Sample	AARS	ABCB6	ABCC5	ABCF1	ABCF3	ABHD4	ABHD6	ABL1	ACAA1	...
0	CCLE.22RV1	8.31	7.17	4.12	5.64	6.04	3.94	2.08	5.24	5.23	...
1	CCLE.2313287	8.94	6.30	3.83	6.60	5.99	6.34	3.72	4.67	5.78	...
2	CCLE.253J	7.58	6.53	3.59	5.94	5.77	5.93	2.35	4.84	4.50	...
3	CCLE.253JBV	7.79	6.01	4.05	6.44	5.97	5.58	2.89	5.09	4.39	...
4	CCLE.42MGBA	7.84	6.72	3.09	6.92	5.43	5.38	3.99	5.85	5.17	...

5 rows × 943 columns

Example Molecular Descriptor Frame

	NAME	ABC	ABCGG	nAcid	nBase	SpAbs_A	SpMax_A	SpDiam_A	SpAD_A	SpMAD_A	...
0	CCLE.18	48.9918	38.3185	0	0	78.4273	2.72997	5.45994	78.4273	1.26496	...
1	CCLE.14	24.9674	18.9098	0	1	40.2912	2.60670	5.14670	40.2912	1.29971	...
2	CCLE.13	34.5673	23.3780	0	1	57.5460	2.61744	5.15821	57.5460	1.33828	...
3	CCLE.24	21.7990	16.5705	0	0	37.7352	2.44674	4.89349	37.7352	1.30121	...
4	CCLE.5	31.6465	21.3953	0	1	51.1869	2.44602	4.87551	51.1869	1.27967	...

Let’s make a few assumptions about these panels:

1. Your data is dose independent
   a. (if it’s not, just add a dose column)
2. You have precomputed ”features” or labels for columns
3. You have two populations of non-comparable things you want to mix
   a. Cells and drugs
   b. Drugs and proteins
   c. Proteins and cells (?)

QUESTIONS?

ML FIRST APPROX

$\mathcal{R}$ $\mathcal{T}$ $\mathcal{D}$

>4,000,000 by 3, where our target, y, is AUC

$N_d \times \mid F_d \mid$

	CELL	DRUG	AUC
0	CCLE.1321N1	CCLE.1	0.8330
1	CCLE.1321N1	CCLE.10	0.7909
2	CCLE.1321N1	CCLE.11	0.5255
3	CCLE.1321N1	CCLE.12	0.8532
4	CCLE.1321N1	CCLE.14	0.5688

	CELL	FEATURE
0	CCLE.MFE280	CCLE.MFE280
1	CTRP.KASUMI-1	CTRP.KASUMI-1
2	gCSI.SU-86-86	gCSI.SU-86-86
3	CTRP.BT139	CTRP.BT139
4	CTRP.HEC-251	CTRP.HEC-251

	DRUG	FEATURE
0	NSC.641536	NSC.641536
1	NSC.689732	NSC.689732
2	NSC.153365	NSC.153365
3	NSC.626117	NSC.626117
4	NSC.711897	NSC.711897

$$\widehat{\mathcal{R}} = \widehat{f}_\theta(\mathcal{T}, \mathcal{D})$$

This is the simplest approach to featurizing, we will assign a ordinal number to each feature to represent it for the algorithm

“y”
Target variable

$\mathcal{R}$ $\mathcal{T}$ $\mathcal{D}$

	AUC	FEATURE_x	FEATURE_y
0	0.8330	CCLE.1321N1	CCLE.1
1	0.7153	CCLE.22RV1	CCLE.1
2	0.8126	CCLE.42MGBA	CCLE.1
3	0.7833	CCLE.5637	CCLE.1
4	0.7675	CCLE.639V	CCLE.1

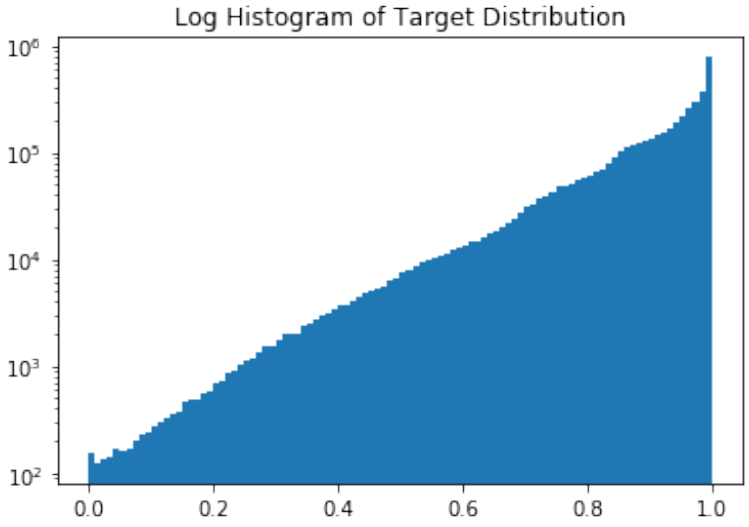
“X”,
Training features

tmp = JOIN(R, T, on='CELL')

JOIN(tmp, D, on='CELL)

Visual inspection

- Extreme data imbalance,
  - If we bin at 0.5,only 2% of data is in the positive class
- Will this matter?



Regression

Linear Regression

```
cv = sklearn.model_selection.KFold(5, random_state=42)
lin_avg_r2 = Avg()
for i, (train, test) in enumerate(cv.split(X,y)):
          X_train, X_test, y_train, y_test = X[train], X[test], y[train], y[test]
          lr = sklearn.linear_model.LinearRegression()
          lr.fit(X_train, y_train)
          test_r2 = lr.score(X_test, y_test)
          print("Cross fold ", i, ":", test_r2)
          lin_avg_r2(test_r2)
```

Random Forest Regressor

```
cv = sklearn.model_selection.KFold(5, random_state=42)
lin_avg_r2 = Avg()
for i, (train, test) in enumerate(cv.split(X,y)):
          X_train, X_test, y_train, y_test = X[train], X[test], y[train], y[test]
          lr = sklearn.ensemble.RandomForestRegressor()
          lr.fit(X_train, y_train)
          test_r2 = lr.score(X_test, y_test)
          print("Cross fold ", i, ":", test_r2)
          lin_avg_r2(test_r2)
```

Classification

Linear Classification

```
cv = sklearn.model_selection.StratifiedKFold(5, random_state=42)
lin_avg_r2 = Avg()
for i, (train, test) in enumerate(cv.split(X,y)):
          X_train, X_test, y_train, y_test = X[train], X[test], y[train], y[test]
          lr = sklearn.linear_model.LogisticRegression()
          lr.fit(X_train, y_train)
          test_r2 = lr.score(X_test, y_test)
          print("Cross fold ", i, ":", test_r2)
          lin_avg_r2(test_r2)
```

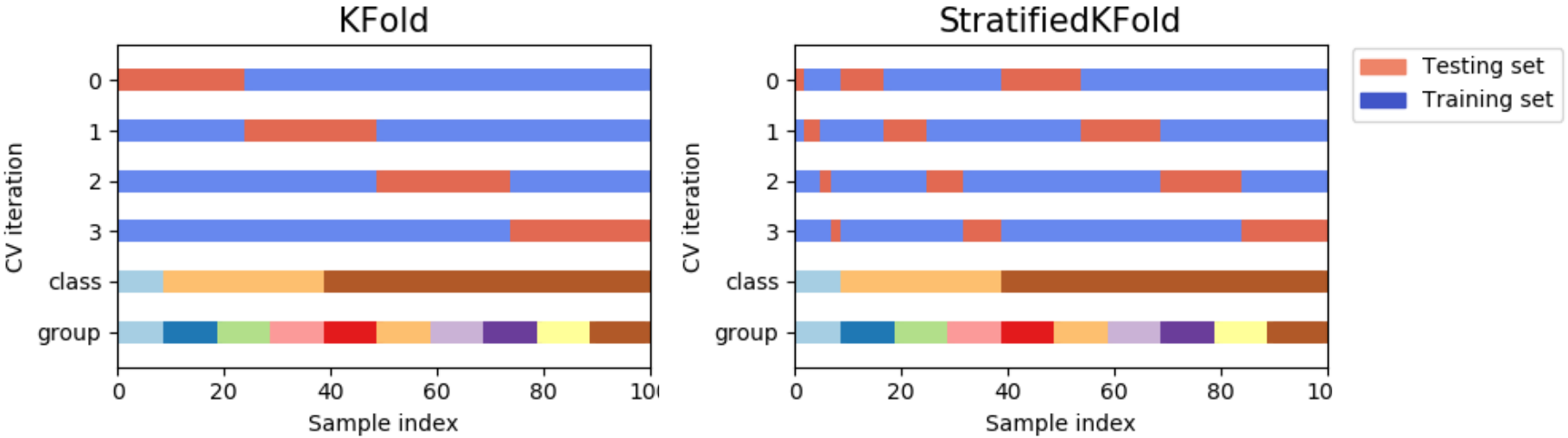
Random Forest Classifier

```
cv = sklearn.model_selection.StratifiedKFold(5, random_state=42)
lin_avg_r2 = Avg()
for i, (train, test) in enumerate(cv.split(X,y)):
          X_train, X_test, y_train, y_test = X[train], X[test], y[train], y[test]
          lr = sklearn.ensemble.RandomForestClassifier()
          lr.fit(X_train, y_train)
          test_r2 = lr.score(X_test, y_test)
          print("Cross fold ", i, ":", test_r2)
          lin_avg_r2(test_r2)
```

```python
import utils as my_utils
from tabulate import tabulate

classif_models = [sklearn.ensemble.RandomForestClassifier, sklearn.linear_model.LogisticRegression]
reg_models = [sklearn.ensemble.RandomForestRegressor, sklearn.linear_model.LinearRegression]

model_perf = {}
for use_binned, problem_type in [(True, classif_models), (False, reg_models)]:
    y_ = (y <= CUTOFF).astype(np.int32) if use_binned else y
    score_func = my_utils.get_bclassif_metrics if use_binned else my_utils.get_regression_metrics
    model_scores = []
    for model in problem_type:
        cv = (sklearn.model_selection.StratifiedKFold if use_binned else sklearn.model_selection.KFold)(2)
        avg_metrics = my_utils.DictAvg()
        for train, test in cv.split(X,y_):
            X_train, X_test, y_train, y_test = X[train], X[test], y_[train], y_[test]
            m = model()
            m.fit(X_train, y_train)
            avg_metrics(score_func(y_test, m.predict(X_test)))
        model_scores.append(avg_metrics)

    model_perf['classif' if use_binned else 'reg'] = model_scores

for key in model_perf.keys():
    print("Model Selection Results, %s:" % key)
    tmp_scores = pd.DataFrame.from_dict([v.avg() for v in model_perf[key]])
    print(tabulate(tmp_scores, tablefmt='psql',headers=tmp_scores.columns))
```

```
Model Selection Results, classif:
+----+----------+----------+----------+-------------+----------+----------+----------+----------+----------+
|    |      acc |   balacc |      mcc |   precision |   recall |       tp |       fp |       tn |       fn |
|----+----------+----------+----------+-------------+----------+----------+----------+----------+----------|
|  0 | 0.690295 | 0.539585 | 0.0657106 |   0.549778  | 0.360415 | 0.360415 | 0.281244 | 0.718756 | 0.718756 |
|  1 | 0.625038 | 0.554821 | 0.0677625 |   0.0506765 | 0.471396 | 0.471396 | 0.361755 | 0.638245 | 0.638245 |
+----+----------+----------+----------+-------------+----------+----------+----------+----------+----------+
Model Selection Results, reg:
+----+-----------+----------+----------------------+
|    |        r2 |      mae |   mean_squared_error |
|----+-----------+----------+----------------------|
|  0 | -0.492116 | 0.136996 |             0.037855 |
|  1 | -0.096791 | 0.128003 |             0.0247447 |
+----+-----------+----------+----------------------+
```

VALIDATION


KFold
StratifiedKFold
CV iteration
0
1
2
3
class
group
0
20
40
60
80
100
Sample index
Testing set
Training set

Understand your use case.

Splitter Classes

`model_selection.GroupKFold` ([n_splits])	K-fold iterator variant with non-overlapping groups.
`model_selection.GroupShuffleSplit` ([…])	Shuffle-Group(s)-Out cross-validation iterator
`model_selection.KFold` ([n_splits, shuffle, …])	K-Folds cross-validator
`model_selection.LeaveOneGroupOut`	Leave One Group Out cross-validator
`model_selection.LeavePGroupsOut` (n_groups)	Leave P Group(s) Out cross-validator
`model_selection.LeaveOneOut`	Leave-One-Out cross-validator
`model_selection.LeavePOut` (p)	Leave-P-Out cross-validator
`model_selection.PredefinedSplit` (test_fold)	Predefined split cross-validator
`model_selection.RepeatedKFold` ([n_splits, …])	Repeated K-Fold cross validator.
`model_selection.RepeatedStratifiedKFold` ([…])	Repeated Stratified K-Fold cross validator.
`model_selection.ShuffleSplit` ([n_splits, …])	Random permutation cross-validator
`model_selection.StratifiedKFold` ([n_splits, …])	Stratified K-Folds cross-validator
`model_selection.StratifiedShuffleSplit` ([…])	Stratified ShuffleSplit cross-validator
`model_selection.TimeSeriesSplit` ([n_splits, …])	Time Series cross-validator

sklearn.model_selection.PredefinedSplit

QUESTIONS?

FEATURIZING A SINGLE INDEPENDENT VARIABLE

Without any continuous variable, the models will not be very good.

- **Single Task Model**—the model takes features to perform a single task.
  - Given a drug, what cells would respond against them? (Precision Medicine, we featurize cells)
  - Given a cell, which drugs would cause a response? (Drug discovery, we featurize drugs)

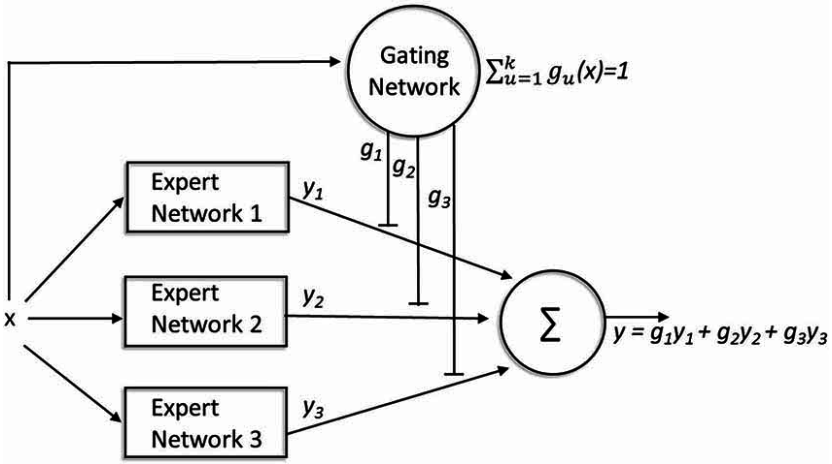
$$\widehat{\mathcal{R}}_{\text{pm}} = \{\widehat{f}_{\theta,d}(\mathcal{T}) \text{ for } d \in \mathcal{D}\}$$

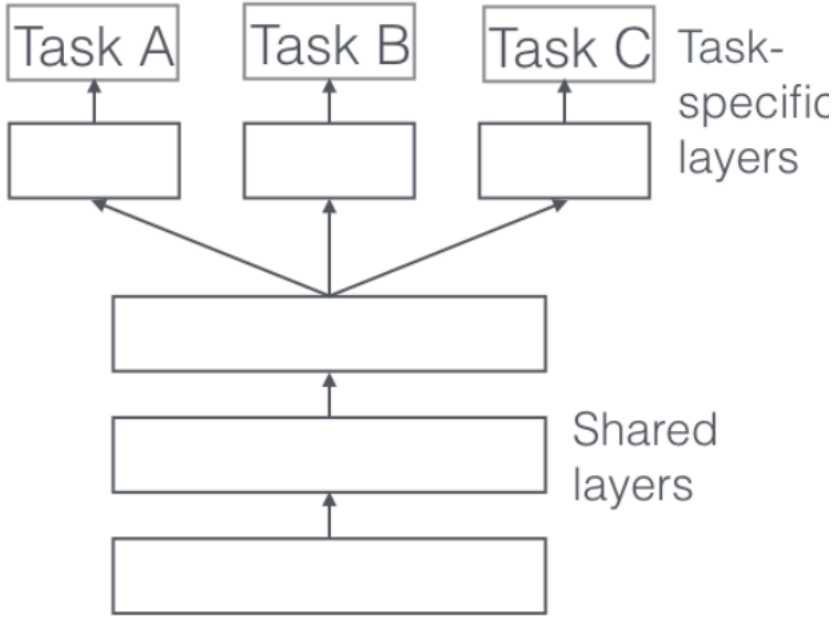
$$\widehat{\mathcal{R}}_{\text{dd}} = \{\widehat{f}_{\theta,t}(\mathcal{D}) \text{ for } t \in \mathcal{T}\}$$

- **Multi Task Model**—The model takes features and attempts to predict for multiple tasks, utilizing some synergy between them
  - Given a handful of drugs, what cells would respond against them?

$$\widehat{\mathcal{R}}_{\text{pm}} = \bigotimes_{d \in \mathcal{D}} \widehat{f}_{\theta,d}(\mathcal{T})$$

$$\widehat{\mathcal{R}}_{\text{dd}} = \bigotimes_{t \in \mathcal{T}} \widehat{f}_{\theta,t}(\mathcal{D})$$


Gating Network
$\sum_{u=1}^{k} g_u(x)=1$
g_1
g_2
g_3
Expert Network 1
Expert Network 2
Expert Network 3
y_1
y_2
y_3
x
Σ
$y = g_1y_1 + g_2y_2 + g_3y_3$


Task A
Task B
Task C
Task-specific layers
Shared layers

Data preparation

- Precision Medicine:
  - We will featurize tumors using a subset of RNA-seq from LINCS1000. It is already scaled.

	Sample	AARS	ABCB6	ABCC5	ABCF1	ABCF3	ABHD4	ABHD6	ABL1	ACAA1	...
0	CCLE.22RV1	0.64360	1.6660	-0.003286	-1.61200	0.4407	-0.6035	-0.6885	-0.2593	0.1775	...
1	CCLE.2313287	1.46500	1.0390	-0.309300	0.02362	0.3325	1.2310	0.9450	-0.9575	0.9307	...
2	CCLE.253J	-0.30830	1.2050	-0.562500	-1.10100	-0.1426	0.9180	-0.4194	-0.7495	-0.8223	...
3	CCLE.253JBV	-0.03450	0.8306	-0.077150	-0.24900	0.2896	0.6504	0.1183	-0.4430	-0.9730	...
4	CCLE.42MGBA	0.03072	1.3420	-1.091000	0.56900	-0.8770	0.4976	1.2140	0.4880	0.0953	...

	AUC	AARS	ABCB6	ABCC5	ABCF1	ABCF3	ABHD4	ABHD6	ABL1	ACAA1	...	FEATURE
0	0.7153	0.64360	1.666	-0.003286	-1.6120	0.4407	-0.6035	-0.6885	-0.2593	0.1775	...	11.0
1	0.8126	0.03072	1.342	-1.091000	0.5690	-0.8770	0.4976	1.2140	0.4880	0.0953	...	6.0
2	0.7833	-2.20000	-1.106	-1.133000	-0.0275	-0.7905	-0.9320	-1.0770	0.9536	-0.1512	...	23.0
3	0.7675	-0.86870	-0.732	0.155000	0.3984	0.6350	-0.7026	0.6360	-0.4062	0.7390	...	11.0
4	0.7692	0.23930	-2.150	-0.056060	0.2622	-1.4170	-2.8360	-0.3398	0.5250	0.3008	...	8.0

```
print(X.shape, y.shape)
```

```
(10971, 943) (10971,)
```

Single Task

Predict a cell line's response to PACLITAXEL

```
Trying regression model:
Avg regression metrics:
+--------------------+---------------------+
|        key         |        value        |
+--------------------+---------------------+
|         r2         | -0.4771972705313562 |
|        mae         |  0.1362578201051367 |
| mean_squared_error | 0.03736409993161219 |
+--------------------+---------------------+
Trying classif model:
Avg classif Metrics
+-----------+----------------------+
|    key    |        value         |
+-----------+----------------------+
|    acc    |  0.8103503485411754  |
|   balacc  |  0.46994857187209726 |
|    mcc    | -0.031176640022032513|
| precision |  0.23141515254034908 |
|   recall  |  0.06514942528735632 |
|     tp    |  0.06514942528735632 |
|     fp    |  0.12525228154316176 |
|     tn    |  0.8747477184568382  |
|     fn    |  0.8747477184568382  |
+-----------+----------------------+
```

Multi-Task

Predict a cell line's response to

PACLITAXEL

ERLOTINIB

NILOTINIB

LAPATINIB

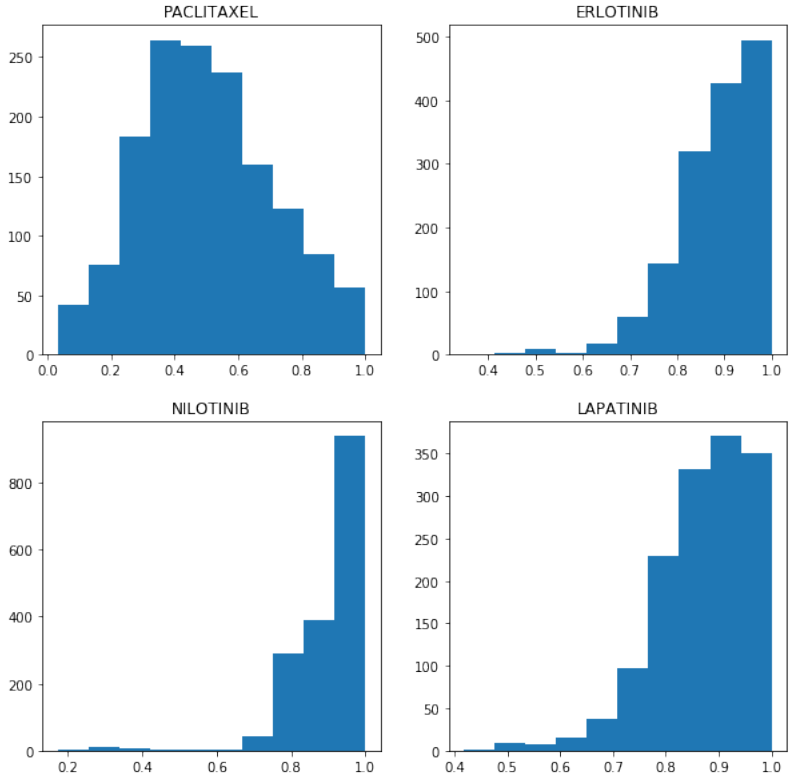
```
Trying regression model:
Avg regression metrics:
+--------------------+---------------------+
|        key         |        value        |
+--------------------+---------------------+
|         r2         |  0.4269324057940695 |
|        mae         | 0.09525294917915436 |
| mean_squared_error | 0.017419461817240262 |
+--------------------+---------------------+
Trying classif model:
Avg classif Metrics
+-----------+--------------------+
|    key    |        value       |
+-----------+--------------------+
|    acc    |  0.895751074566683 |
|   balacc  | 0.7594775728053225 |
|    mcc    |  0.575126667999007 |
| precision | 0.7193455077826594 |
|   recall  | 0.5728750923872875 |
|     tp    | 0.5728750923872875 |
|     fp    | 0.05391994677664254 |
|     tn    | 0.9460800532233575 |
|     fn    | 0.9460800532233575 |
+-----------+--------------------+
```

Wait what?

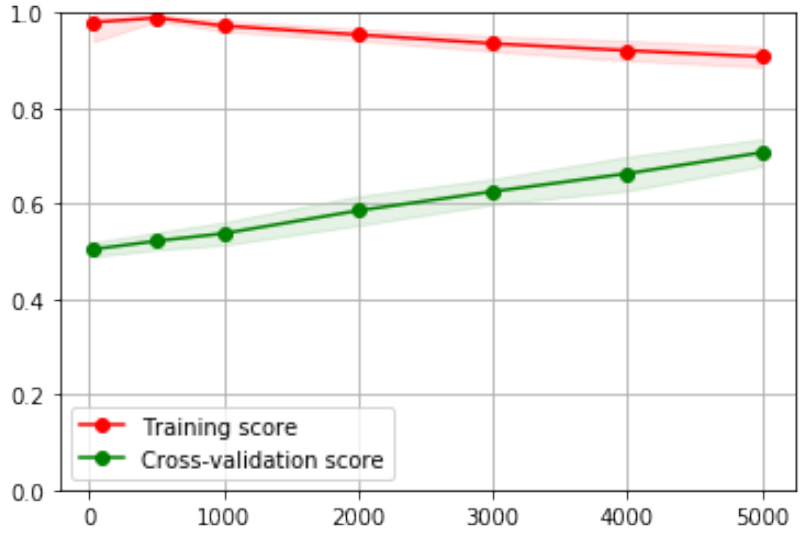
Well did we improve on our PACLITAXEL predictions?

No, we're just good at the others.

```
0.0    0.556701
1.0    0.981735
2.0    1.000000
3.0    0.987288
Name: acc, dtype: float64
```



LEARNING CURVES



What to use? Multi-task or single task?

- Sometimes, using a multi-task model can actually improve the model performance for the single tasks job. It is important to evaluate your model across all viable options to see where you are performing best.
- Also the models presented here are not multi-task in the usual sense. I will get to this, but I'll go out on a limb and argue its useful to think of it this way.

QUESTIONS?

DEEP LEARNING

These are all the same:

Weights to learn:

$$W^{(0)} \in \mathbb{R}^{4,3}, b^{(0)} \in \mathbb{R}^4$$

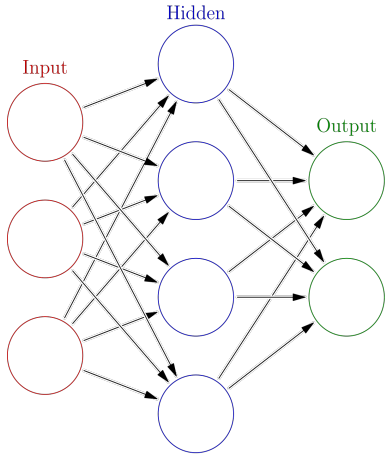
$$W^{(1)} \in \mathbb{R}^{2,4}, b^{(1)} \in \mathbb{R}^2$$

$$I = Input \in \mathbb{R}^3$$

$$H = \text{ReLU}(W^{(0)}I - b^{(0)}) \in \mathbb{R}^4$$

$$O = \text{softmax}(W^{(1)}H - b^{(1)}) \in \mathbb{R}^2$$

```
from keras.models import Model
from keras.layers import Input, Dense

input_layer = Input((3,))
hidden_layer = Dense(4, activation='relu')(input_layer)
output_layer = Dense(2, activation='softmax')(hidden_layer)
model = Model(inputs=input_layer, outputs=output_layer)


model.compile(optimizer='sgd',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(data, labels)
```

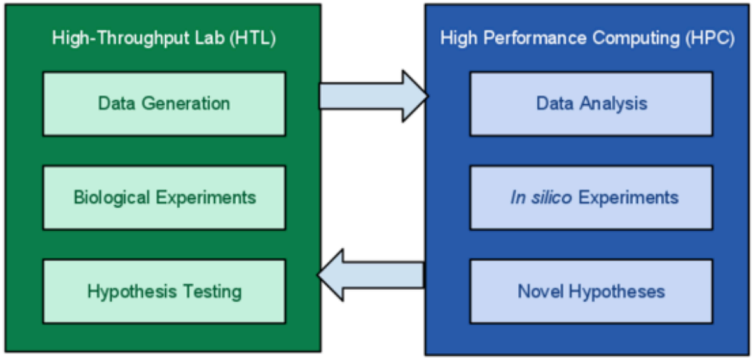


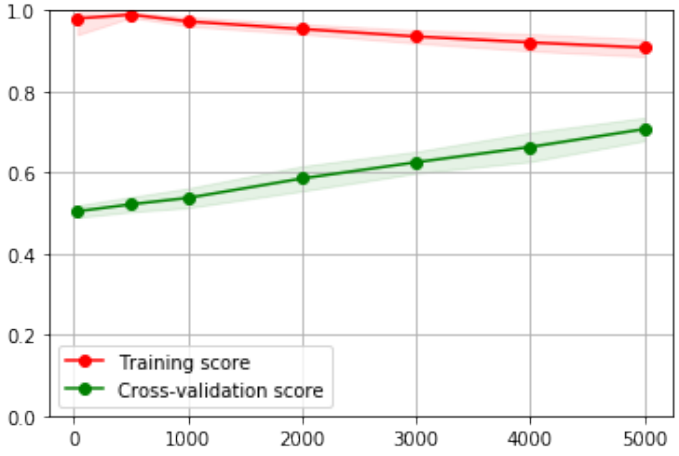
Neural networks are just giant functions!

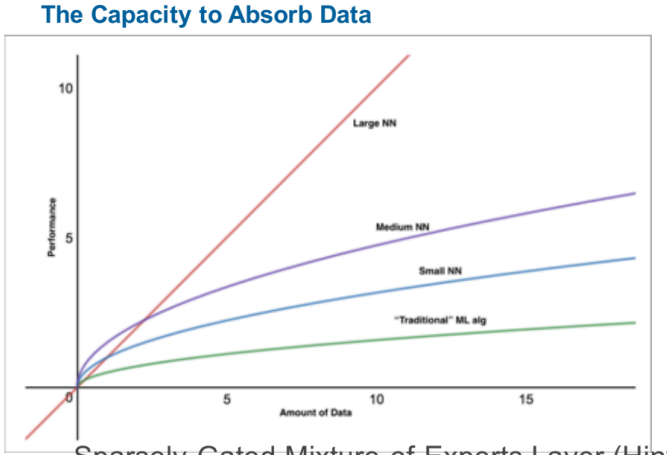
$$O = \text{softmax}\left[W^{(1)}\left[\text{ReLU}(W^{(0)}I - b^{(0)}) \in \mathbb{R}^4\right] - b^{(1)})\right]$$

When to use?

- When your learning curves are saturated







Sparsely-Gated Mixture-of-Experts Layer (Hinton, 2017)
137 billion parameters, thousands of subnetworks
1000x improvements in model capacity

Practically speaking (use Keras)

1. Choice of architecture (stick with simple, and generally is dictated by features)
2. Choice of loss function
3. Choice of optimizer and learning rate strategy
4. Choice of validation metrics, and when to stop training

Open Source Framework Comparison

	Languages	Tutorials and training materials	CNN modeling capability	RNN modeling capability	Architecture: easy-to-use and modular front end	Speed	Multiple GPU support	Keras compatible
Theano	Python, C++	++	++	++	+	++	+	+
Tensor-Flow	Python	+++	+++	++	+++	++	++	+
Torch	Lua, Python (new)	+	+++	++	++	+++	++	
Caffe	C++	+	++		+	+	+	
MXNet	R, Python, Julia, Scala	++	++	+	++	++	+++	+
Neon	Python	+	++	+	+	++	+	
CNTK	C++	+	+	+++	+	++	+	+

Architectures

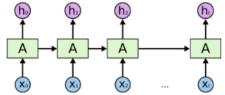
There's a whole lot here.

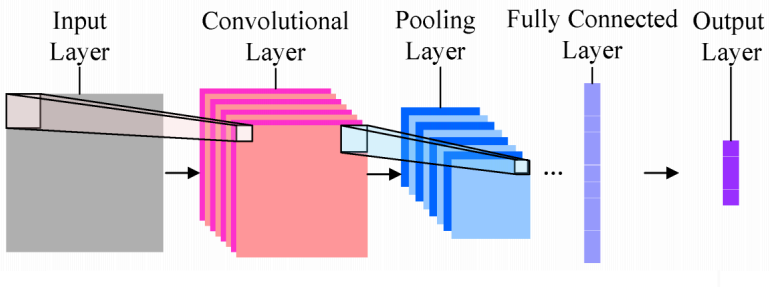
Rule of thumb:

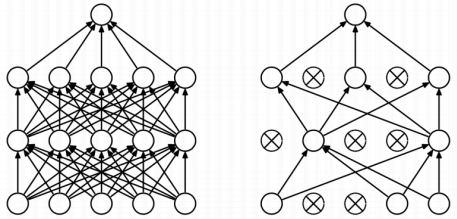
- Start small,
- Decreasing layer width
- Total params < samples
- Start with 3-5 layers

Layer (type)	Output Shape	Param #
conv1d_1 (Conv1D)	(None, 60464, 128)	2688
activation_1 (Activation)	(None, 60464, 128)	0
max_pooling1d_1 (MaxPooling1	(None, 60464, 128)	0
conv1d_2 (Conv1D)	(None, 60455, 128)	163968
activation_2 (Activation)	(None, 60455, 128)	0
max_pooling1d_2 (MaxPooling1	(None, 6045, 128)	0
flatten_1 (Flatten)	(None, 773760)	0
dense_1 (Dense)	(None, 200)	154752200
activation_3 (Activation)	(None, 200)	0
dropout_1 (Dropout)	(None, 200)	0
dense_2 (Dense)	(None, 20)	4020
activation_4 (Activation)	(None, 20)	0
dropout_2 (Dropout)	(None, 20)	0
dense_3 (Dense)	(None, 36)	756
activation_5 (Activation)	(None, 36)	0

Total params: 154,923,632
Trainable params: 154,923,632
Non-trainable params: 0







(a) Standard Neural Net (b) After applying dropout.

Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1+e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

Loss functions

- Few simple facts to get you started:
- When you are doing regression:
  - Your last layer must have size of regression target, probably shouldn't use activation function
- Positive/Negative Class
  - Binary Cross-entropy
  - Do not one hot encode
  - Final layer should have sigmoid activation
- Multiple Classes
  - Cross entropy loss
  - Must one hot encode
  - Final layer must have softmax activation

Regression: $$R(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - f_k(x_i))^2.$$

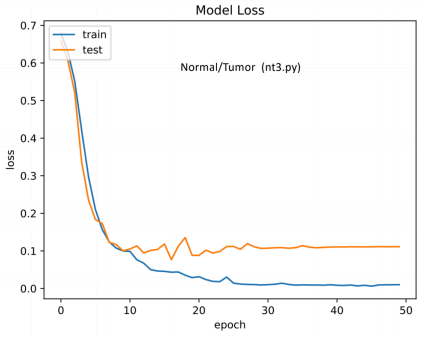
Classification: cross-entropy (deviance)

$$R(\theta) = -\sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log f_k(x_i)$$

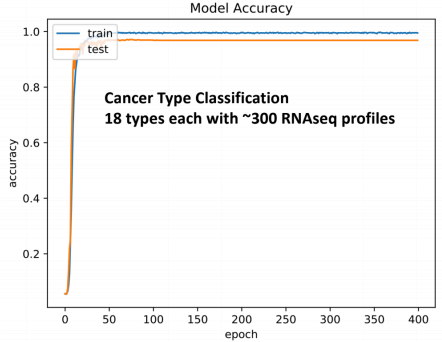
Optimization

$$\Phi^* = \underset{\Phi}{\operatorname{argmin}} \; E_{(x,y)\sim \mathrm{Pop}} \; -\ln \; P_\Phi(y|x)$$

When to stop training

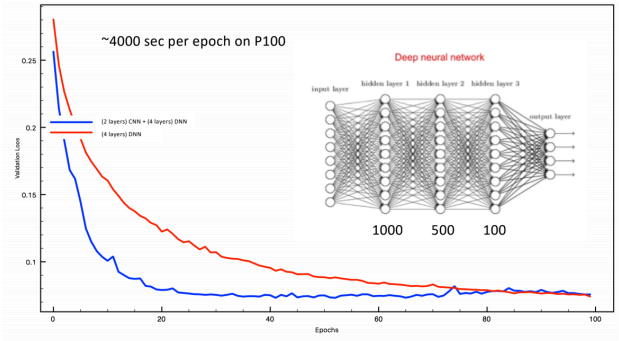


https://github.com/ECP-CANDLE/Benchmarks/tree/frameworks/Pilot1/NT3

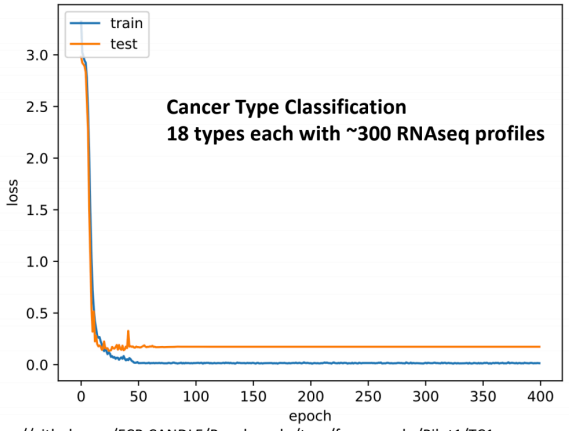


https://github.com/ECP-CANDLE/Benchmarks/tree/frameworks/Pilot1/TC1

P1B3 Convergence
([C(100)xC(50)]x1000x500x100x50)

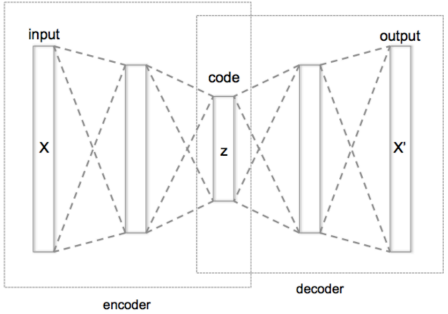


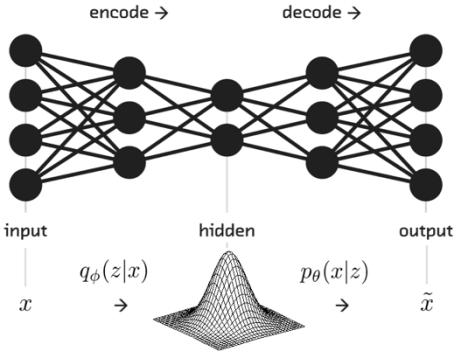
https://github.com/ECP-CANDLE/Benchmarks/tree/frameworks/Pilot1/P1B3



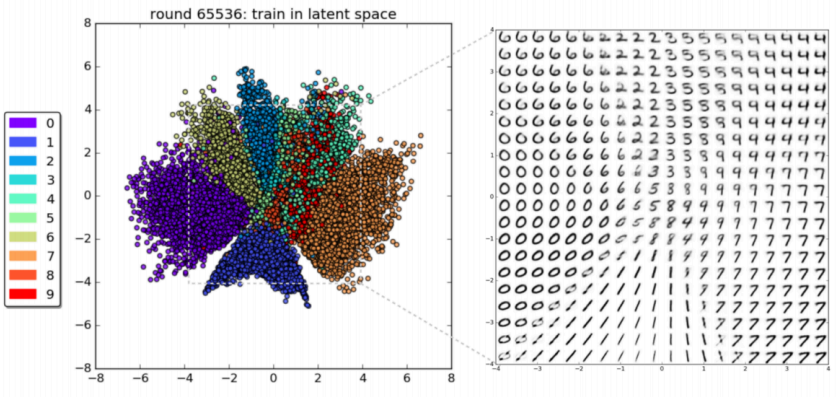
https://github.com/ECP-CANDLE/Benchmarks/tree/frameworks/Pilot1/TC1

Other things besides just predictions….





MNIST Latent Space Sampling



FULLY CONTINUOUS MODELS

Target data:

- Growth response
- Binding Affinity
- Cell death
- Etc.

Machine learning algorithm:

- May have parameters ”theta”

$$\hat{\mathcal{R}} = \hat{f}_\theta(\mathcal{T}, \mathcal{D})$$

Tumor data:

- Cell Name
- Type
- RNA-seq
- SNPs

Drug Data:

- Drug name
- Molecular properties
- Fingerprints
- Formula
- SMILES

Just featurize both things and continue! Seriously

	AUC	AARS	ABCB6	ABCC5	ABCF1	ABCF3	ABHD4	ABHD6	ABL1	ACAA1	...	FEATURE
0	0.7153	0.64360	1.666	-0.003286	-1.6120	0.4407	-0.6035	-0.6885	-0.2593	0.1775	...	11.0
1	0.8126	0.03072	1.342	-1.091000	0.5690	-0.8770	0.4976	1.2140	0.4880	0.0953	...	6.0
2	0.7833	-2.20000	-1.106	-1.133000	-0.0275	-0.7905	-0.9320	-1.0770	0.9536	-0.1512	...	23.0
3	0.7675	-0.86870	-0.732	0.155000	0.3984	0.6350	-0.7026	0.6360	-0.4062	0.7390	...	11.0
4	0.7692	0.23930	-2.150	-0.056060	0.2622	-1.4170	-2.8360	-0.3398	0.5250	0.3008	...	8.0

	DRUG	ABC	ABCGG	nAcid	nBase	SpAbs_A	SpMax_A	SpDiam_A	SpAD_A	SpMAD_A	...
0	CCLE.18	48.9918	38.3185	0	0	78.4273	2.72997	5.45994	78.4273	1.26496	...
1	CCLE.14	24.9674	18.9098	0	1	40.2912	2.60670	5.14670	40.2912	1.29971	...
2	CCLE.13	34.5673	23.3780	0	1	57.5460	2.61744	5.15821	57.5460	1.33828	...
3	CCLE.24	21.7990	16.5705	0	0	37.7352	2.44674	4.89349	37.7352	1.30121	...
4	CCLE.5	31.6465	21.3953	0	1	51.1869	2.44602	4.87551	51.1869	1.27967	...

VALIDATION

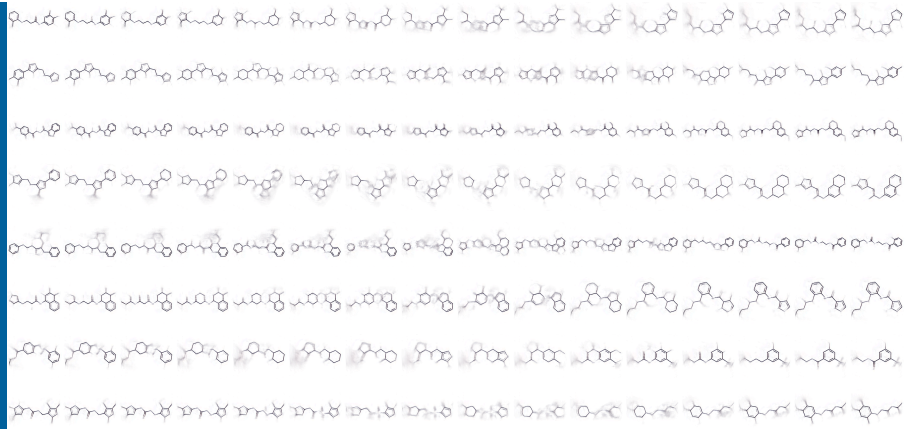
Judge the model by what you want, not by metrics

- By cell
  - Uniqueness
  - Type
  - Source
- By drug
  - Uniqueness
  - Scaffold
  - Time (train on drugs developed before 1970, for example)
- By study
  - If we added a new batch of compounds from another hospital, could we predict on them?

BREAK

Argonne NATIONAL LABORATORY

MOLECULAR VIRTUAL DRUG SCREENING WITH DEEP LEARNING

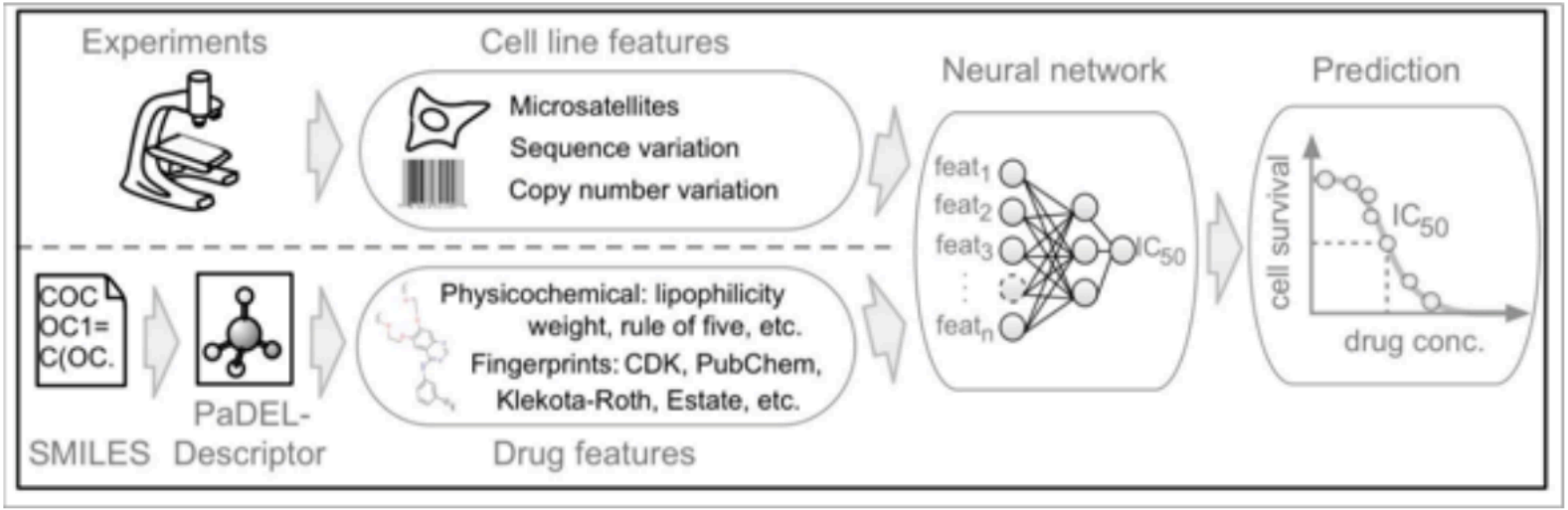


AUSTIN CLYDE

Computational Science, ANL
Ph.D. student, University of Chicago

October 23, 2019
NIH.AI Workshop on Applications of Machine Learning for Next Generation Sequencing & Drug Data

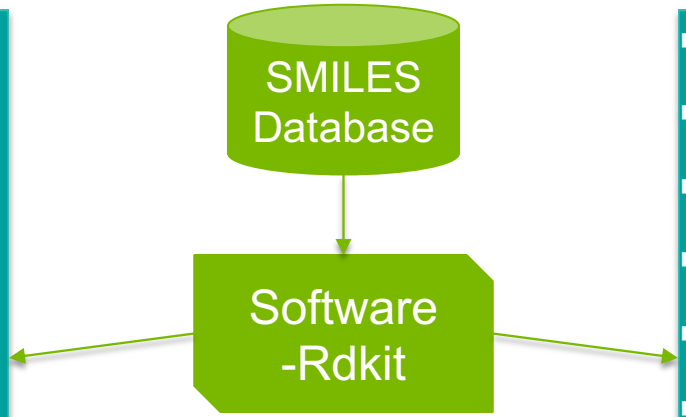
MOLECULES


Experiments
Cell line features
Microsatellites
Sequence variation
Copy number variation
Neural network
Prediction
feat1
feat2
feat3
featn
IC50
cell survival
IC50
drug conc.
COC
OC1=
C(OC.
SMILES
PaDEL-
Descriptor
Physicochemical: lipophilicity
weight, rule of five, etc.
Fingerprints: CDK, PubChem,
Klekota-Roth, Estate, etc.
Drug features

MOLECULAR MODALITIES

What *is* a “molecule” in the sense we’re after?

- 2D Graphs
- 3D Coordinates
- 2D Images
- 3D Images (voxels)
- SMILES
  - Canonical
  - Kekule
- SELFIES (L2 Chomsky)
- Surface
- Conformer Sets



how do we chose?

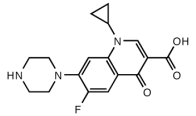
- CFG models
- Docked models
- Chemical descriptors
- Fingerprints
- 3D waveform
- 3D density
- MACCS
- …many more in literature

SMILES

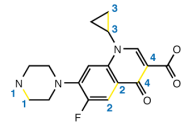
Simplified Molecular Input Line Entry System

- DFS on graph
  - The chemical graph is first trimmed to remove hydrogen atoms and cycles are broken to turn it into a spanning tree.
  - Where cycles have been broken, numeric suffix labels are included to indicate the connected nodes. Parentheses are used to indicate points of branching on the tree.

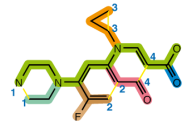
A



B



C

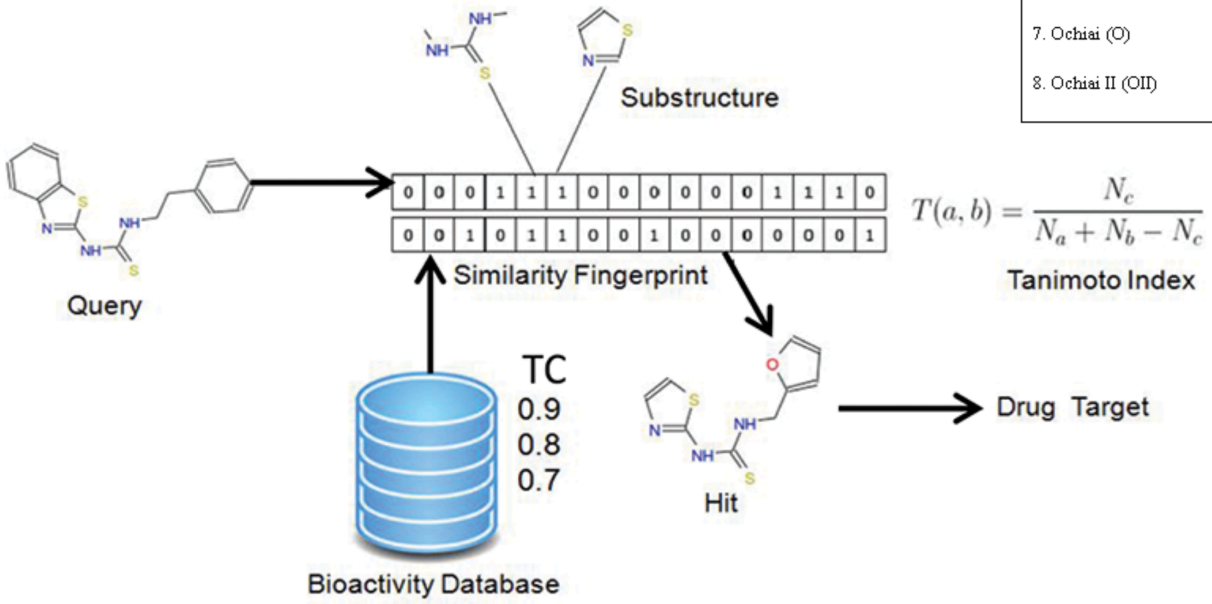


D

N1CCN(CC1)C(C(F)=C2)=CC(=C2C4=O)N(C3CC3)C=C4C(=O)O

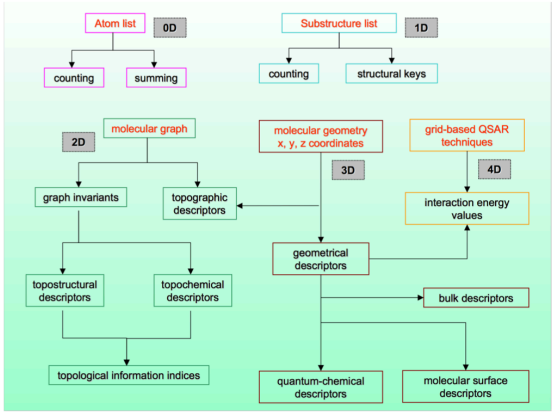
FINGERPRINTS

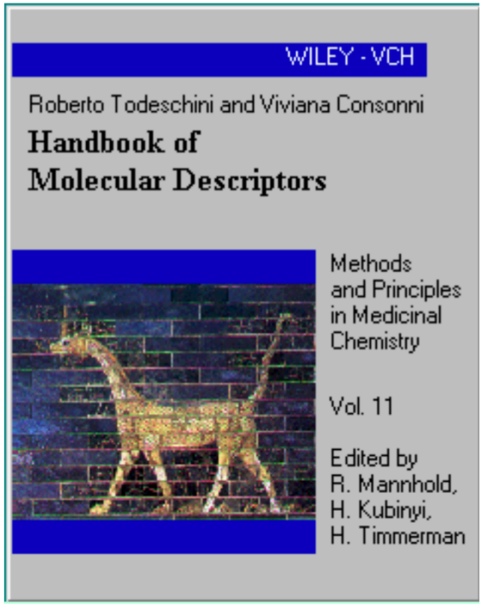
Coefficients	Similarity expression	Source
1. Simple matching (SM)	$\frac{a+d}{a+b+c+d}$	Sokal and Michener, 1958
2. Rogers and Tanimoto (RT)	$\frac{a+d}{a+2b+2c+d}$	Rogers and Tanimoto, 1960
3. Anderberg (A)	$\frac{a}{a+2(b+c)}$	Anderberg, 1973
4. Russel and Rao (RR)	$\frac{a}{a+b+c+d}$	Russel and Rao, 1940
5. Jaccard (J)	$\frac{a}{a+b+c}$	Jaccard, 1901
6. Sorensen-Dice (SD)	$\frac{2a}{2a+b+c}$	Dice, 1945; Sorensen, 1948
7. Ochiai (O)	$\frac{a}{\sqrt{(a+b)(a+c)}}$	Ochiai, 1957
8. Ochiai II (OII)	$\frac{ad}{\sqrt{(a+b)(a+c)(d+b)(d+c)}}$	Ochiai, 1957



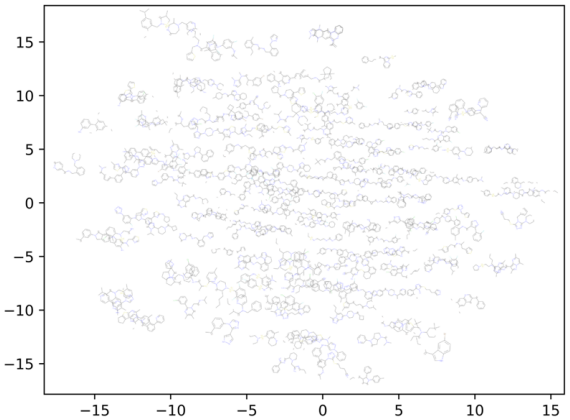
DESCRIPTORS

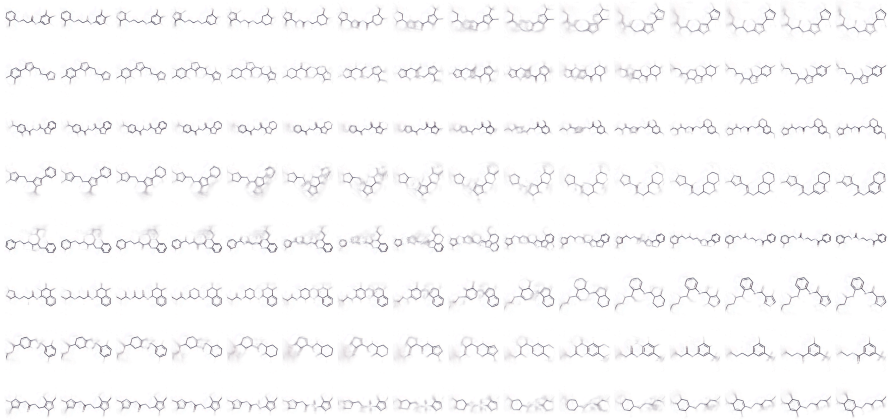
"The molecular descriptor is the final result of a logic and mathematical procedure which transforms chemical information encoded within a symbolic representation of a molecule into a useful number or the result of some standardized experiment."
R. Todeschini and V. Consonni





USING IMAGES





Convolutional **networks** on **graphs** for **learning molecular fingerprints**

DK Duvenaud, D Maclaurin, J Iparraguirre… - Advances in **neural** …, 2015 - papers.nips.cc

… **Figure** 5: Visualizing **fingerprints** optimized for predicting toxicity … The **neural fingerprint**, when viewed in this light, resembles an unrolled message-passing algorithm on the original **graph**. 7 Conclusion … **Neural network** for **graphs**: A contextual constructive approach …

☆ Cited by 803 Related articles All 12 versions »

[HTML] **Molecular graph** convolutions: moving beyond **fingerprints**

S Kearnes, K McCloskey, M Berndl, V Pande… - … -aided **molecular** design, 2016 - Springer

… Since **molecules** are undirected **graphs**, we will also maintain the following: Property 3 … **Figure** 8 gives examples of how the initial atom features for a single **molecule** (ibuprofen) evolve as they progress through **graph** convolution Weave modules …

☆ Cited by 347 Related articles All 16 versions Web of Science: 133 »

Atomic convolutional **networks** for predicting protein-ligand binding affinity

J Gomes, B Ramsundar, EN Feinberg… - arXiv preprint arXiv …, 2017 - arxiv.org

… **Figure** 2. **Diagram** of Atomic Convolutions on Protein Ligand Systems … create three weight-sharing, replica **networks**, one each for complex, protein, and ligand (**Figure** 2). The … Descriptions of the **graph** convolutional models and model hyperparameters are given elsewhere in the …

☆ Cited by 61 Related articles All 3 versions »

[HTML] The rise of **deep learning** in drug discovery

H Chen, O Engkvist, Y Wang, M Olivecrona… - Drug discovery today, 2018 - Elsevier

… Besides the **graph**-based representation **learning** methods, DL methods based on other types of … The upper **plot** shows how the RNN model thinks when generating the structure on the … The bottom left **figure** demonstrates how the RNN actually works in the structure-generation …

☆ Cited by 189 Related articles All 5 versions Web of Science: 95

Learning to smile (s)

S Jastrzębski, D Leśniak, WM Czarnecki - arXiv preprint arXiv:1602.06289, 2016 - arxiv.org

… **Figure** 3: Visualization of CNN filters of size 5 for ac- tive (top row) and inactives **molecules**. 2 EXPERIMENTS … Convolutional **networks** on **graphs** for **learning molecular fingerprints**. CoRR, abs/1509.09292, 2015 … **Deep** convolutional **networks** on **graph**-structured data …

☆ Cited by 24 Related articles All 10 versions »

Seq2seq **fingerprint**: An unsupervised **deep molecular** embedding for drug discovery

Z Xu, S Wang, F Zhu, J Huang - … of the 8th ACM International Conference …, 2017 - dl.acm.org

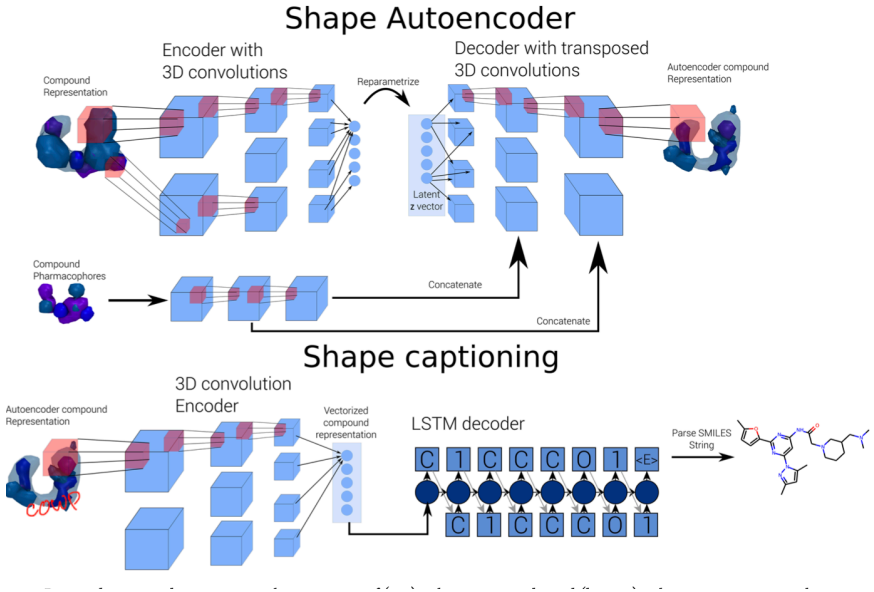
… The **neural fingerprint** is constructed on a supervised **deep graph** convolutional **neural network** … show the impact of seq2seq **fingerprint** length on the accuracy in **Figure** 5. From … both data sets, our meth- ods significantly outperform the circular and **neural fingerprints**, regardless of …

☆ Cited by 29 Related articles All 2 versions »

DEEP LEARNING WITH MOLECULES

PROPERTY PREDICTIONS

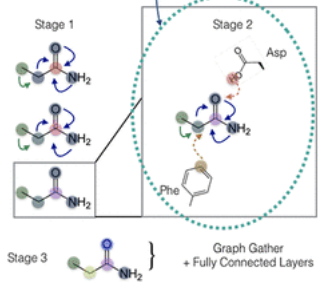
Images, 3D surfaces

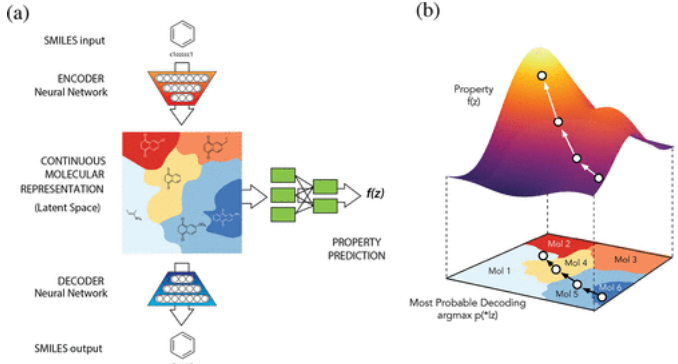


Skalic, Miha, et al. "Shape-Based Generative Modeling for de Novo Drug Design." *Journal of chemical information and modeling* 59.3 (2019): 1205-1214.

$$A = \left(\begin{bmatrix} A_{111} & A_{121} & \cdots & A_{1N1} \\ A_{211} & A_{221} & \cdots & A_{2N1} \\ \vdots & \vdots & \ddots & \vdots \\ A_{N11} & A_{N21} & \cdots & A_{NN1} \end{bmatrix}, \cdots, \begin{bmatrix} A_{11N_{et}} & A_{12N_{et}} & \cdots & A_{1NN_{et}} \\ A_{21N_{et}} & A_{22N_{et}} & \cdots & A_{2NN_{et}} \\ \vdots & \vdots & \ddots & \vdots \\ A_{N1N_{et}} & A_{N2N_{et}} & \cdots & A_{NNN_{et}} \end{bmatrix} \right) \in \mathbb{R}^{N \times N \times N_{et}}, \text{where: } A_{ijk} = \begin{cases} 1, v_j \in N(v_i) \text{ and } e_{i,j} = k \\ 0, \text{otherwise.} \end{cases}$$

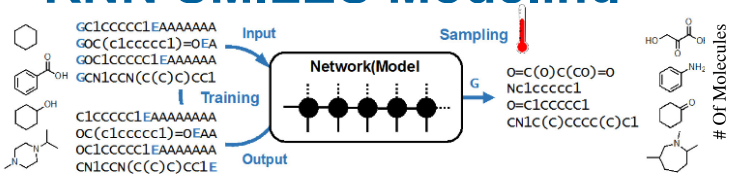
Feinberg, Evan N., et al. "Potentialnet for molecular property prediction." *ACS central science* 4.11 (2018): 1520-1530.



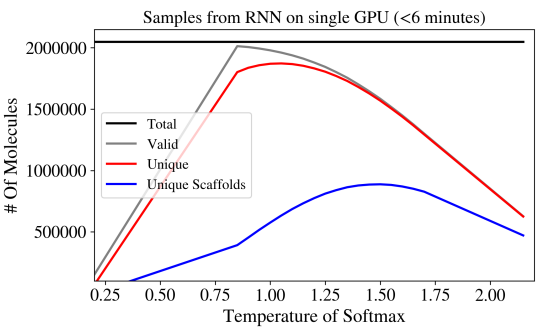


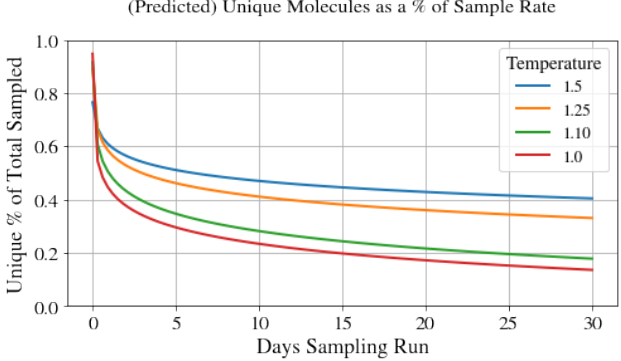
Gómez-Bombarelli, Rafael, et al. "Automatic chemical design using a data-driven continuous representation of molecules." ACS central science 4.2 (2018): 268-276.

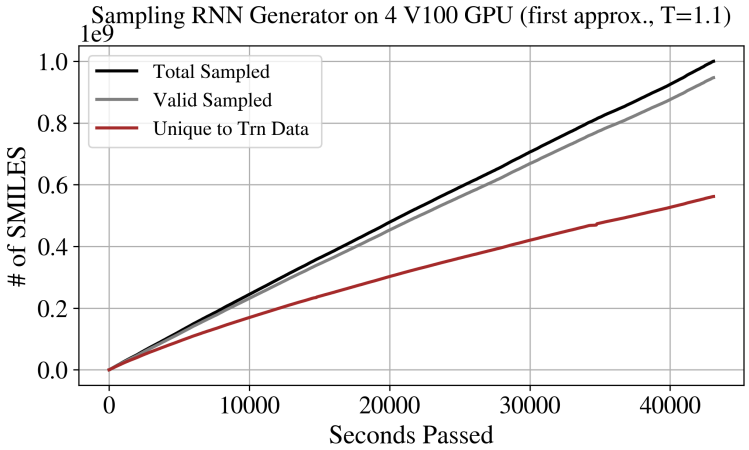
RNN SMILES Modeling

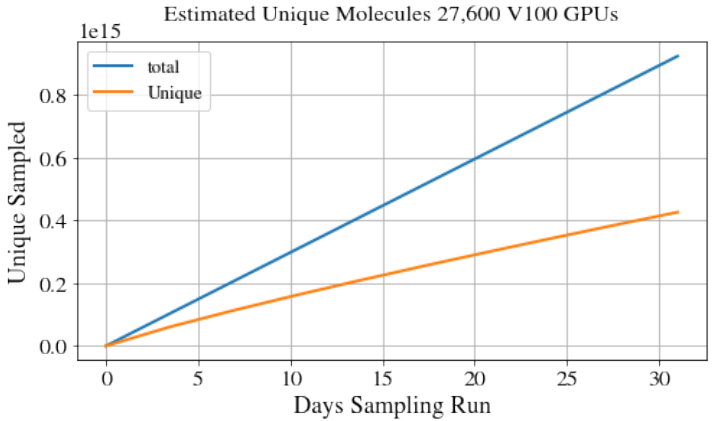


Gupta, Anvita, et al. "Generative recurrent networks for de novo drug design." Molecular informatics 37.1-2 (2018): 1700111.





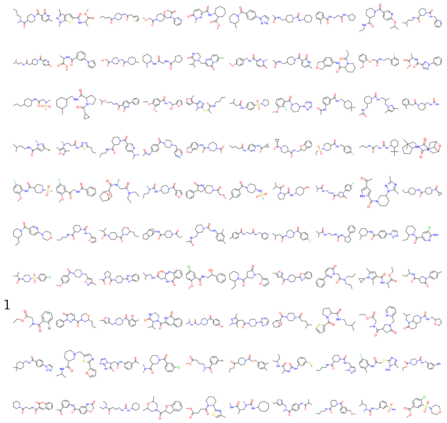




KINASE INHIBITOR DESIGN USING DEEP LEARNING AND MODELING

Generate Molecules Via Deep Learning

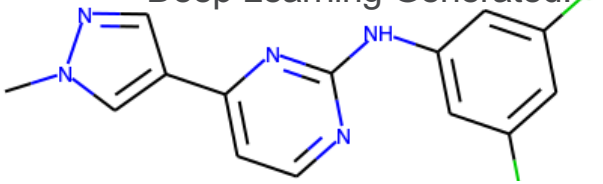
- Generative deep learning models can produce novel compounds mimicking distributions of molecules in training data.
- 1,900 unique and valid SMILES can be generated per second per GPU



Fast 3D-overlay Query

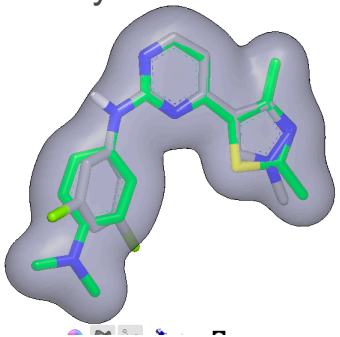
- Find novel 2D scaffolds that have conformers similar to known kinase inhibitors:

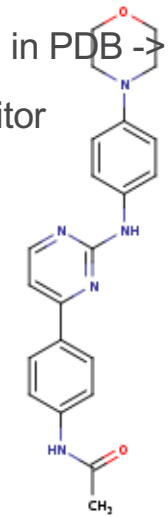
Deep Learning Generated:



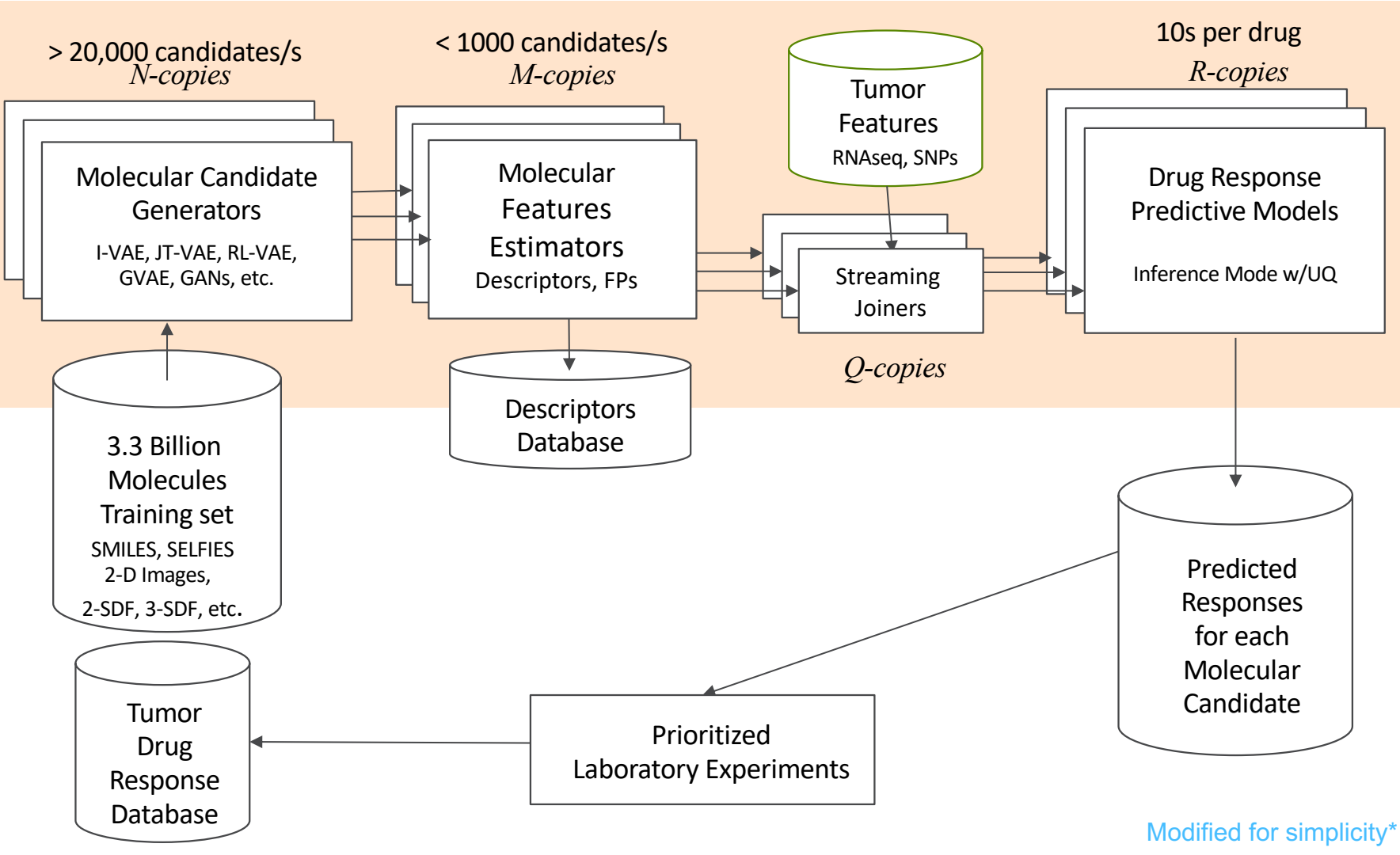
Closest 2D Similarity to any Ligand in PDB ->

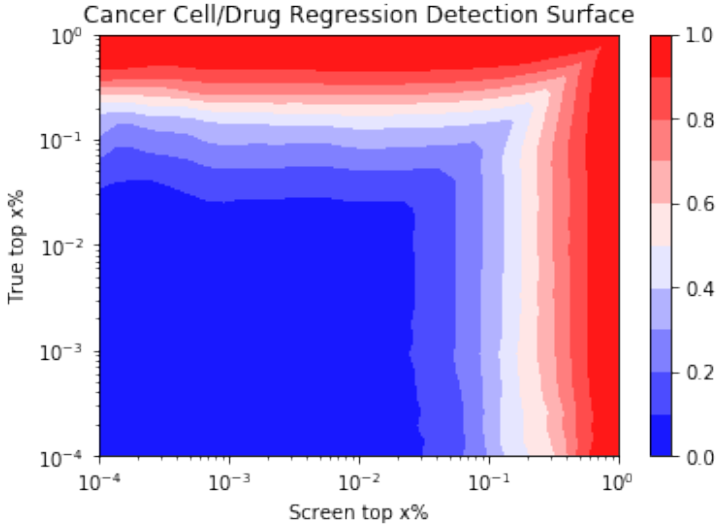
3D overlay to Known Kinase Inhibitor

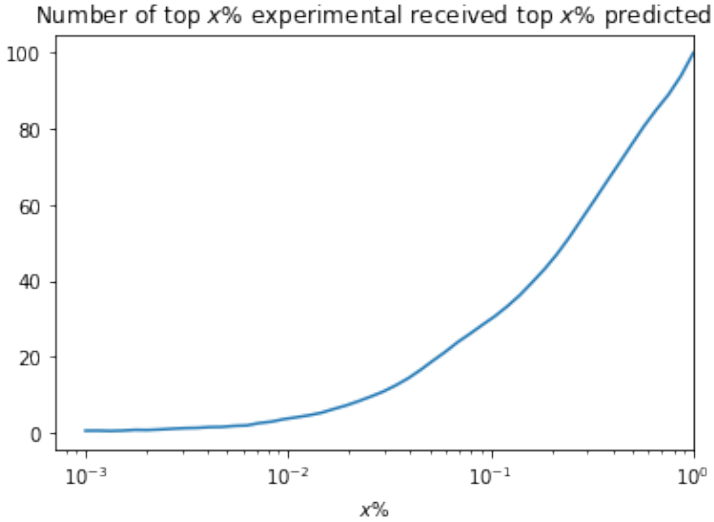




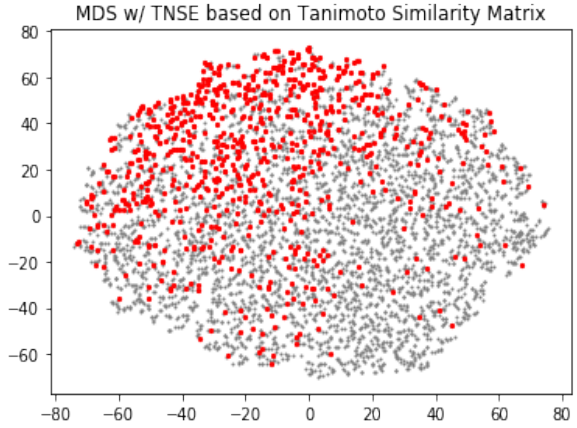
Suppose we trained the continuous model
How can we create a virtual screen?


> 20,000 candidates/s
N-copies
Molecular Candidate Generators
I-VAE, JT-VAE, RL-VAE, GVAE, GANs, etc.
< 1000 candidates/s
M-copies
Molecular Features Estimators
Descriptors, FPs
Tumor Features
RNAseq, SNPs
Streaming Joiners
Q-copies
10s per drug
R-copies
Drug Response Predictive Models
Inference Mode w/UQ
3.3 Billion Molecules Training set
SMILES, SELFIES
2-D Images,
2-SDF, 3-SDF, etc.
Descriptors Database
Predicted Responses for each Molecular Candidate
Tumor Drug Response Database
Prioritized Laboratory Experiments
Modified for simplicity*





$$\hat{\mathcal{R}} = \hat{f}_{\theta}(\mathcal{T}, \mathcal{D})$$



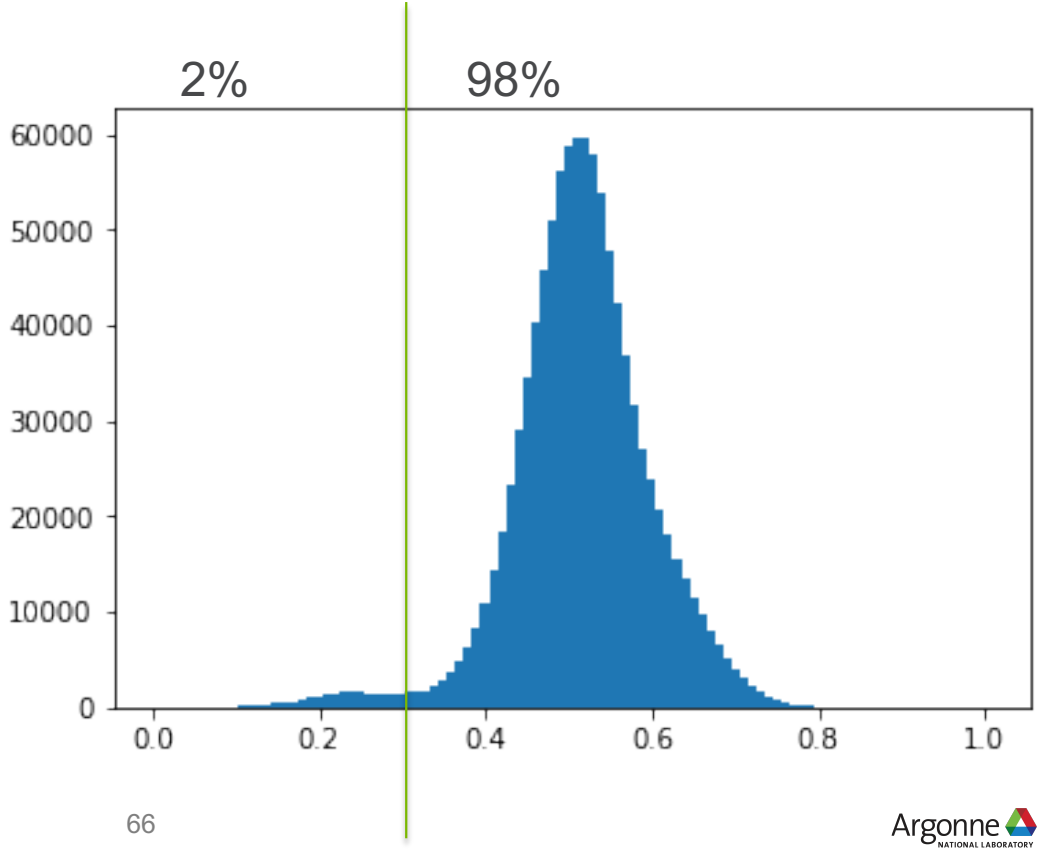
DIFFERENT EXAMPLE
SAME IDEAS

EXAMPLE: ML FOR DOCKING SCORING

Interested in the left tail

What is r2 score if we just guess everything in that right tail is clipped at the normal distribution? 0.75

Your balanced accuracy? 50%

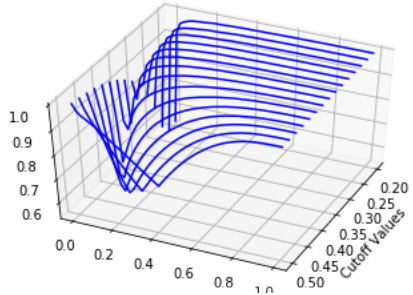


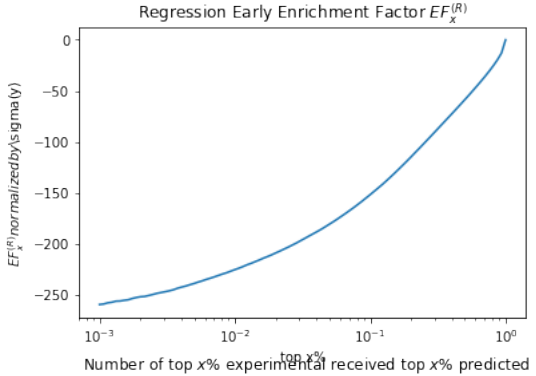
It's not obvious that's what we want either…

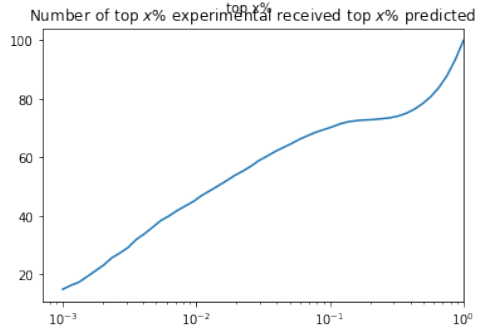
$$EF_{x\%} = \frac{1}{EF_{x,\max}}\left(\frac{\#\text{ of actives in top r, model ranked}}{x \cdot \#\text{ of actives}}\right)$$

$$EF_{x\%}^{(R)} = \frac{1}{xN}\sum_{i=0}^{xN}\frac{y_i - \bar{y}}{\sigma(y)}$$

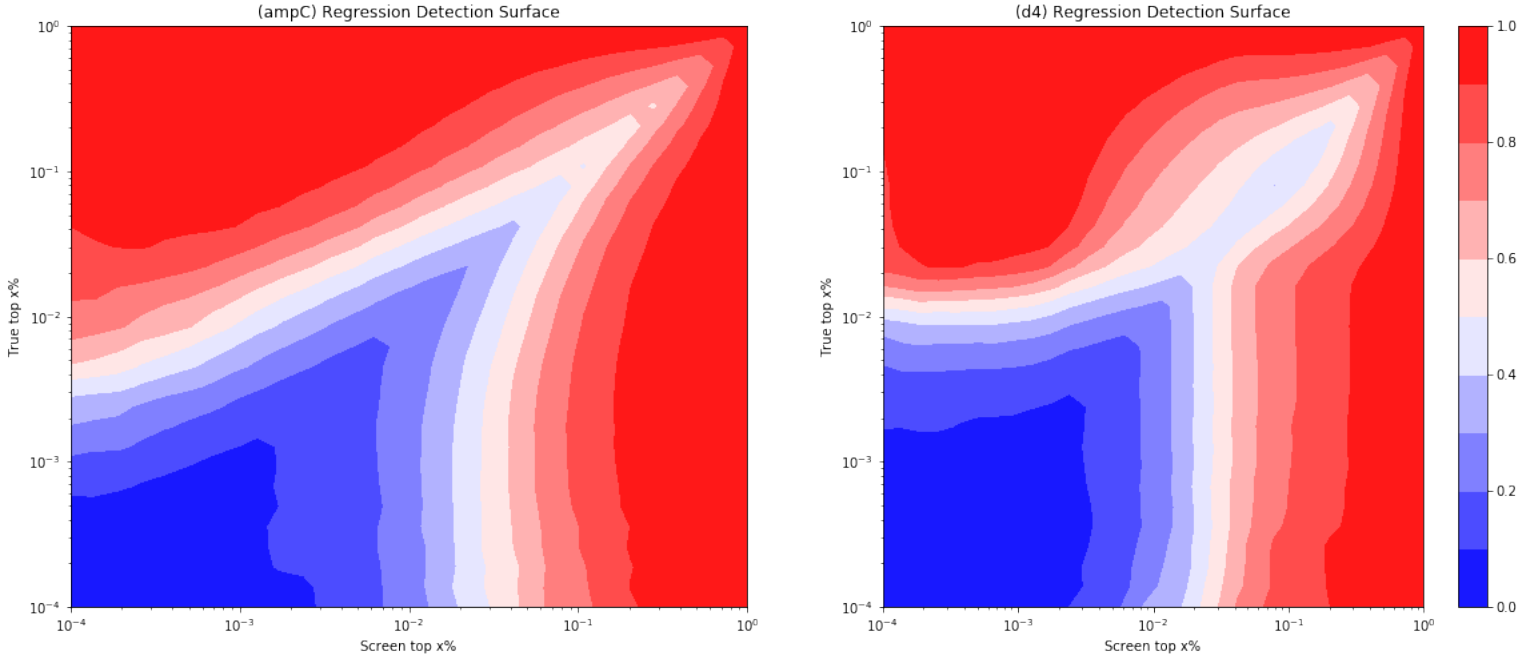
$$EF_{x\%}^{(\mathrm{COUNT})} = \frac{|\ \mathrm{TopR}(y,x) \cap \mathrm{TopR}(\hat{y},x)\ |}{xN}$$





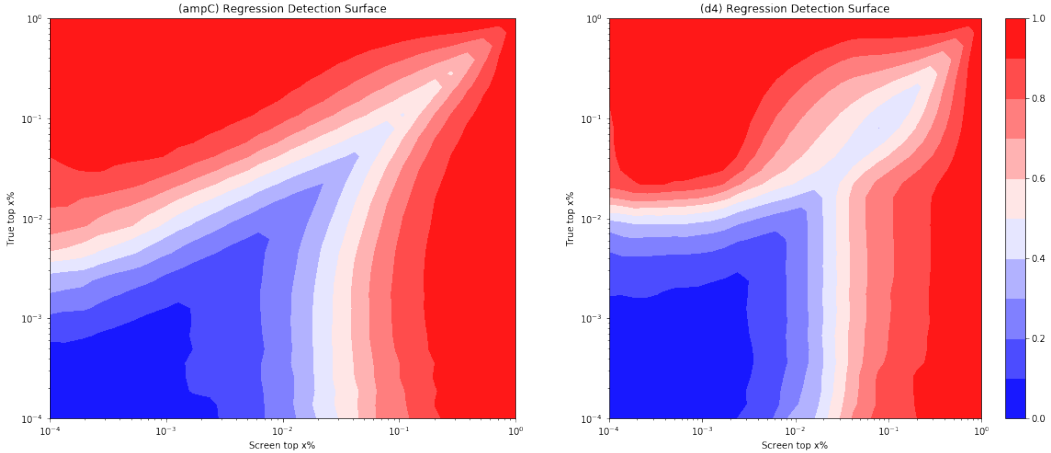


$$EF_{x\%}^{(\mathrm{COUNT})} = \frac{|\ \mathrm{TopR}(y,x) \cap \mathrm{TopR}(\hat{y},x)\ |}{xN}$$



Is this a good model?

- R2 score isn't good
- MAE isn't good
- But we can with >90% certainty tell a drug company they can reduce their search space by at least one order of magnitude.



THANKS