2022 Data Science Challenge

Discovery of Small-Molecule Inhibitors of SARS-CoV-2 using Machine Learning

Hyojin Kim

LLNL-PRES-835518

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

Agenda

- Background
- Task 1: Screen compounds using SMILES and molecular descriptor
- Task 2: Screen compounds using 3D atomic representation
- Bonus Task: Model fusion and performance analysis
- Tips and Tricks: 3D voxelization, network design, model fusion
- Q/A

Background

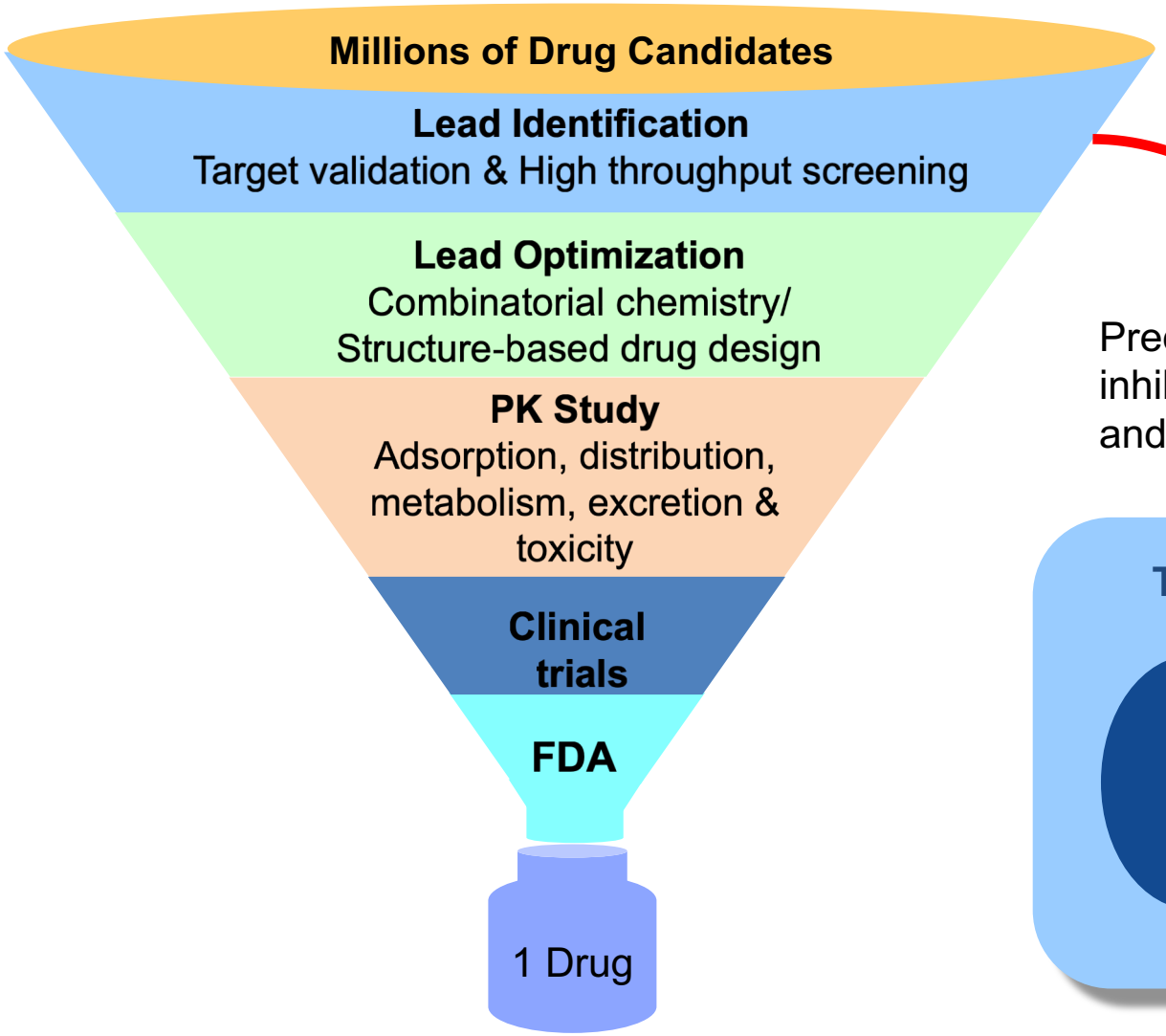
Computational chemistry plays an important role in virtual high-throughput screening (vHTS) for drug design

ChEMBL

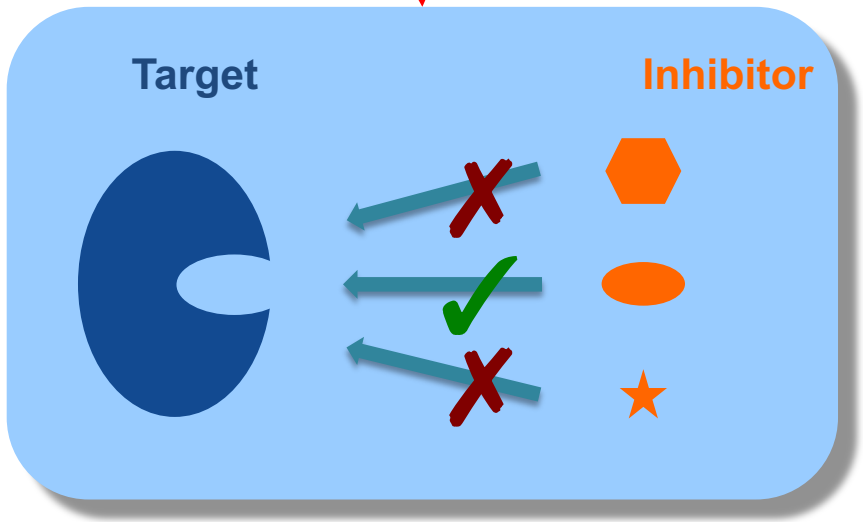
eMolecules®

Enamine

BROAD INSTITUTE



Predicting drug binding to help find inhibitors/activators for therapeutic targets and off-target binding (AutoDock Vina)



Molecular docking is widely used to predict binding modes between ligand and protein in structural molecular biology

- Molecular docking is a method in computational chemistry to find molecule's pose (orientation) and conformation where the ligand and target receptor are bound to form a stable complex.
- The best pose is selected based on the scoring functions (binding free energy).
- Ligands are screened using their best score (lowest binding free energy).

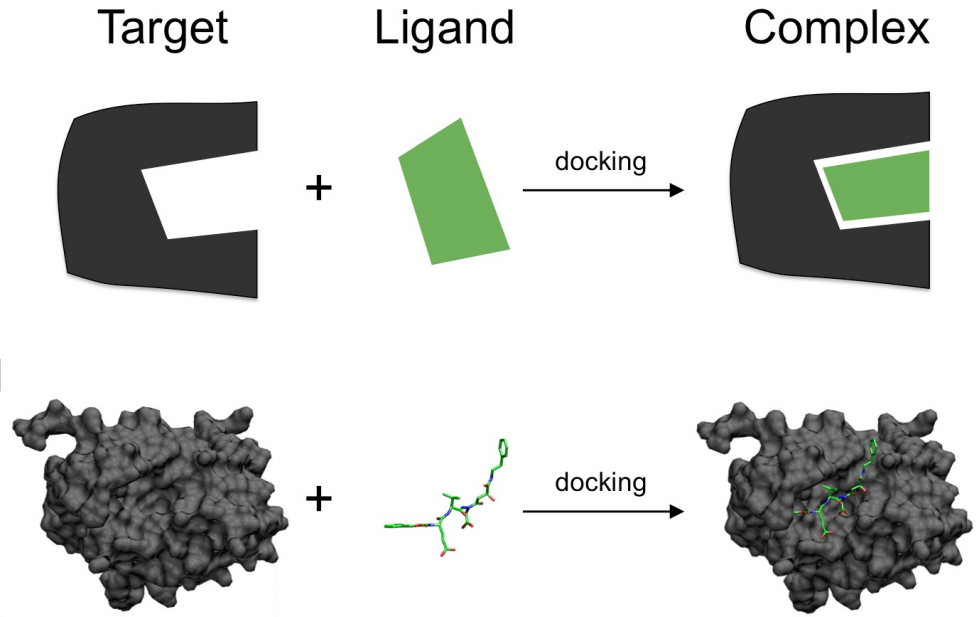
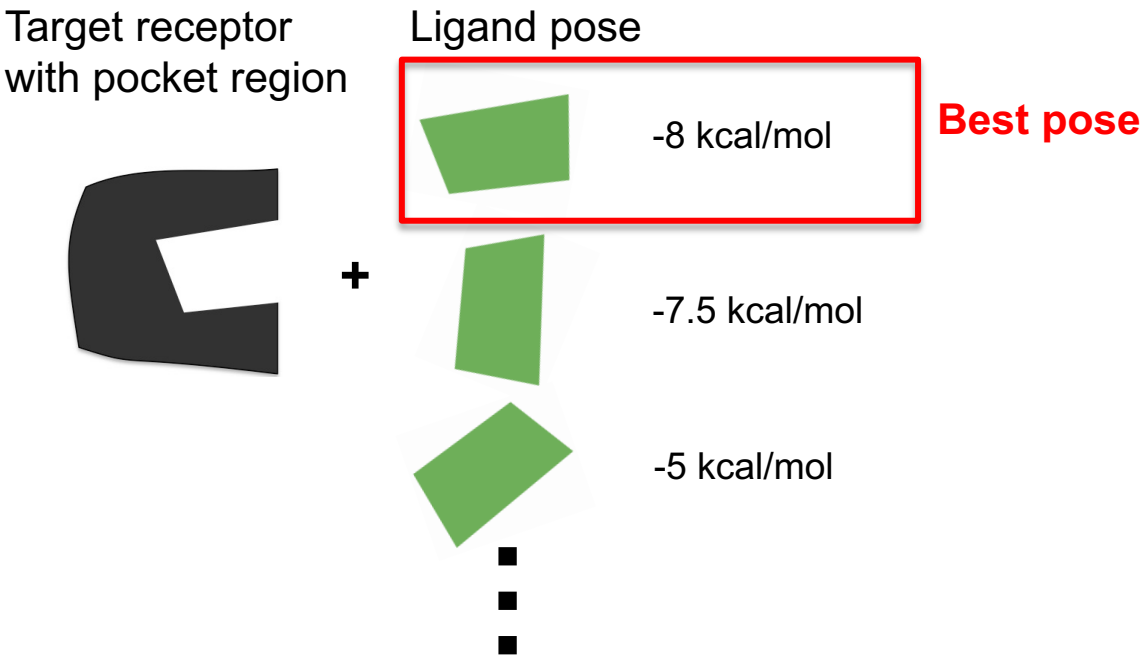
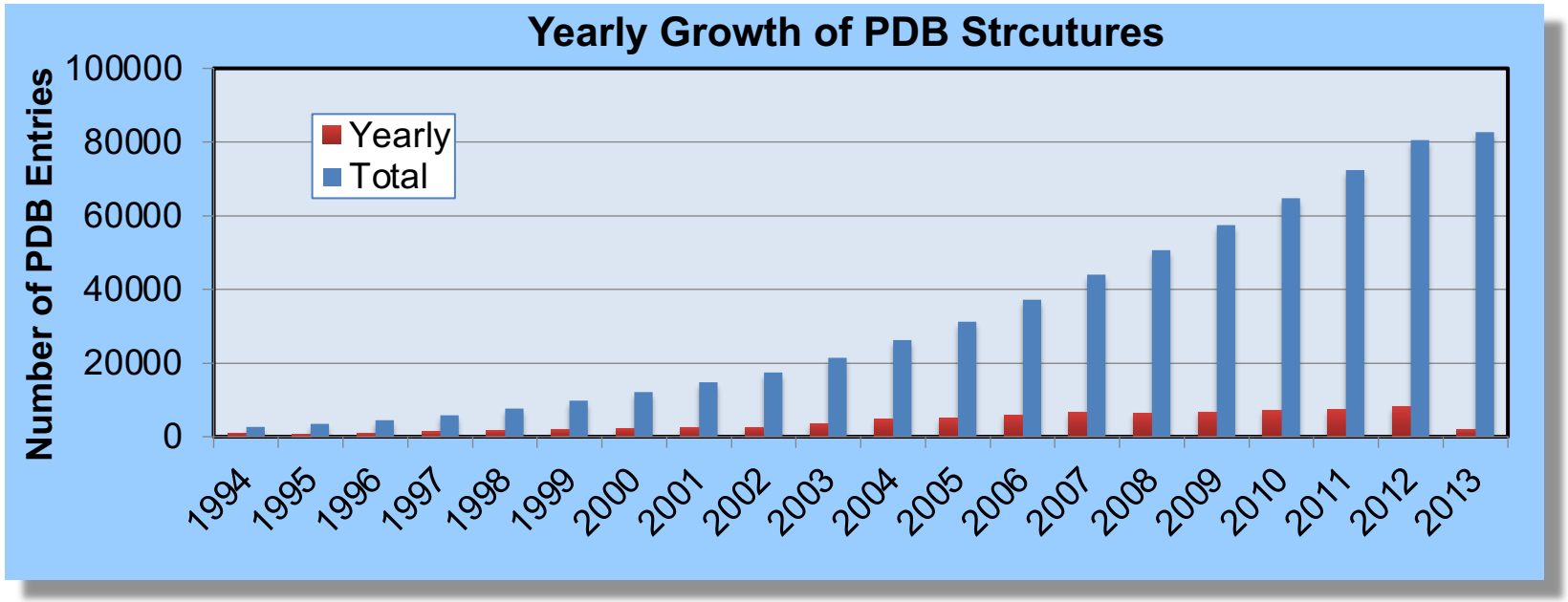


Image courtesy: wikipedia



We need scalable molecular docking with high performance computing for rapid drug screening

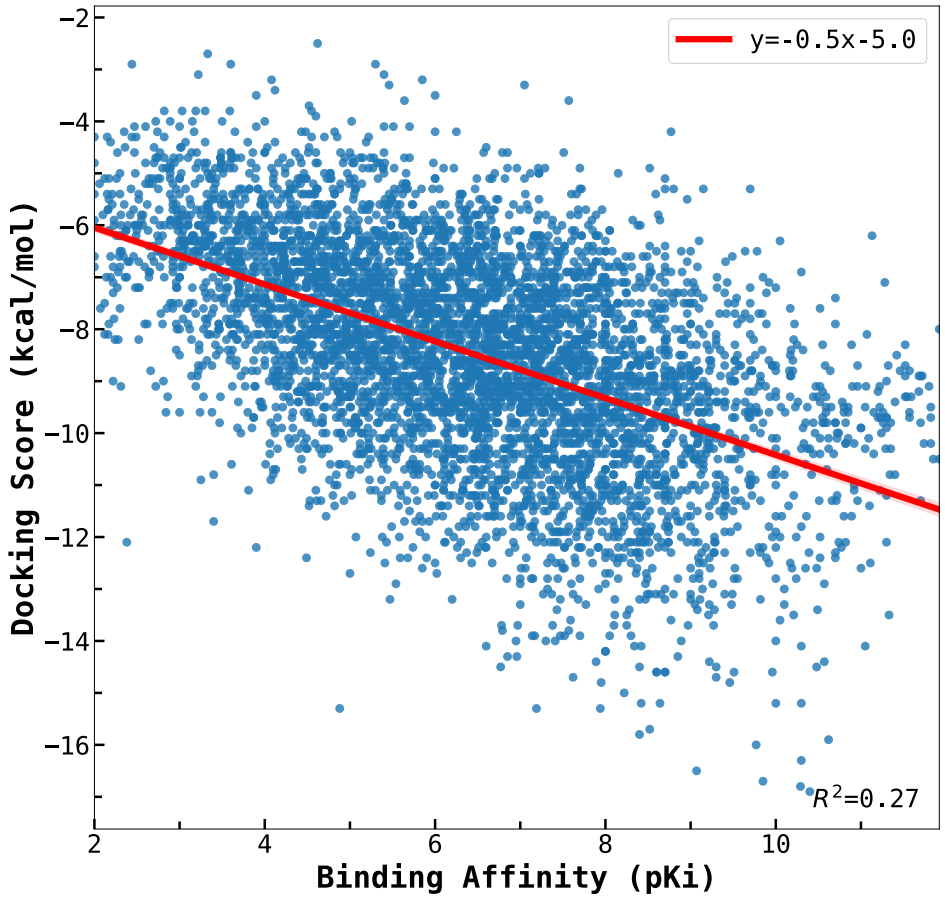
- 1 ligand docking into 1 target takes ~1 min. Rescoring 1 pose takes >10 min.
- Drug-like compounds ~10^{60} possibilities, more chemical compounds available every year
  - e.g., Enamine's REAL database comprises over 4.5 billion molecules



More accurate screening solution is needed to complement molecular docking

- Low correlation between molecular docking's binding free energy and real binding affinity results in incorrect screening with many false positives.

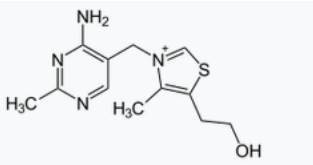
Autodock Vina
for PDBbind 2020

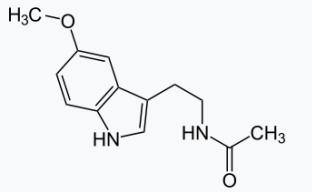


Recent efforts using machine learning technology show great potential to improve vHTS

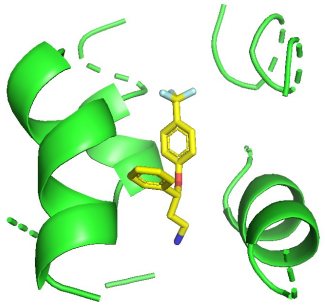
- Supervised machine learning approaches have been developed by training ML models on ligand-only or protein-ligand complex data to predict interaction (binding affinity).
- Another direction is use of unsupervised feature learning (VAE, attention, etc)
- ML models can complement molecular docking

Ligand only (for single protein target)





Protein-ligand complex



Machine Learning Models

Binding Affinity

We have several choices to represent ligand and protein data

- Ligand data are usually derived from the simplified molecular-input line-entry system (SMILES) strings that encode molecular structures and specific instances in 1D string representation, from which we can extract various molecular descriptors or fingerprint
  - File format: .smi
- Ligand data can also be represented as 3D atomic data: list of atoms, each of which has 3D spatial coordinates and atomic features
  - File format: .pdb, .pdbqt, .sdf

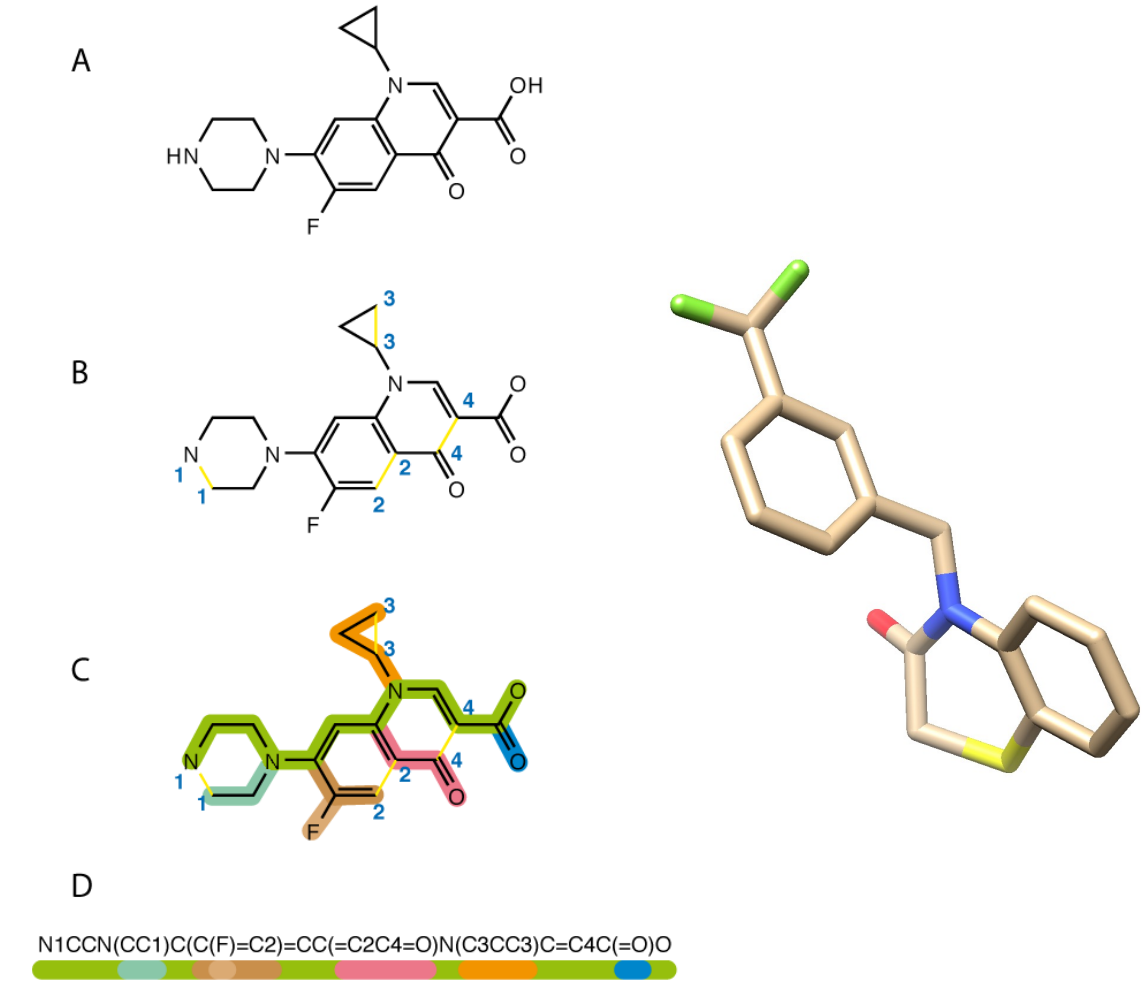
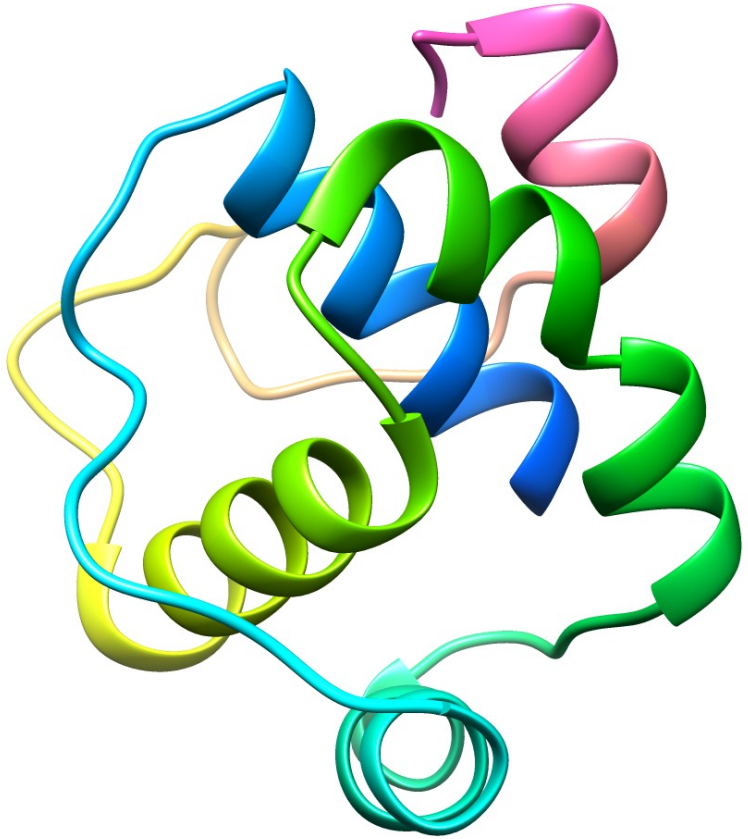


Image courtesy: wikipedia

We have several choices to represent ligand and protein data

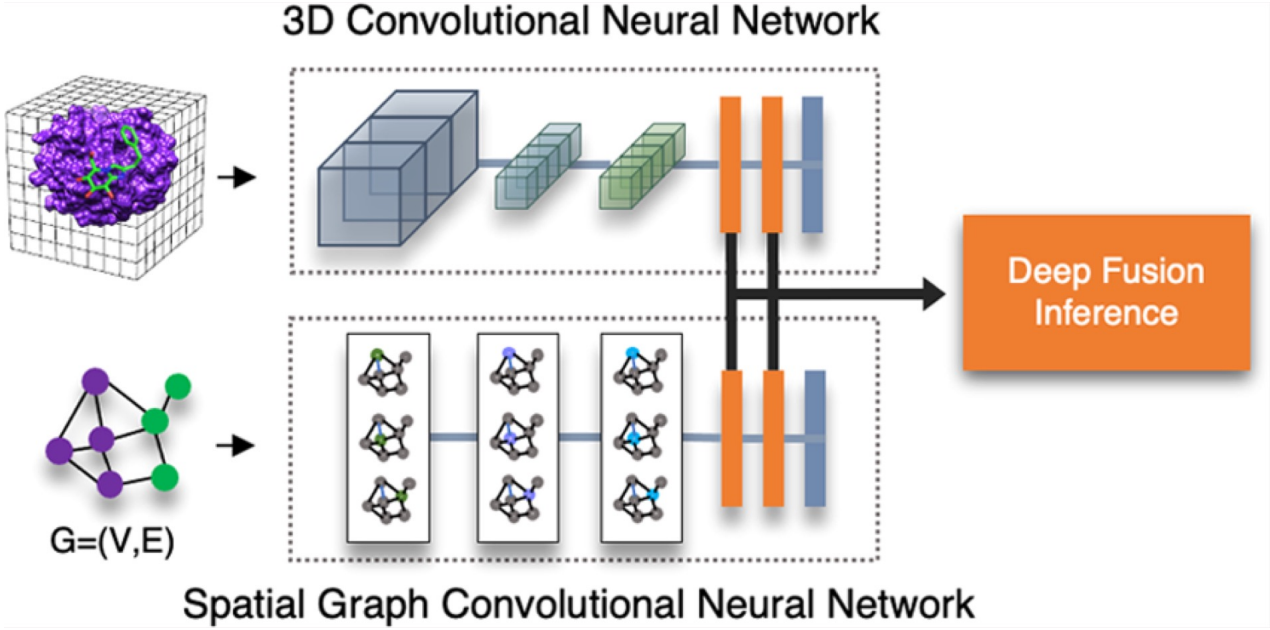
- Protein data can be derived from amino acid sequences and/or 3D atomic representations
  - File format for 3D structures: .pdb, .pdbqt

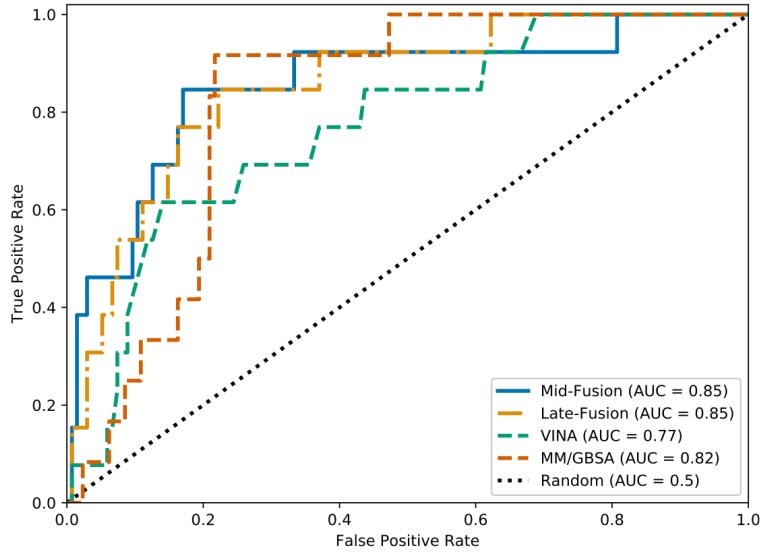
```
Mpro-N0029_0A_apo.pdb/1-103    1 ITVNVLAWLYAAVINGDRWFLNRFTTTLNDFNLVAMKYNYEPLTQDH  47
Mpro-N0029_0A_apo.pdb/1-103   48 VDILGPLSAQTGIAVLDMCASLKELLQNGMNGRTILGSALLEDEFTP  94
Mpro-N0029_0A_apo.pdb/1-103   95 FDVVRQCSG                                        103
```



Our proposed ML approach using 3D structures of protein-ligand data outperforms ligand-only models

- We proposed to use two different ML architectures to complement each other (Fusion): 3D-CNN to capture 3D spatial information (shape), SG-CNN to capture relationships between ligand and protein atoms.
- Our ML approaches trained on protein-ligand data have proven to be fast, effective, compared to existing docking and molecular dynamics (MD) methods





Jones, Kim, et al., 2021

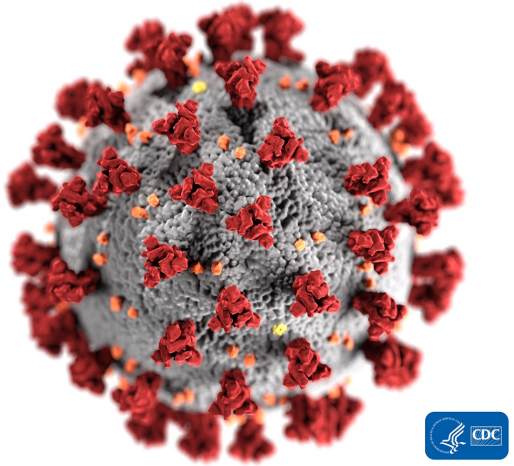
Terminology

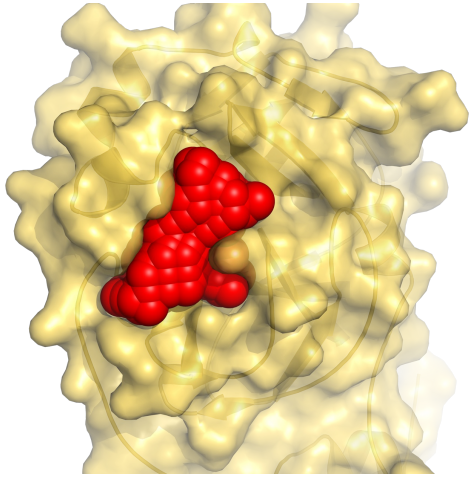
- Ligand: small molecule that forms a complex with a biomolecule (protein receptor)
- Binding affinity: strength of binding interaction between a ligand and a protein target
- Complex: protein-ligand structure where protein and ligand interact with each other
- Crystal structure: 3D arrangement of molecules or atoms throughout a crystal (considered a ground truth pose)
- Docking pose: complex data (predicted binding-conformation of ligand to target binding site) by molecular docking modeling methods (e.g., AutoDock VINA) used in structure-based drug development
- Protein Data Bank (PDB): a database for 3D structural data of biological molecules such as proteins and nucleic acids, each molecule data has its own PDB id
- SMILES: Simplified Molecular Input Line Entry System, 1D representation of a ligand structure

DSC Tasks

DSC 2022 is to develop machine learning methods to screen compounds targeting COVID-19

- The goal is to develop machine learning approaches to find potential inhibitors targeting main protease receptors of SARS-CoV-2 (“mpro”).
- We will use Postera Mpro dataset collecting experimentally measured binding affinity data, with LLNL’s curated compounds with binding affinity data.
- We will perform two challenge tasks to predict binding information using two different data representations.



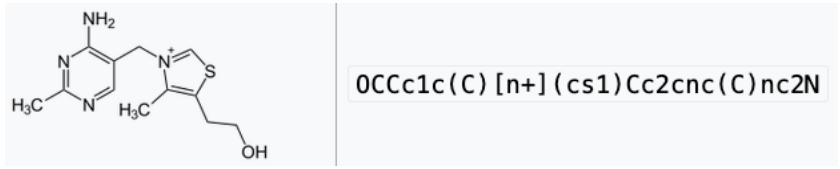


Main protease
PDB ID: 6Y84
Diamond, UK

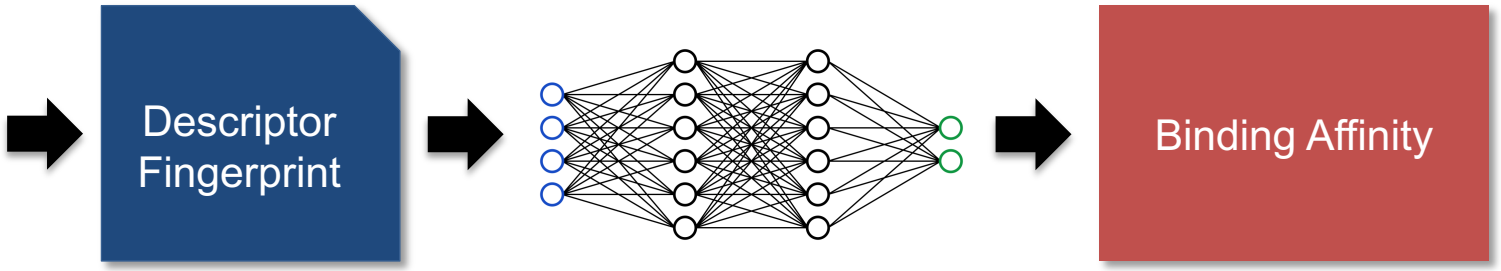
Find inhibitors to stop viral replication

Task 1 is to use molecular descriptors to predict binding affinity

- Molecular descriptors derived from SMILES string representation of ligand data will be used to predict binding affinity.
- We formulate binding affinity as a classification problem: bind vs. no-bind
- The goal is to develop ML models using random forest, neural network, etc.



Compound and its SMILES string



Input

Output Label

Task 1 dataset is based on rdkit feature descriptors

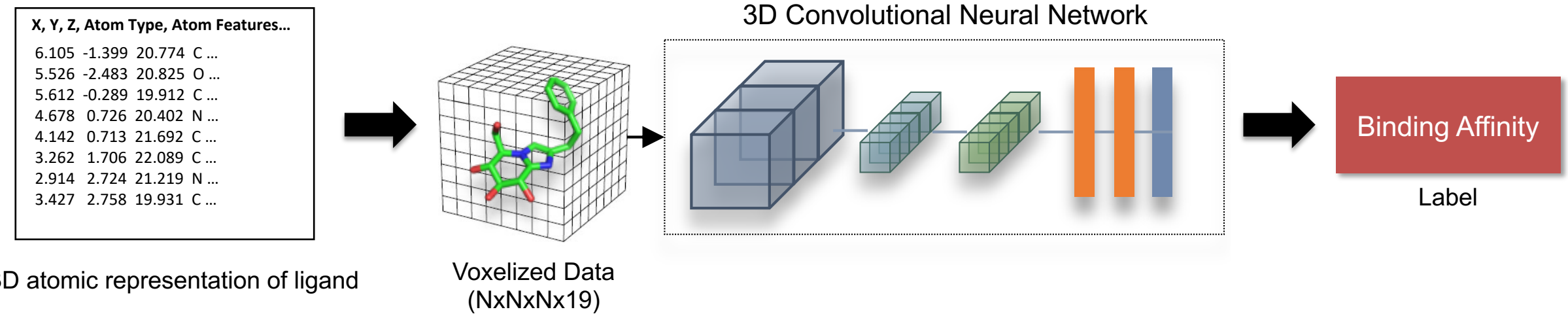
- The input data is ligand feature descriptors using rdkit
  - Rdkit is open-source cheminformatics python package
  - We used rdkit.ML.Descriptors.MoleculeDescriptors.MolecularDescriptorCalculator
- The descriptor contains various compound and chemical attributes such as
  - Molecular weights, average molecular weight of the molecule ignoring hydrogens, number of radical electrons, number of valence electrons, etc
  - 208 feature vectors
- The input file is **mpro_exp_data2_rdkit_feat.csv**
  - cmpd_id: compound id
  - smiles: SMILES string
  - label: 0 (no-bind) or 1 (bind)
  - subset: train/validation/test split (1955 training, 113 validation, 240 test samples)
  - feat_1 to feat_208: rdkit descriptor features
  - Input csv contains **NaN**
- Instead of the provided descriptors, other molecular descriptors or fingerprints can be used.
  - https://rdkit.org/UGM/2012/Landrum_RDKit_UGM.Fingerprints.Final.pptx.pdf

In Task 1, you will develop a ML method to read input molecular descriptors to classify bind or no-bind of the compound

- Any ML method can be used to predict interaction between ligands and the main protease.
  - Random forest
  - Neural network with fully connected layers
- Model performance can be measured using classification metrics: accuracy, precision, recall and F1 score
  - https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics
- Receiver operating characteristic (ROC) curve and Area Under the Curve (AUC) can also be reported
  - https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html

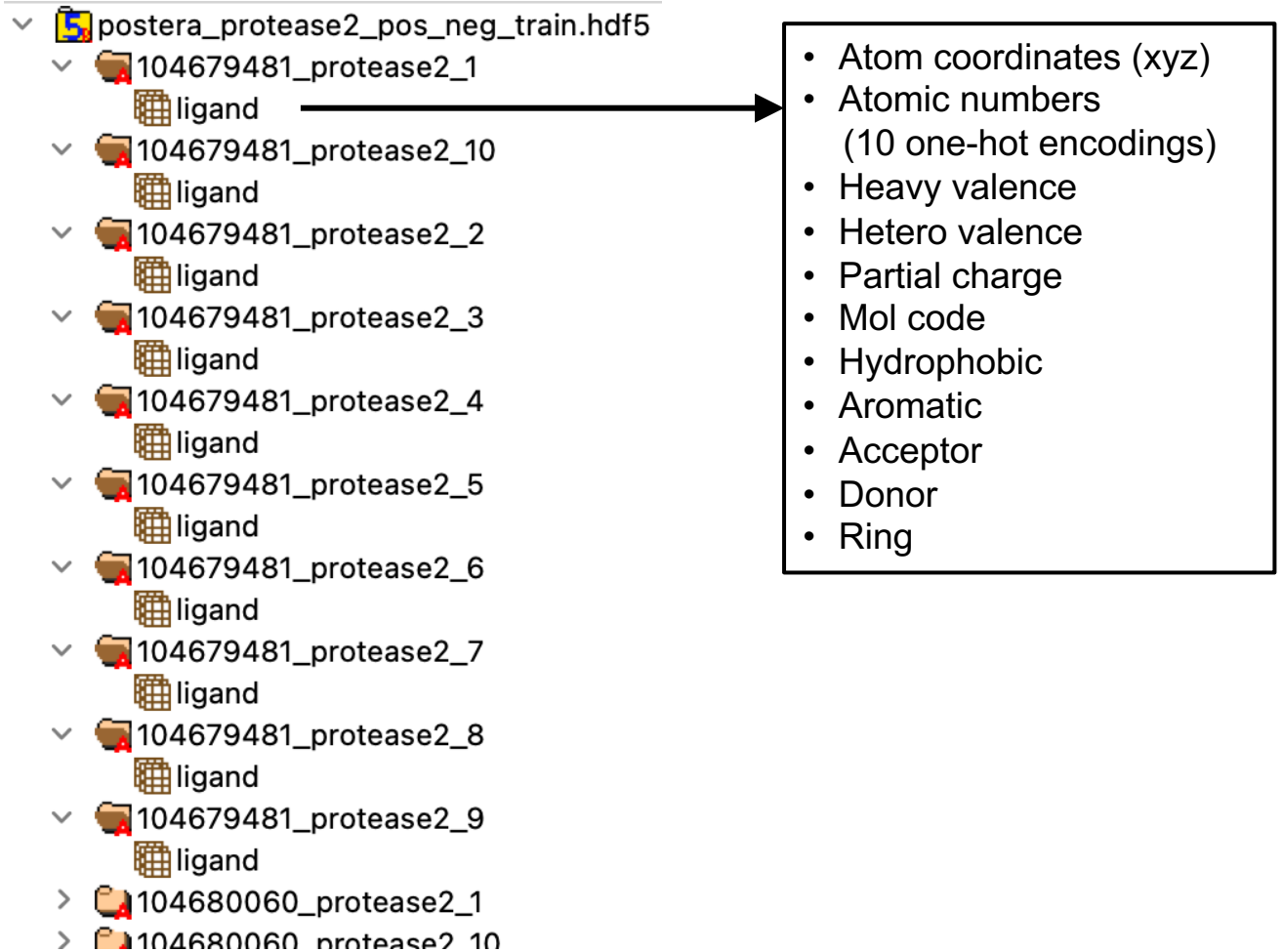
Task 2 is to develop ML models to predict binding affinity using 3D structures of ligand data

- Now let's use 3D structure information of ligand data
- The goal is to develop a method to predict interaction as a classification problem using any CNN method to interpret 3D atomic representation



Task 2 dataset contains 3D atomic representations for ligands

- There are three hierarchical data format (HDF5) files for training, validation and test:
  - postera_protease2_pos_neg_train.hdf5
  - postera_protease2_pos_neg_val.hdf5
  - postera_protease2_pos_neg_test.hdf5
- Each ligand contains 10 different poses (but without the target protein receptor) with a label (0 or 1), each of which contains 3D information in “ligand”
  - Keys: compound id + protease2 + pose id
- Each 3D ligand data is a multi-dimensional array whose size is 100x21 where 100 is the total number of atoms and 21 is the feature size: 3 for xyz coordinates and 19 features
  - In the case of a ligand with less than 100 atoms, the rest rows will be zero padded.



HDF viewer: https://www.hdfgroup.org/downloads/hdfview/

You will use a convolutional neural network architecture to read 3D ligand data

- We recommend 3D-CNN with 3D convolutional filters/kernels. But any CNN method can be used to interpret 3D ligand data.
- For 3D-CNNs, the ligand array data needs to be represented into a 3D voxel grid
- Choice of voxel grid dimension depends on memory, resolution, collision, sparsity, filter size, etc
- Model performance report using classification metrics: accuracy, precision, recall and F1 score
- ROC curve and AUC can also be reported

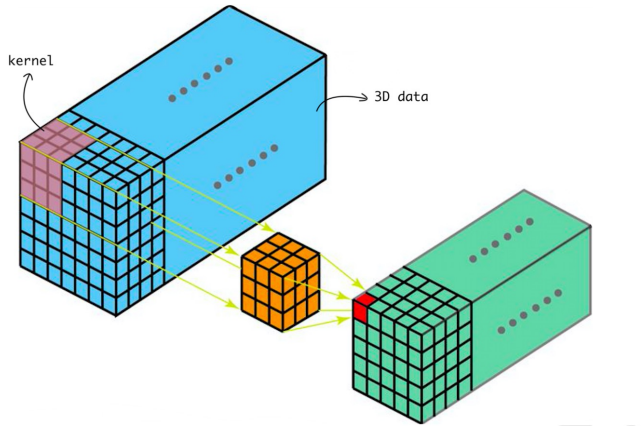
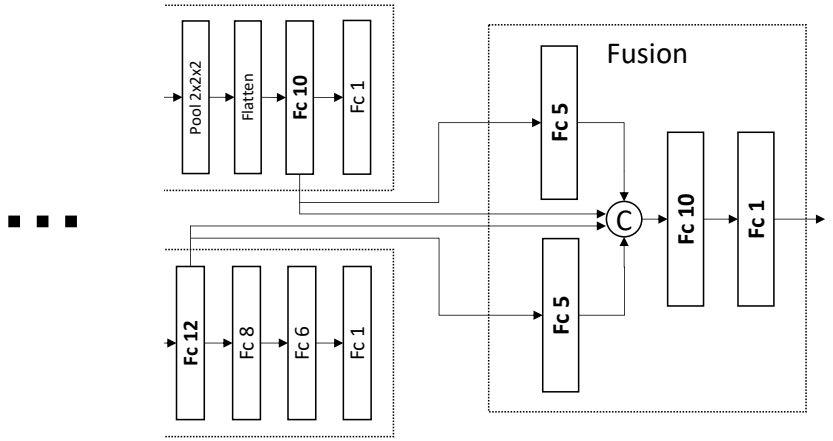


Image courtesy: towardsdatascience.com

Q/A

Bonus Task 1: Two different ML models using two different feature representations could be complementary to improve prediction

- The benefit to fuse different ML models lies in combining their possibly complementary feature representations.
  - e.g., video activity recognition by bridging feature difference between multiple input types (visual and temporal data)
- Our ”Fusion” framework shows an improved predictive power on the protein-ligand binding affinity.
- The simplest way to fuse two models is to average the final layer’s activations.
- Another way would be to fuse intermediate layer’s activations in a separate neural network.



Jones, Kim et al., 2021

Bonus Task 2: Comprehensive performance and behavior analysis is crucial for model interpretability

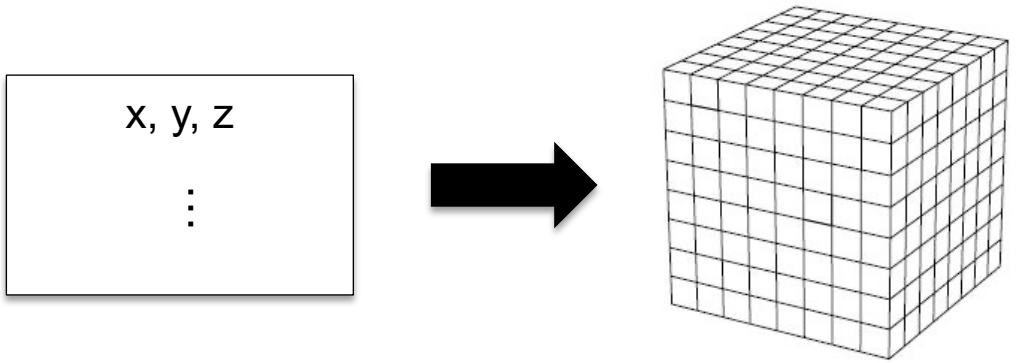
- Comprehensive model analysis includes 1) input feature importance, 2) spatial region relevance, 3) performance by compound family/similarity
  - Cluster compounds by SMILES, fingerprint, descriptors
  - Analyze performance per compound group

Tips and Tricks

3D voxelization

- The list of 3D (x,y,z) atom coordinates needs to be represented in a voxel grid for 3D-CNN.
- Things to be considered:
  - Voxel grid size (volume dimension)
  - Resolution (angstroms per voxel)
  - Collision
  - Sparsity
- Reference: https://github.com/LLNL/FAST/blob/master/model/3dcnn/img_util.py

  (Pytorch version instead of numpy operation can be implemented)



Network Design

- Convolutional layers – number of layers, filter/kernel size
- Convolutional layers - residual/skip connection
- Hyperparameter settings, Dropout, Batch normalization

Network Design

- Convolutional layers – number of layers, filter/kernel size
  - Do not use too many convolutional layers due to the limited number of training samples
  - Do not use too small or too large kernel sizes
- 1x1 convolutional kernels (1x1x1 for 3D)
  - Channel-wise or depth-wise dimension reduction
  - Feature pooling: projection of the feature maps
  - Additional nonlinearity after regular convolutions

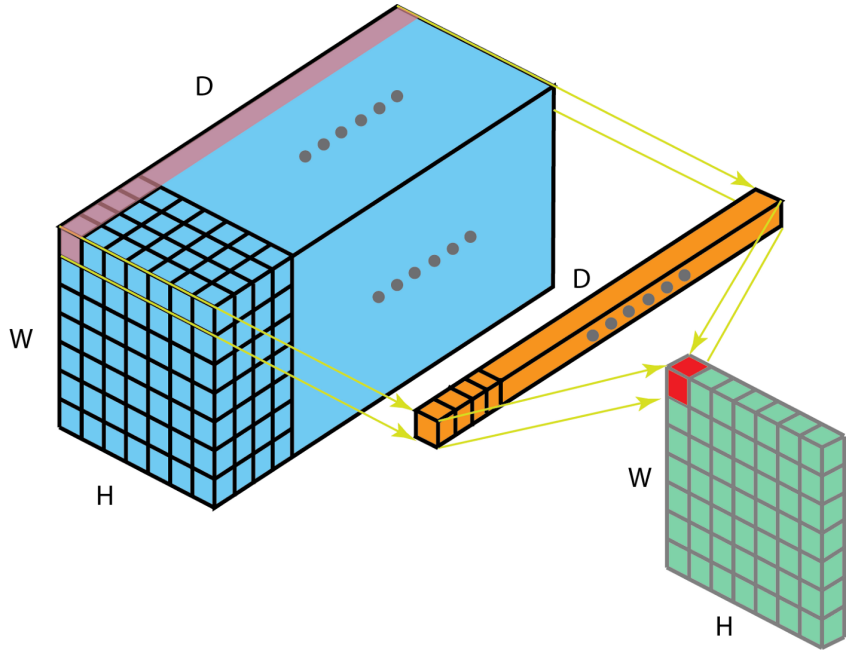
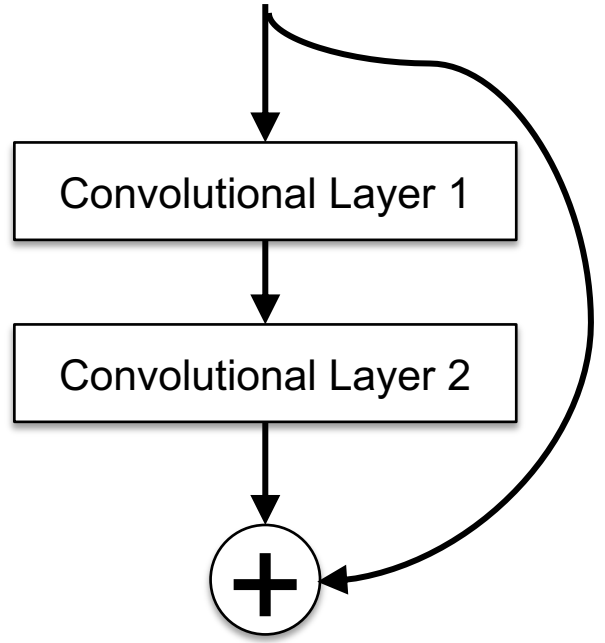


Image courtesy: https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

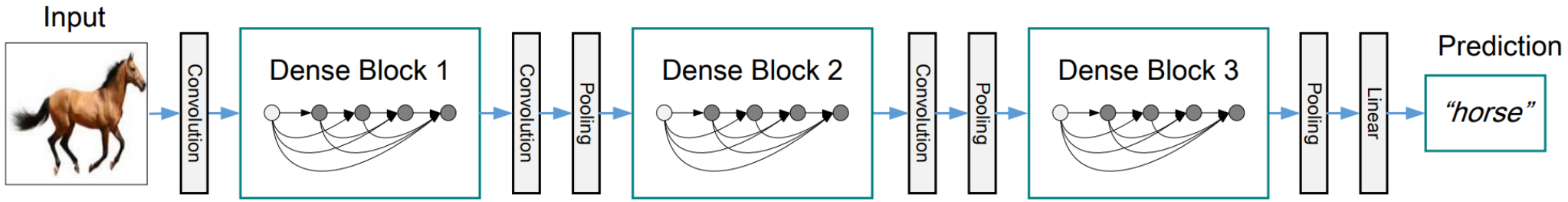
Network Design

- Residual/skip connection
  - Finding an optimal number of layers is non-trivial.
  - Can be an ensemble of different models to get improved accuracy
  - Addition or concatenation

```
conv1 = self.conv_block1(x)
conv2 = self.conv_block2(conv1)
conv_res = x + conv2
conv3 = self.conv_block3(conv_res)
…
```



- DenseNet: utilize “dense block” where all layers are connected



DenseNet, Huang, et al., 2018

Network Design

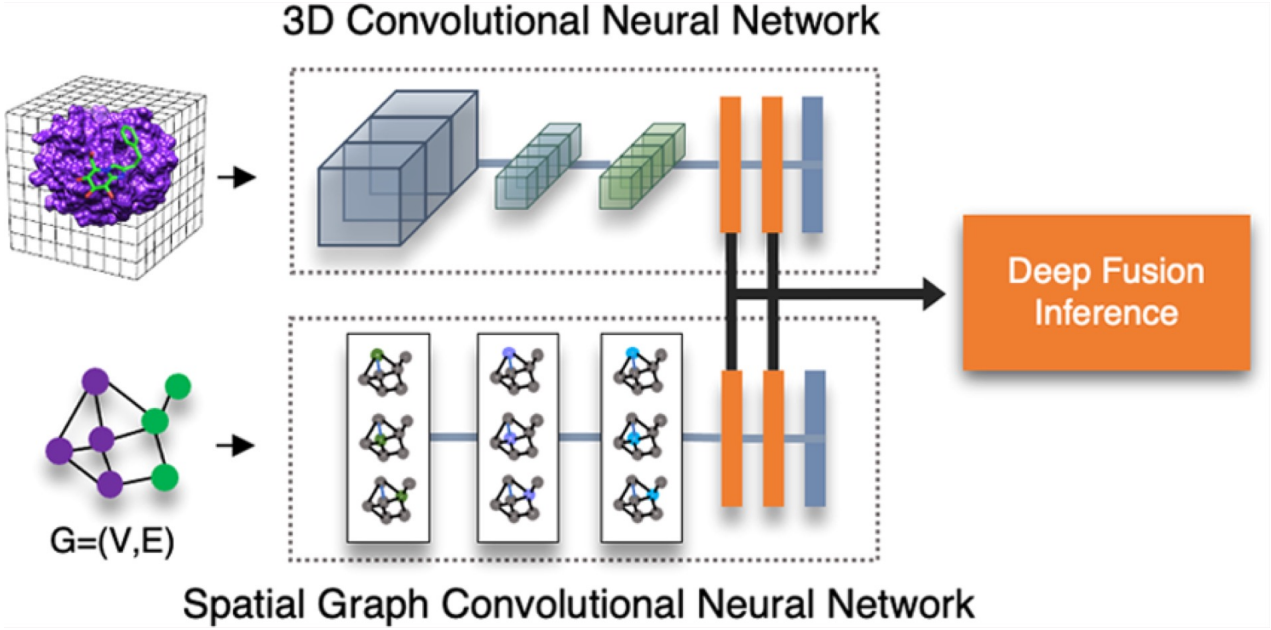
- Hyperparameter settings, Dropout, Batch normalization
  - To avoid overfitting
  - To converge fast

```
def __conv_layer_set__(self, in_c, out_c, k_size, stride, padding):
    conv_layer = nn.Sequential(
        nn.Conv3d(in_c, out_c, kernel_size=k_size, stride=stride, padding=padding, bias=True),
        nn.ReLU(),
        nn.BatchNorm3d(out_c))
    return conv_layer

self.conv_block1 = self.__conv_layer_set(19, 64, 3, 2, 1)
…
```

Model Fusion

- Two models from task 1 and 2 can be further fused using an additional neural network to achieve “complementary” feature learning
- Reference: https://github.com/LLNL/FAST



Jones, Kim, et al., 2021

Q/A