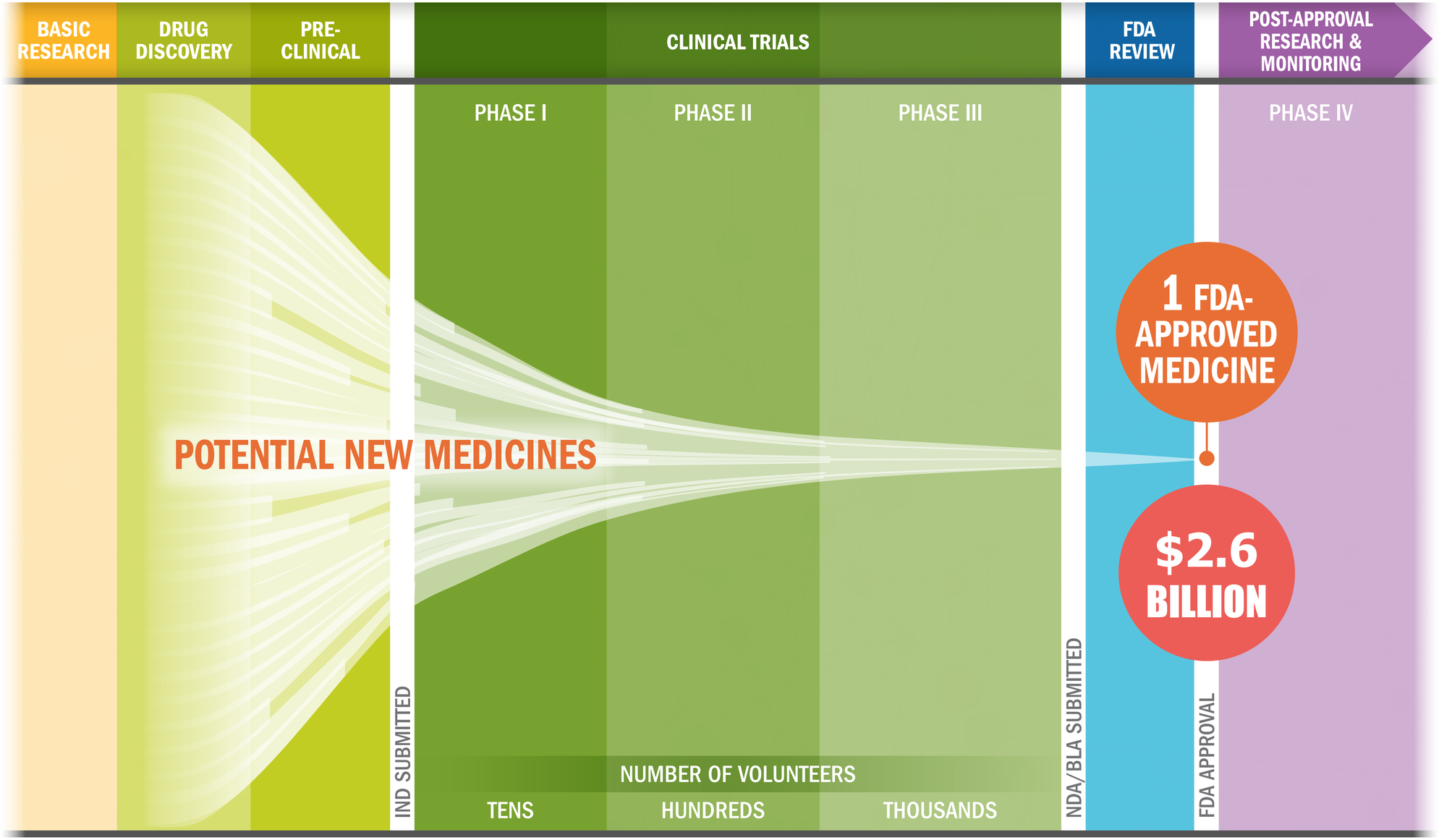
Deep Learning for Computational Drug Discovery

David Koes

@david_koes

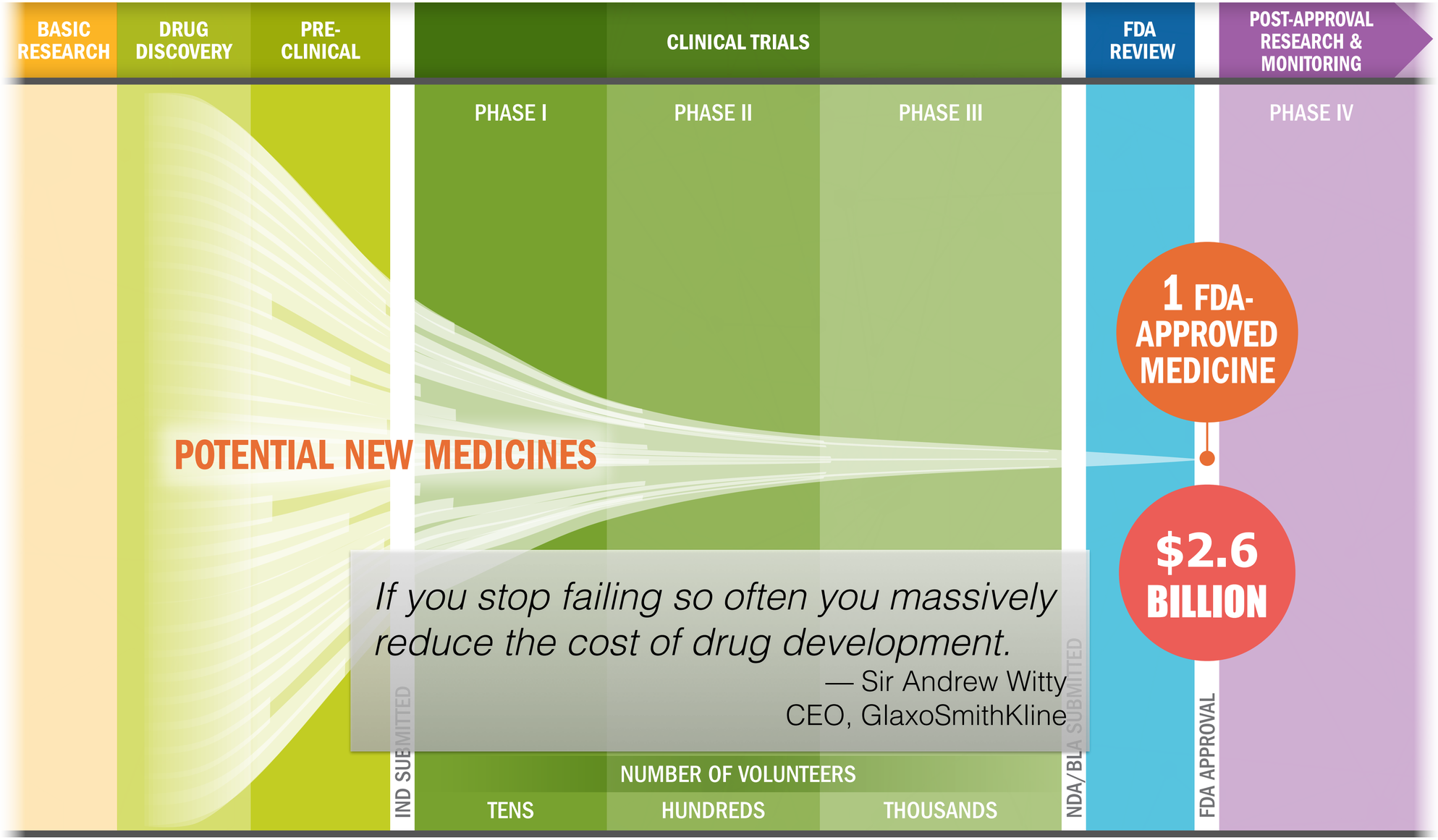
Intelligent Systems Program
AI Forum
October 19, 2018

THE BIOPHARMACEUTICAL RESEARCH AND DEVELOPMENT PROCESS



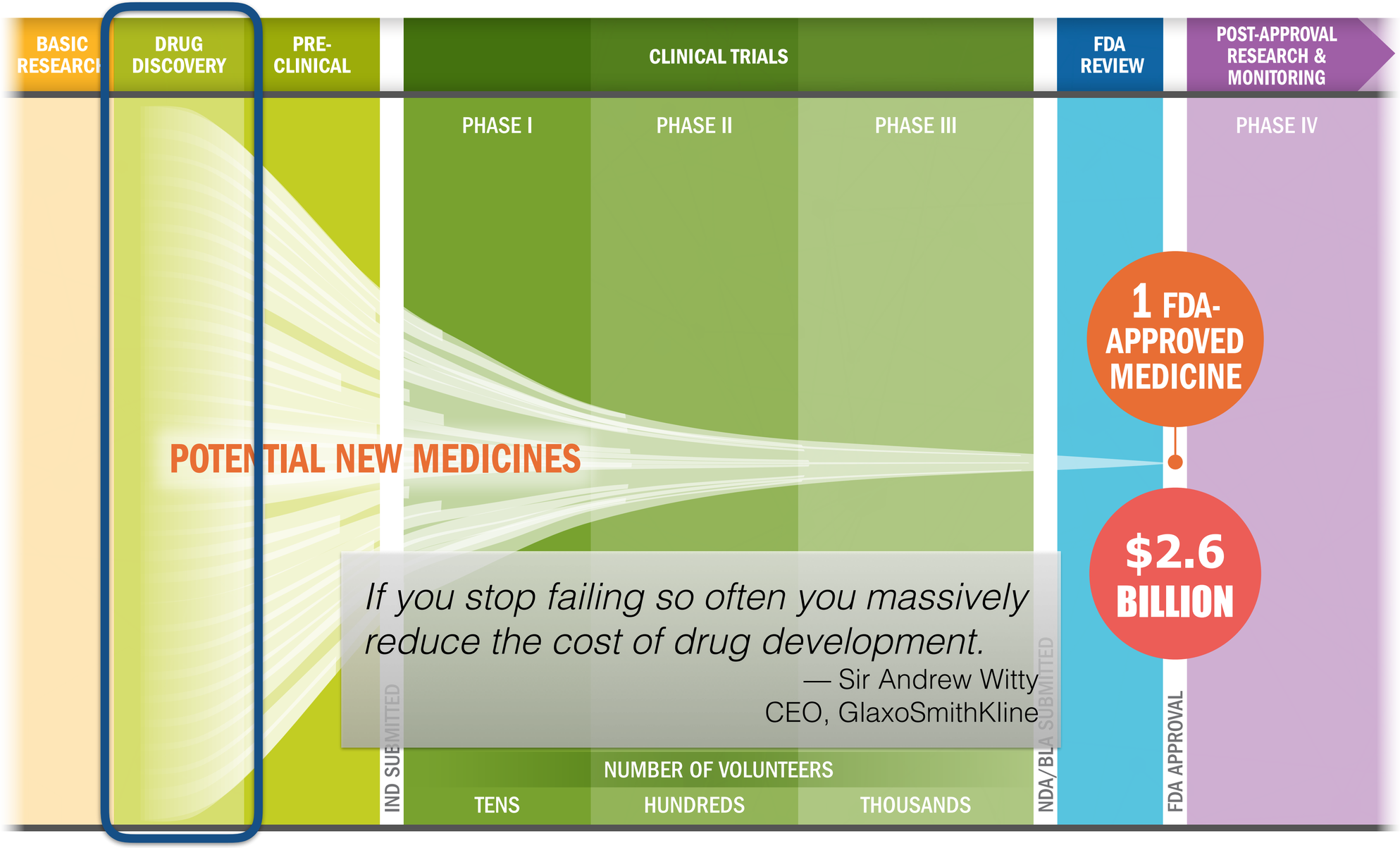
Source: Pharmaceutical Research and Manufacturers of America (http://phrma.org)

THE BIOPHARMACEUTICAL RESEARCH AND DEVELOPMENT PROCESS



Source: Pharmaceutical Research and Manufacturers of America (http://phrma.org)

THE BIOPHARMACEUTICAL RESEARCH AND DEVELOPMENT PROCESS

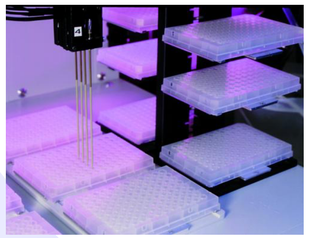


Source: Pharmaceutical Research and Manufacturers of America (http://phrma.org)

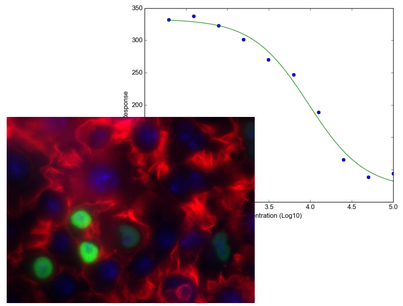
1. Does the compound do what you want it to?
2. Does the compound **not** do what you **don't** want it to?
3. Is what you want it to do the right thing?

Drug Discovery

Target Identification



Screening



Lead Identification

Lead Optimization

Compounds **Hits** **Leads** **Clinical Candidates**

Cost

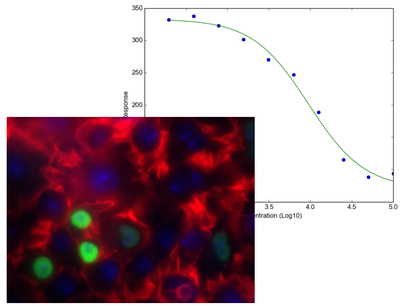
Computational Drug Discovery

Omics

Target Identification

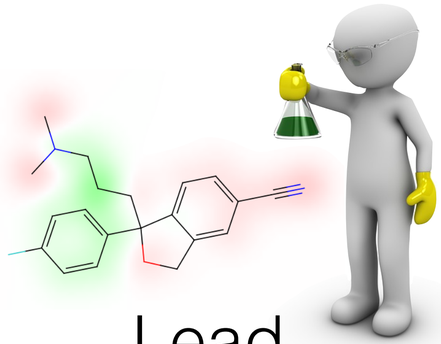
Virtual

Screening



Lead Identification

Modeling



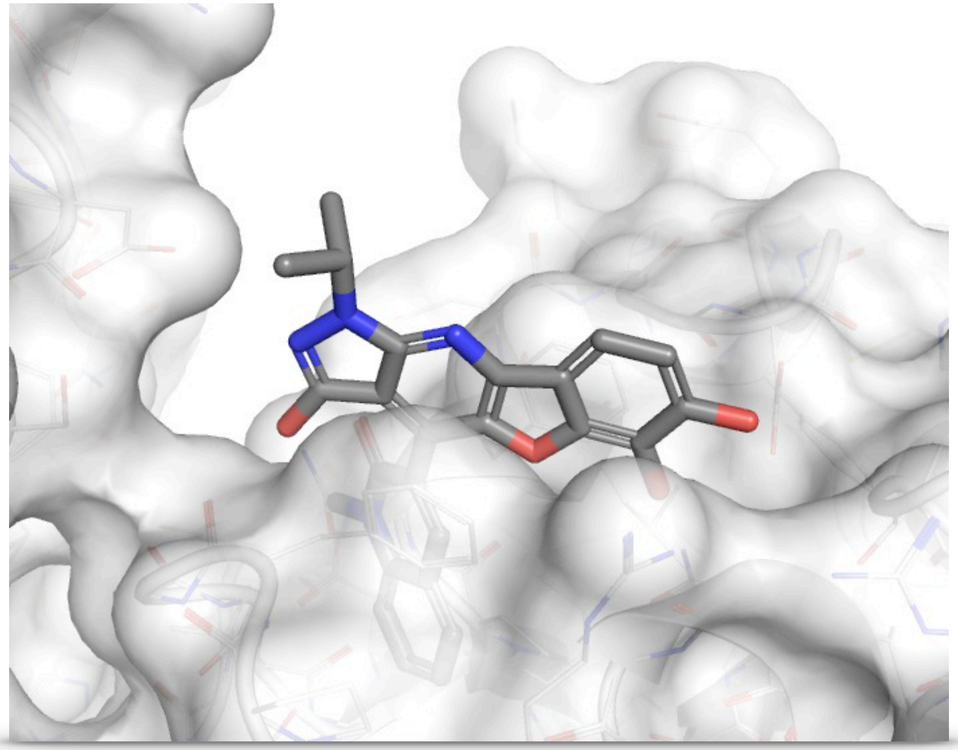
Lead Optimization

Compounds **Hits** **Leads** **Clinical Candidates**

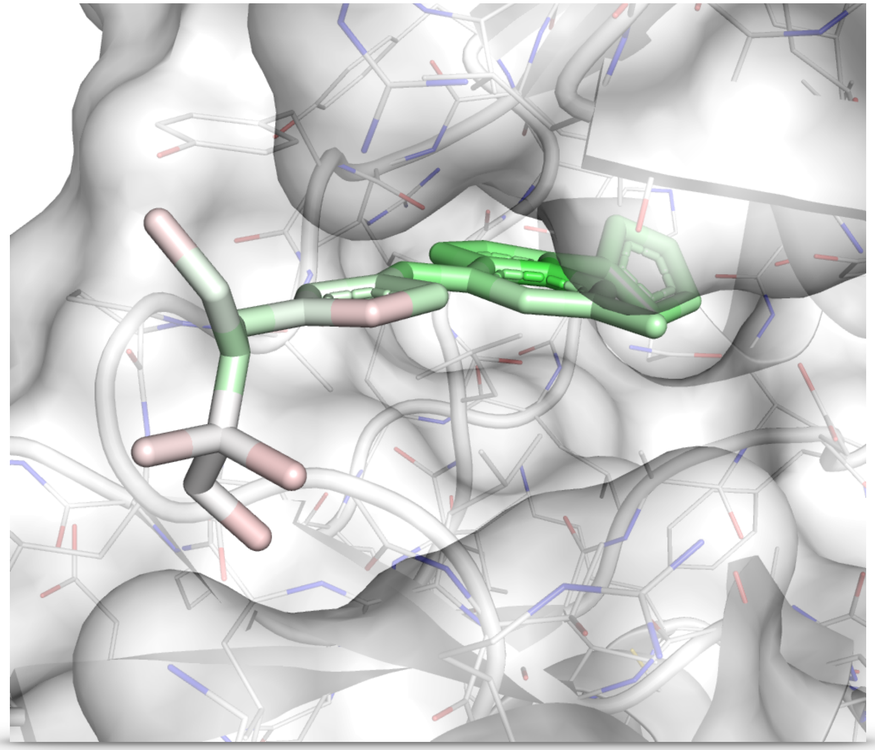
Cost

Structure Based Drug Design

Pose Prediction Binding Discrimination Affinity Prediction



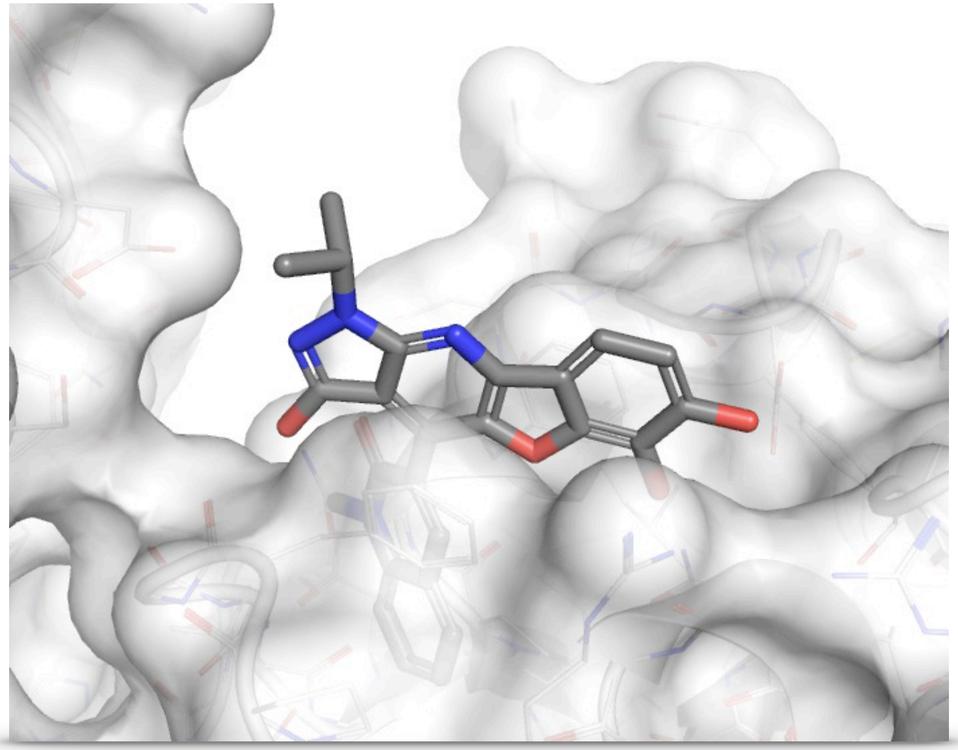
Virtual Screening



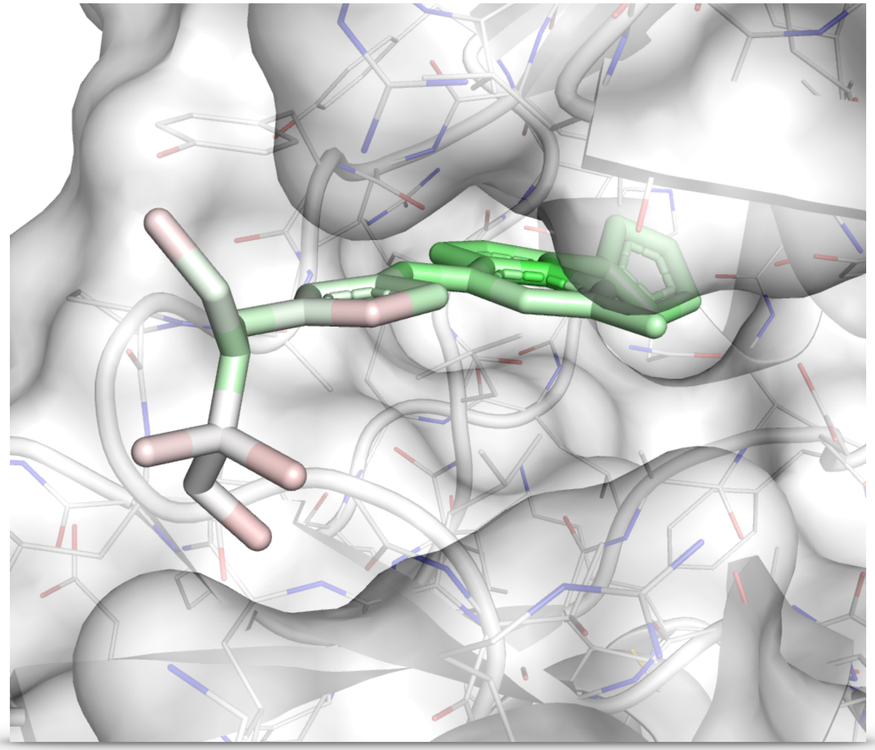
Lead Optimization

Structure Based Drug Design

Pose Prediction Binding Discrimination Affinity Prediction

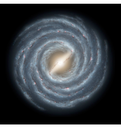


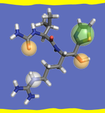
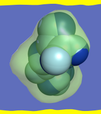
Virtual Screening

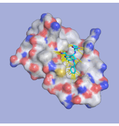
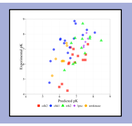


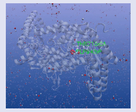
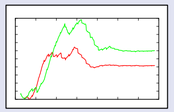
Lead Optimization

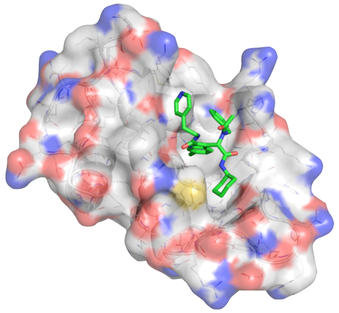
Drug Discovery Funnel

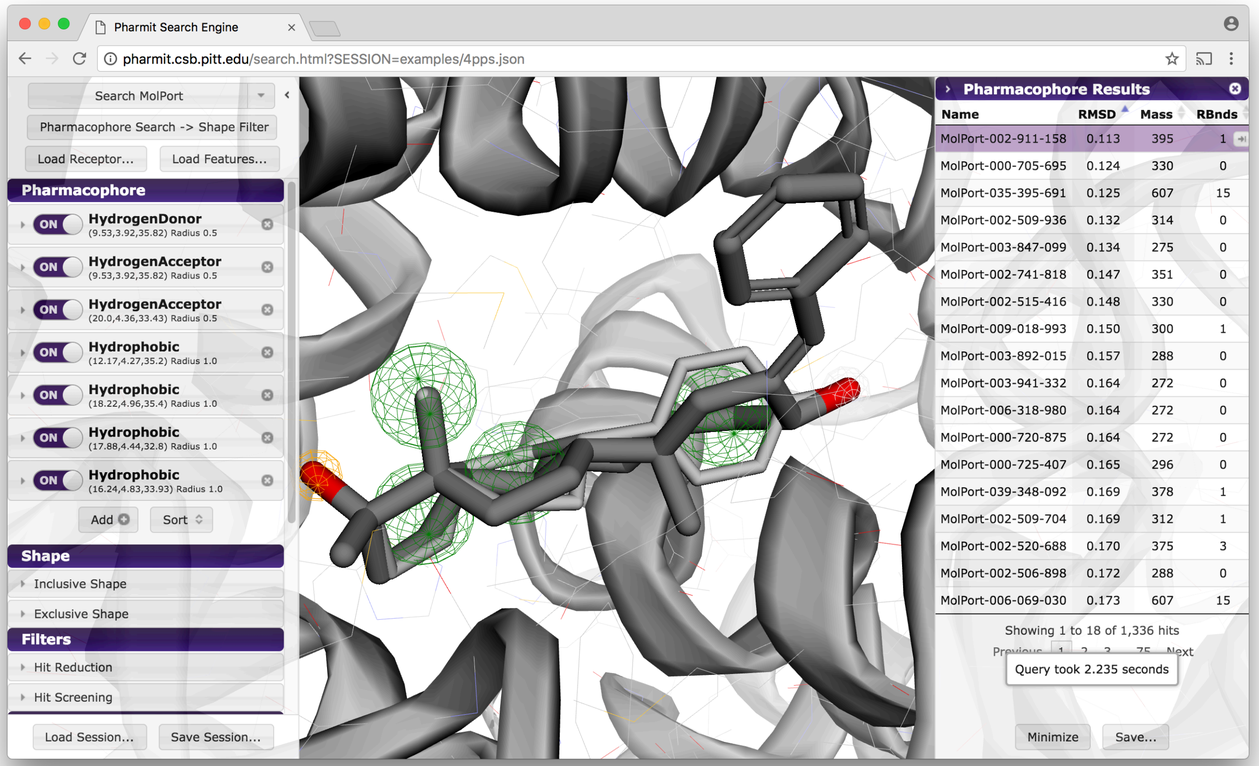
Purchasable  Accessible


Matching



Scoring


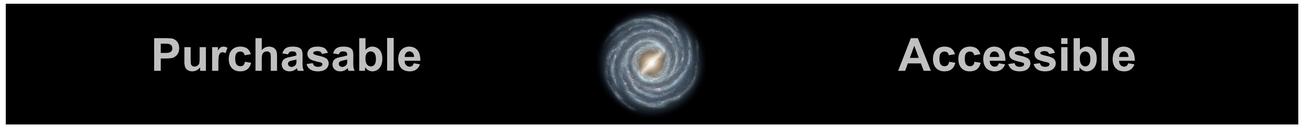

Dynamics


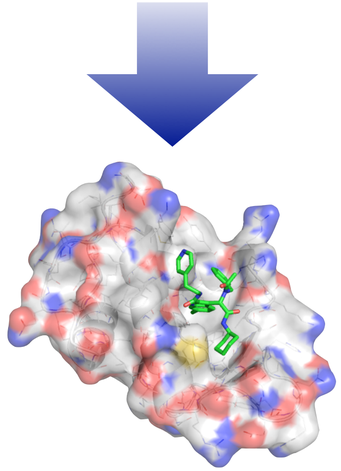


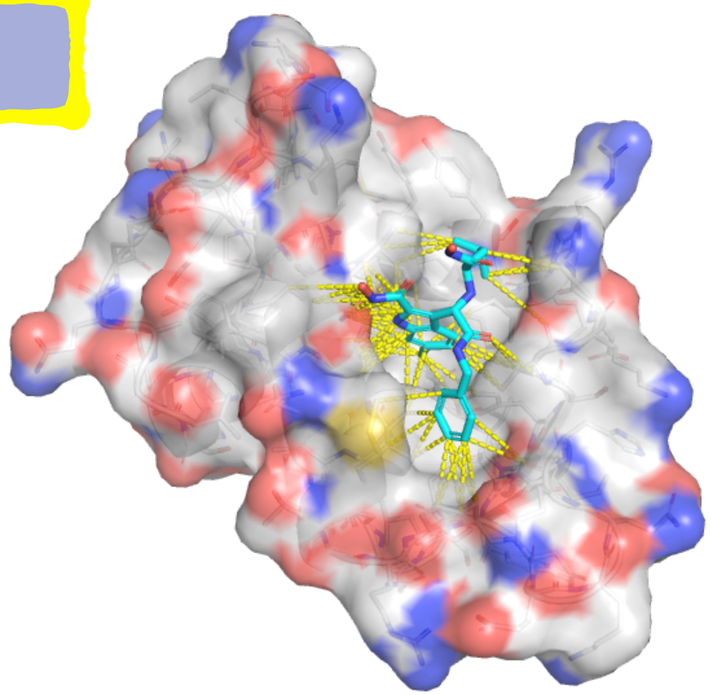


http://pharmit.csb.pitt.edu

Drug Discovery Funnel



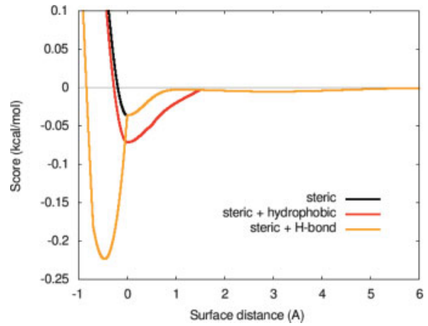




$$\text{gauss}_1(d) = w_{\text{guass}_1} e^{-(d/0.5)^2}$$
$$\text{gauss}_2(d) = w_{\text{guass}_2} e^{-((d-3)/2)^2}$$
$$\text{repulsion}(d) = \begin{cases} w_{\text{repulsion}} d^2 & d < 0 \\ 0 & d \geq 0 \end{cases}$$

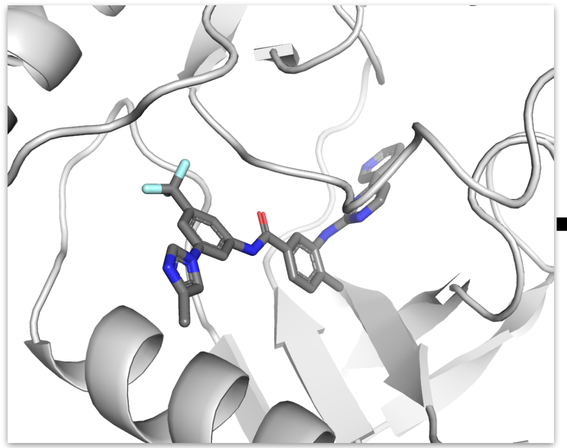
$$\text{hydrophobic}(d) = \begin{cases} w_{\text{hydrophobic}} & d < 0.5 \\ 0 & d > 1.5 \\ w_{\text{hydrophobic}}(1.5 - d) & otherwise \end{cases}$$

$$\text{hbond}(d) = \begin{cases} w_{\text{hbond}} & d < -0.7 \\ 0 & d > 0 \\ w_{\text{hbond}}(-\frac{10}{7} d) & otherwise \end{cases}$$



O. Trott, A. J. Olson, AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization and multithreading, *Journal of Computational Chemistry* 31 (2010) 455-461

Protein-Ligand Scoring



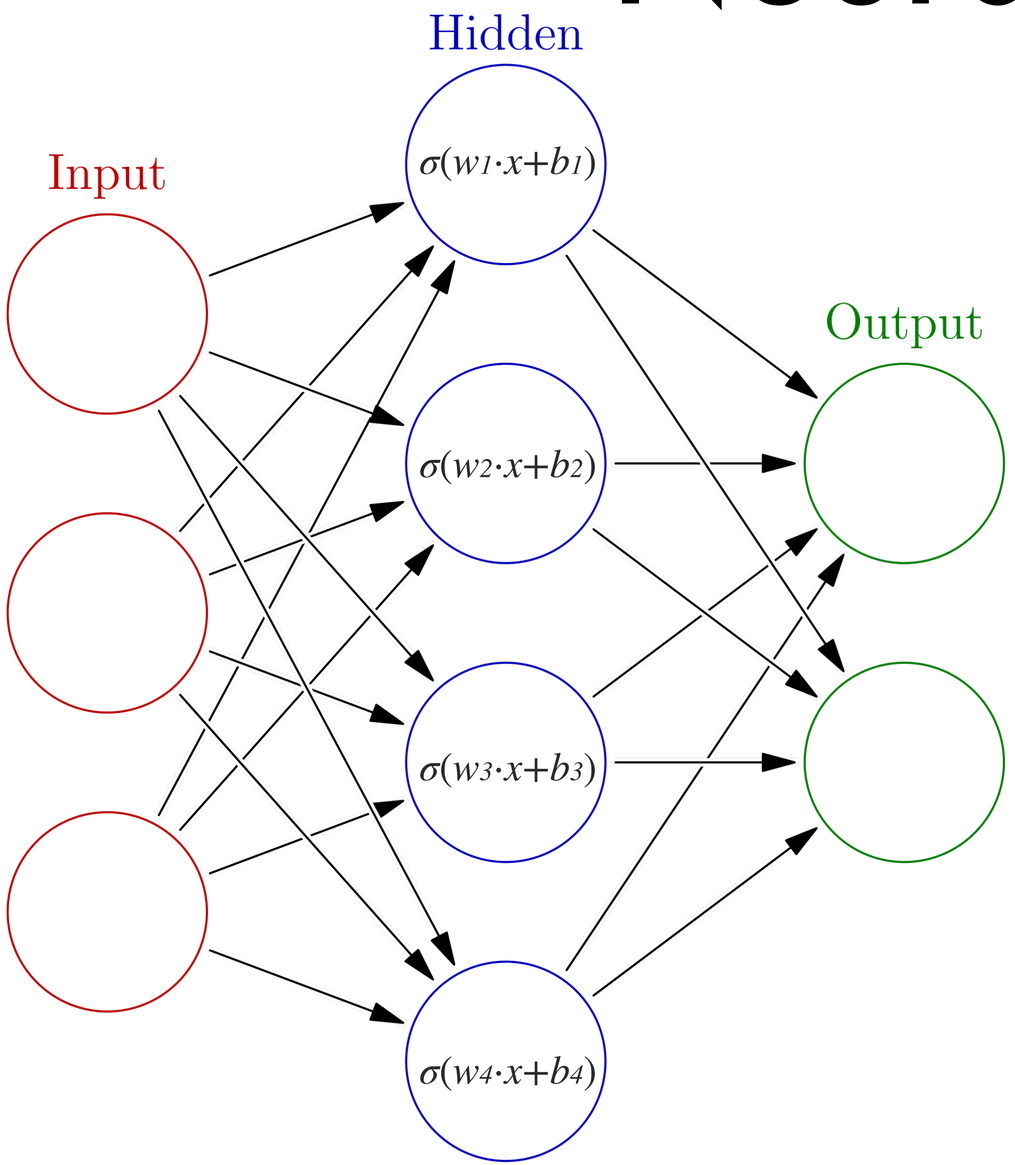
Model

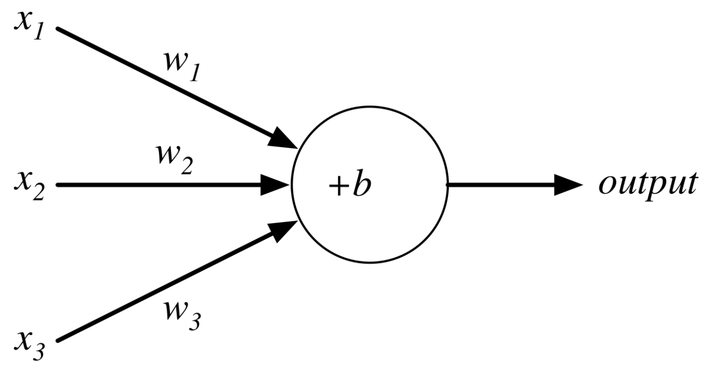
Pose Prediction

Binding Discrimination

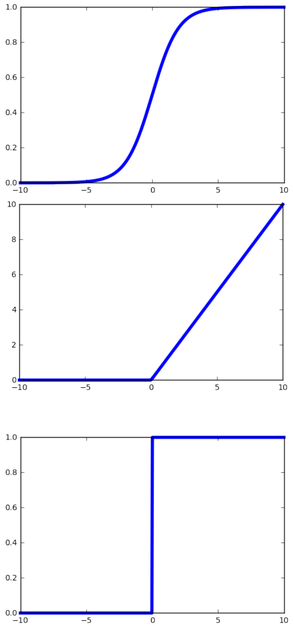
Affinity Prediction

Neural Networks



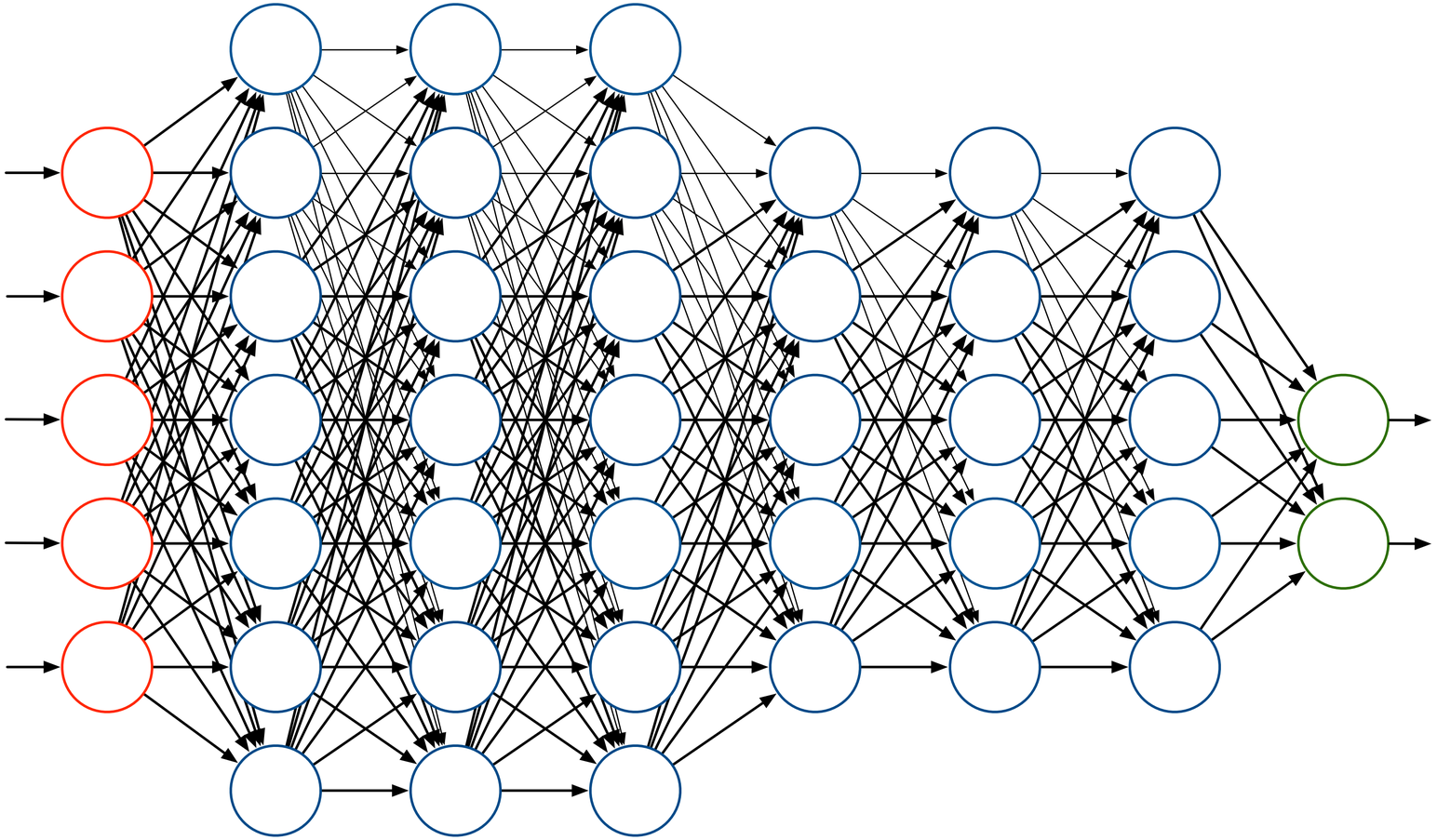


$$output = \sigma\left(\sum_i w_i x_i + b\right)$$

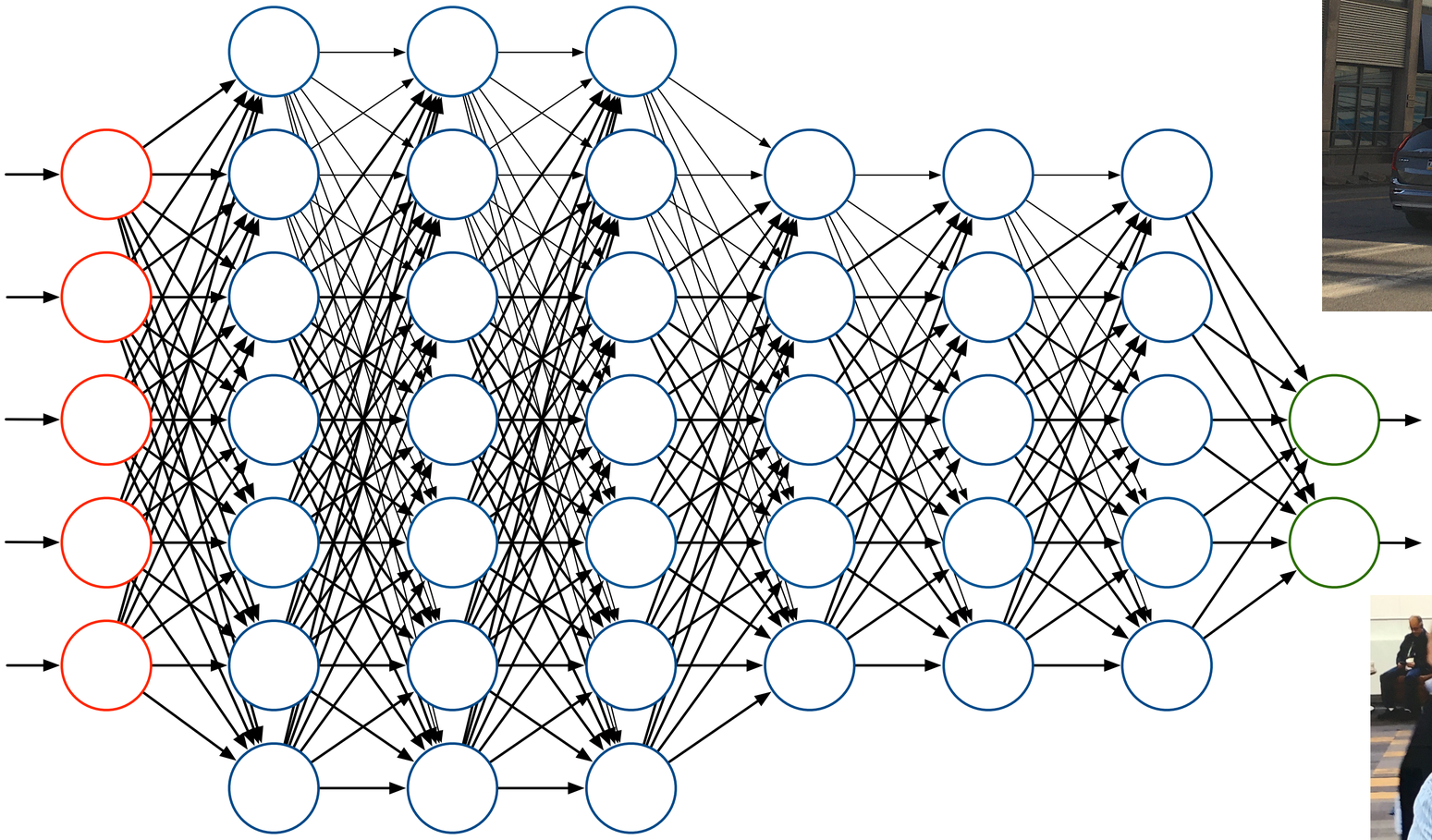


The **universal approximation theorem** states that, under reasonable assumptions, a feedforward **neural network** with a finite number of nodes **can approximate any continuous** function to within a given error over a bounded input domain.

Deep Learning



Deep Learning

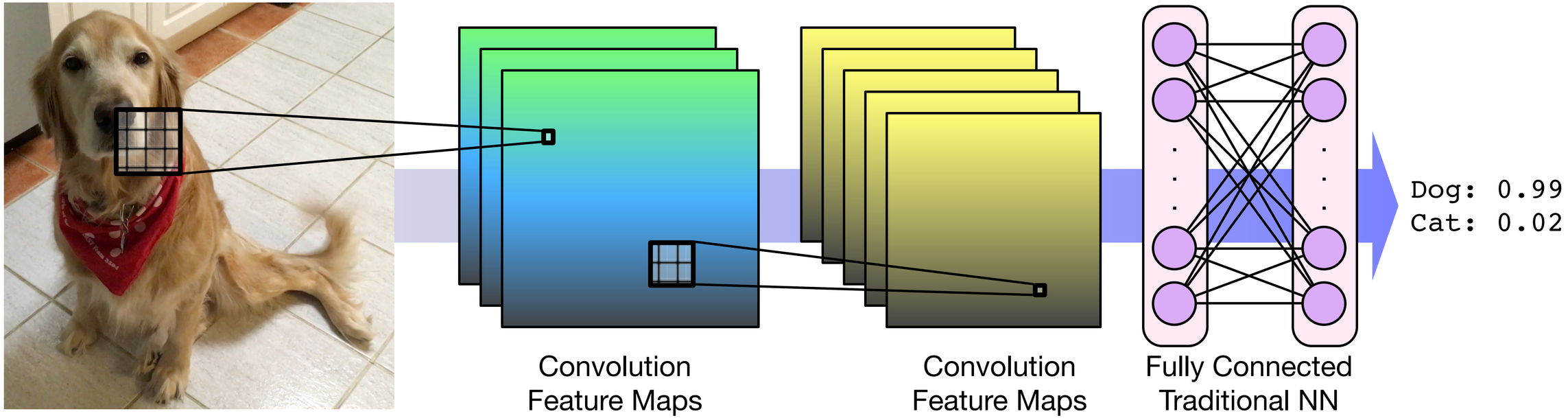


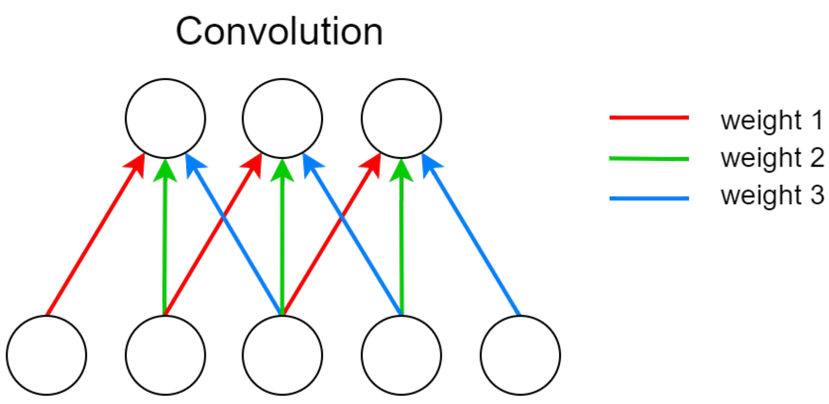
Deep Learning

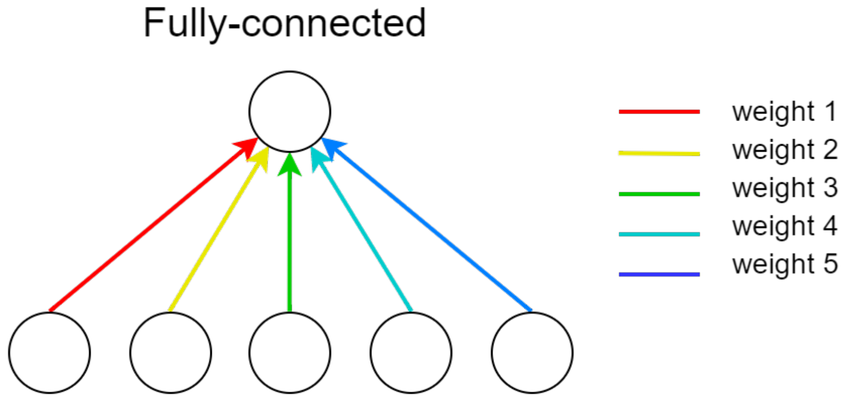
$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$$

$$\frac{\partial L}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j \text{ and } \frac{\partial L}{\partial b^l_j} = \delta^l_j$$

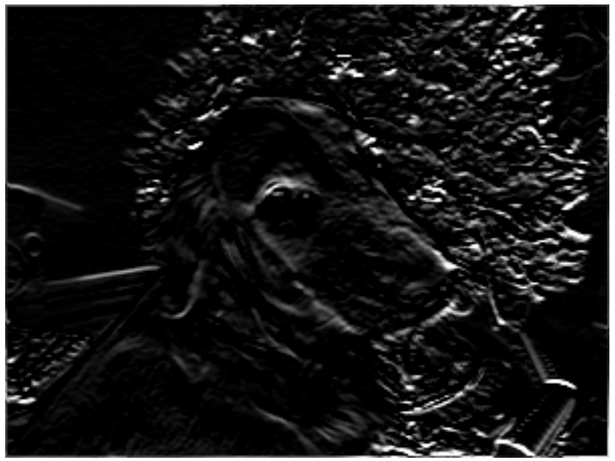
Convolutional Neural Networks



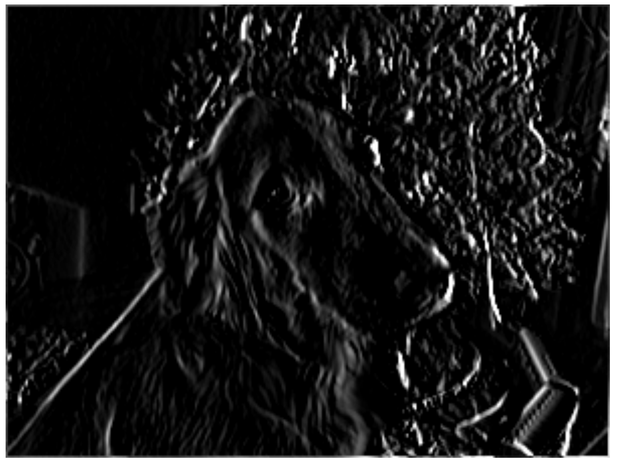




Convolutional Filters



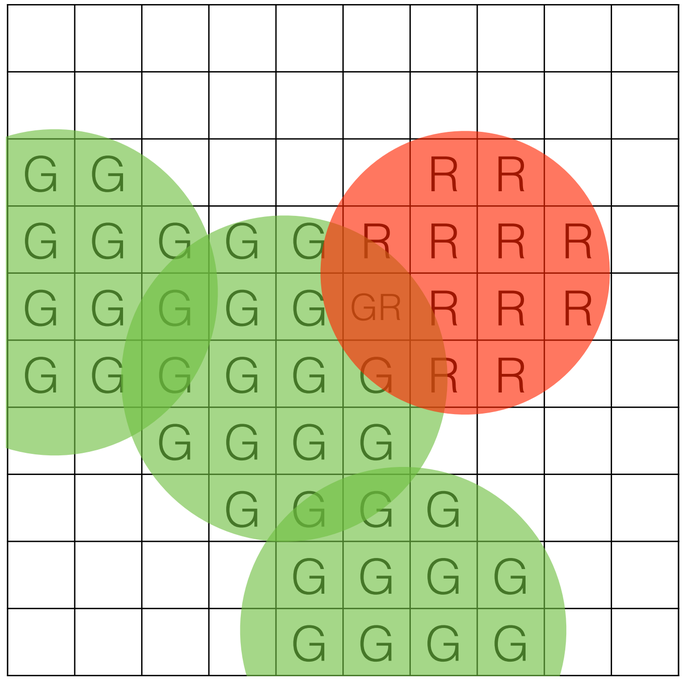
-1	-1	-1
0	0	0
1	1	1



-1	0	1
-1	0	1
-1	0	1

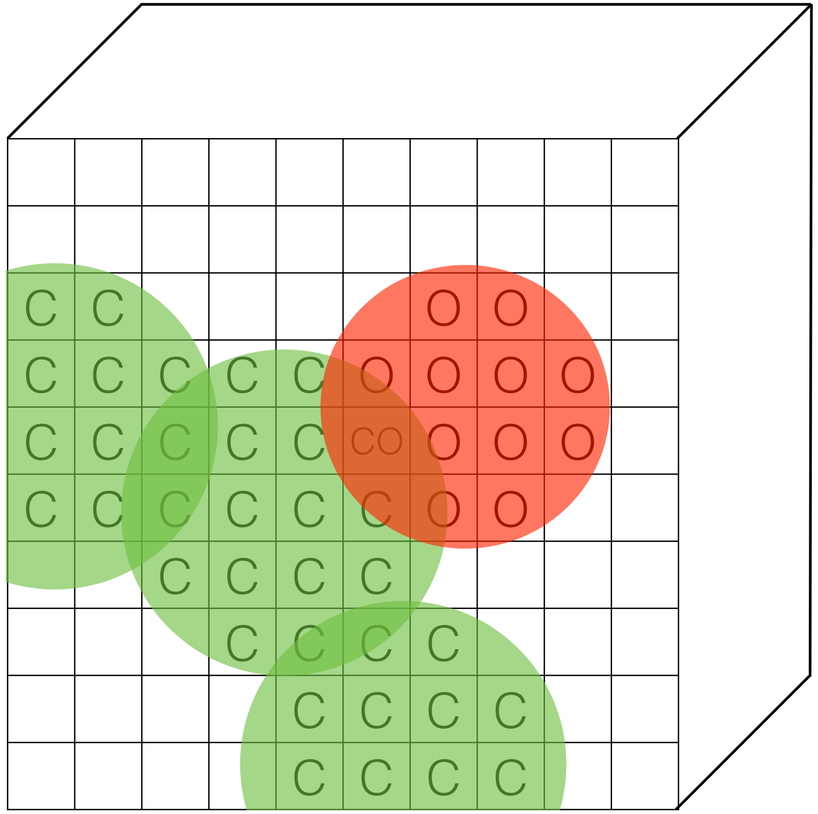
-1	-1	-1
-1	8	-1
-1	-1	-1

Protein-Ligand Representation



(R,G,B) pixel

Protein-Ligand Representation



(R,G,B) pixel →

(Carbon, Nitrogen, Oxygen,…) **voxel**

The only parameters for this representation are the choice of **grid resolution**, **atom density**, and **atom types**.

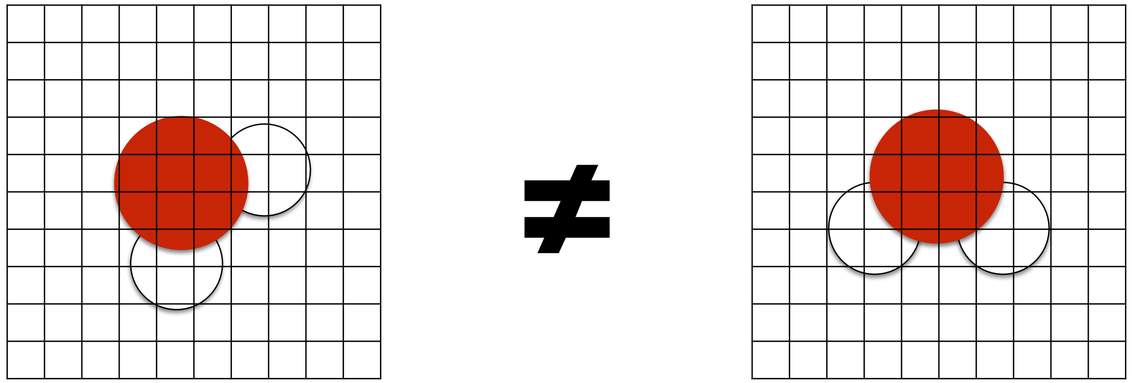
Why Grids?

Cons

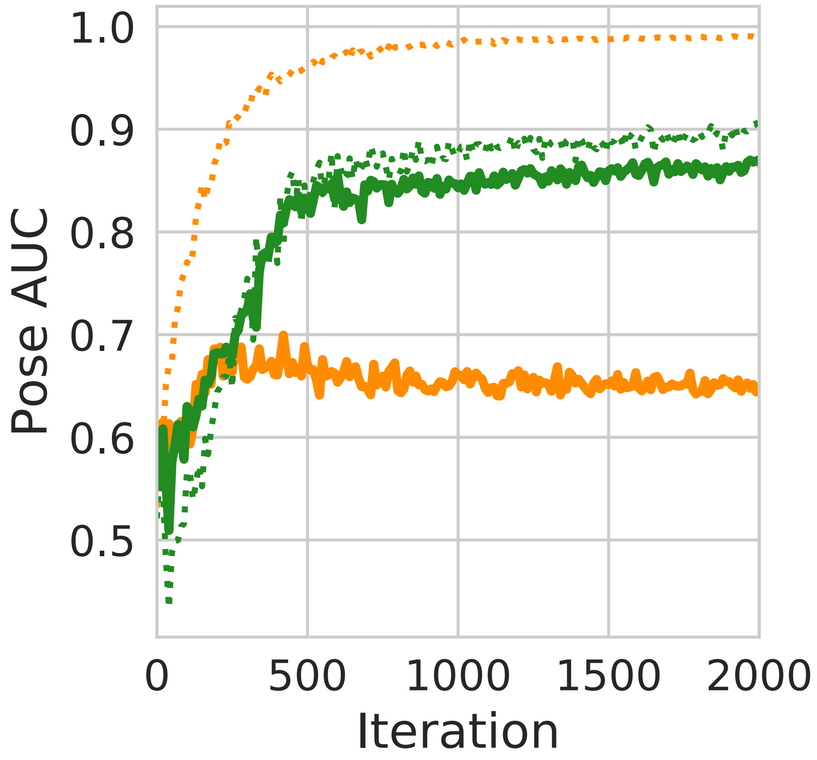
- *coordinate frame dependent*
- pairwise interactions not explicit

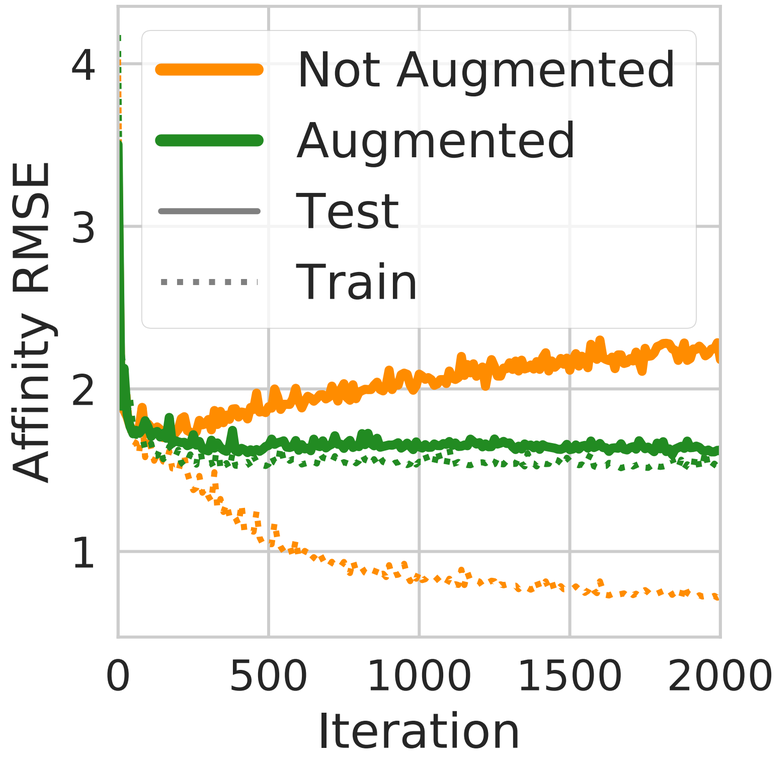
Pros

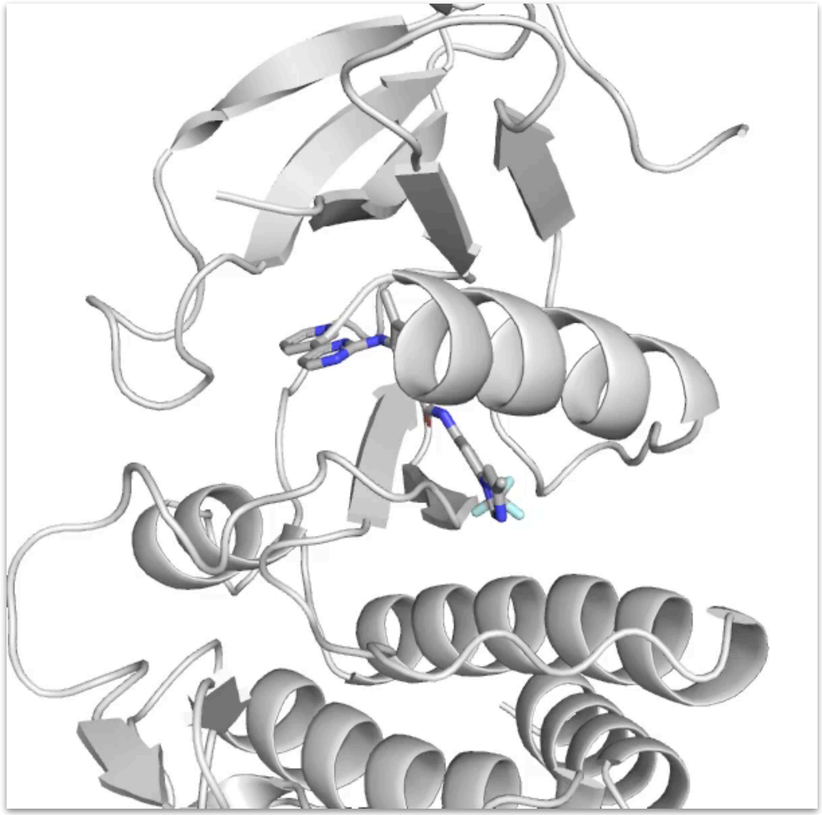
- clear spatial relationships
- amazingly parallel
- easy to interpret



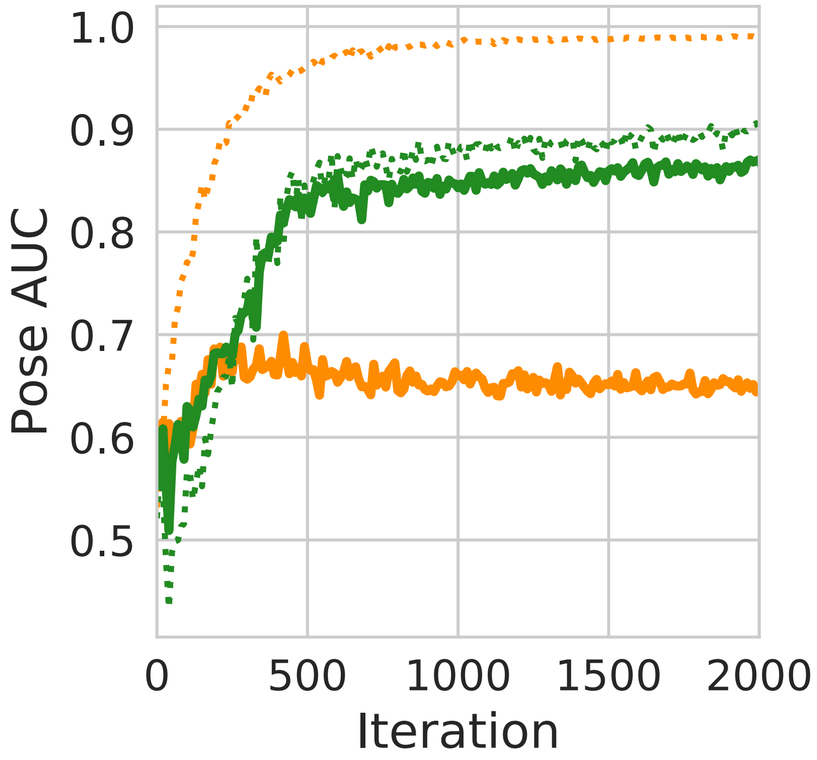
Data Augmentation

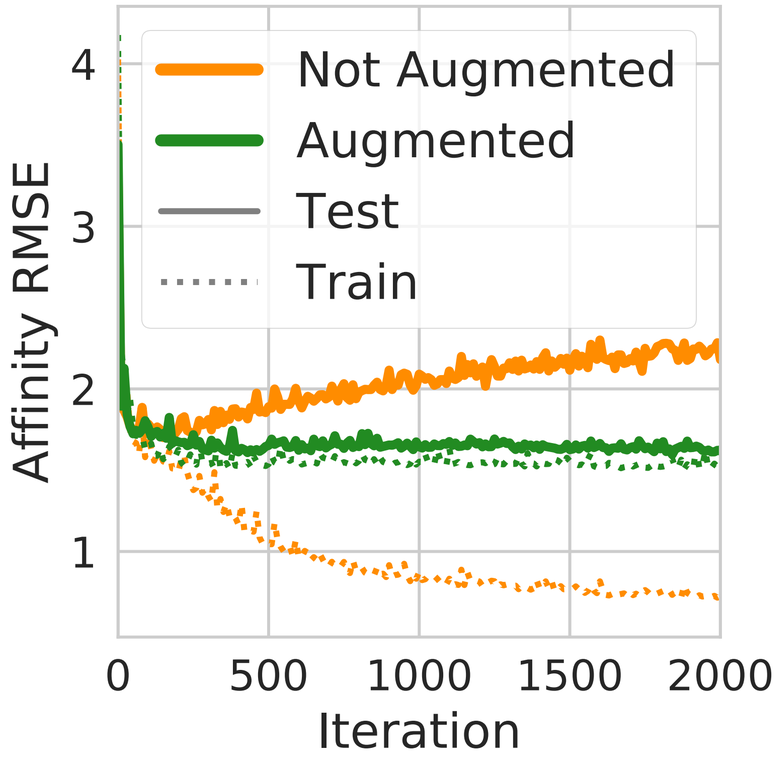


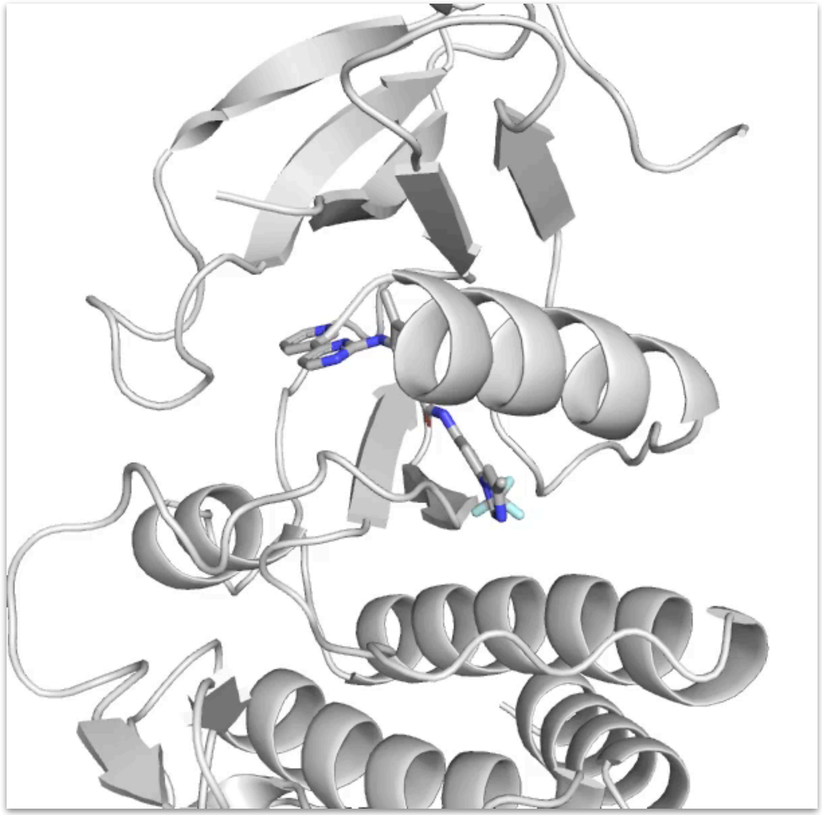




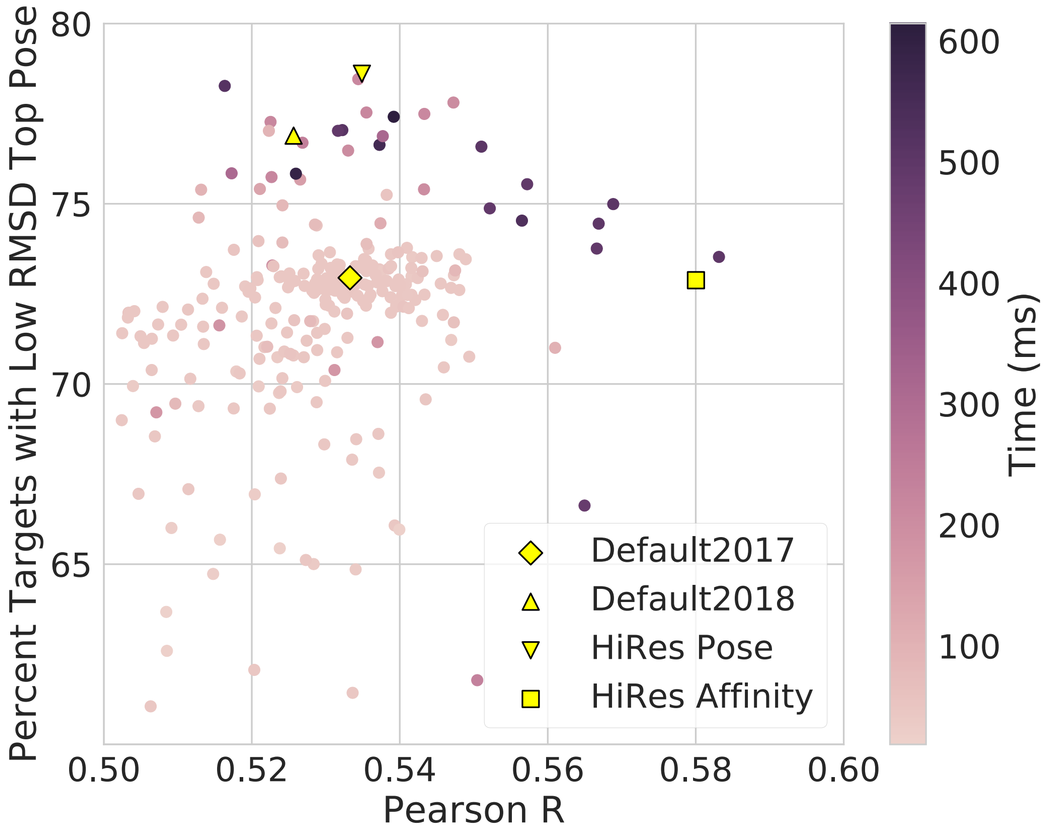
Data Augmentation



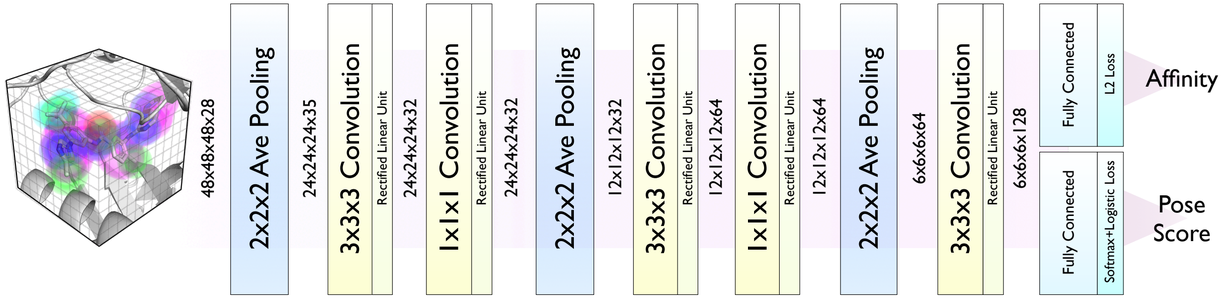




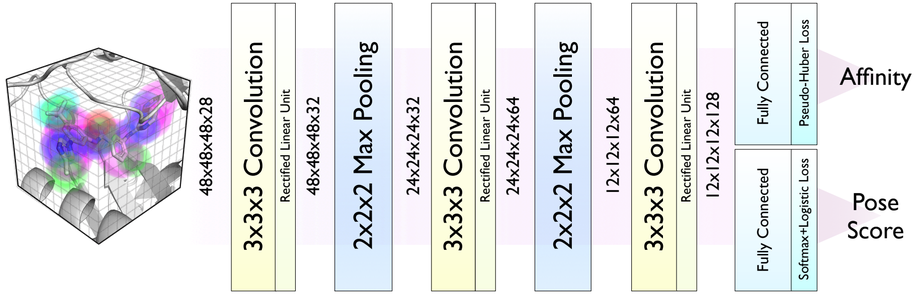
Optimized Models



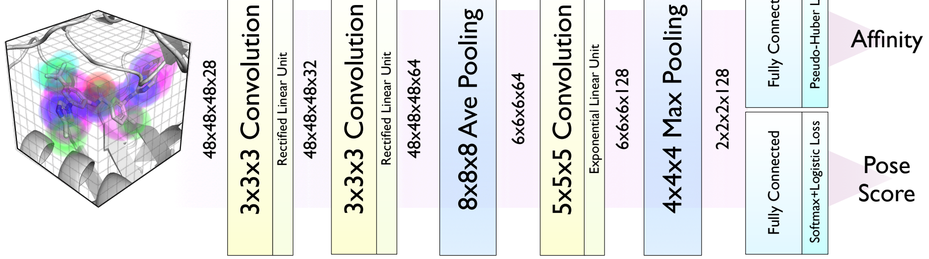
Default2018



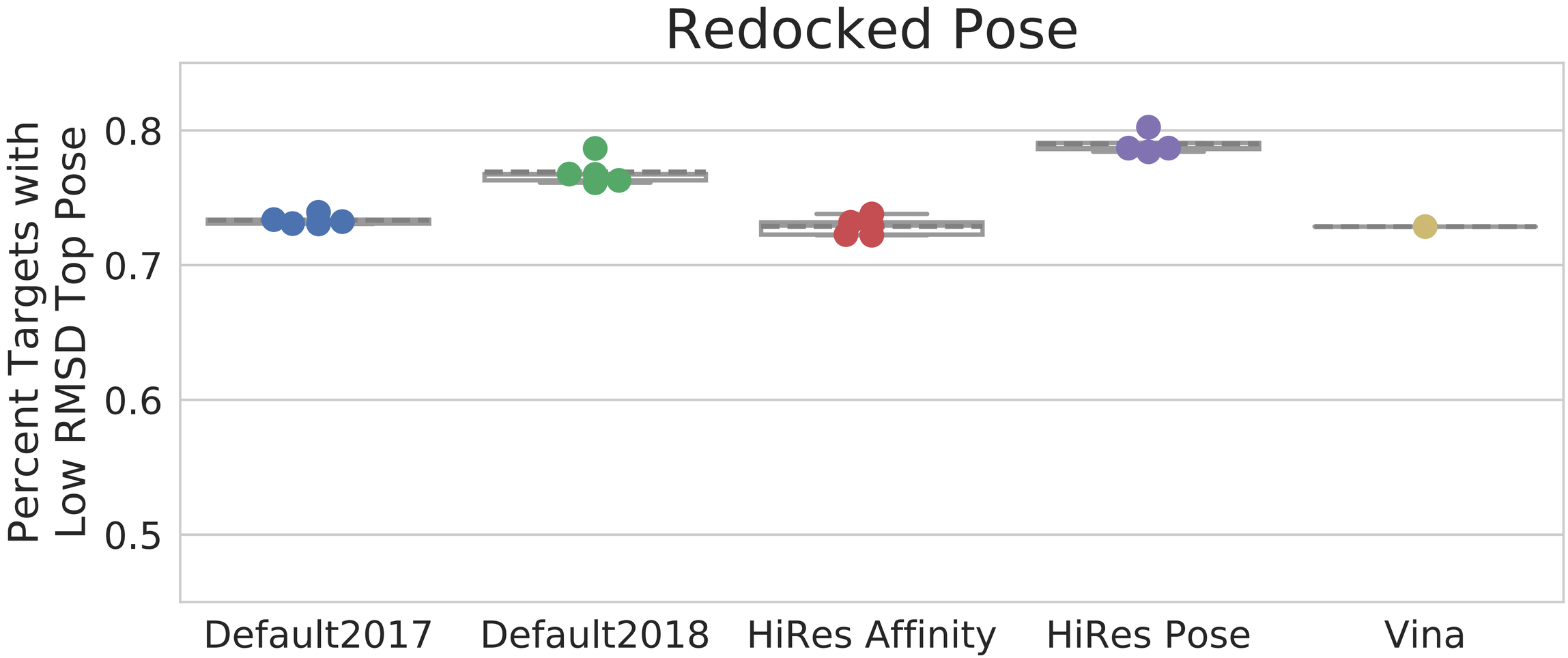
HiRes Pose



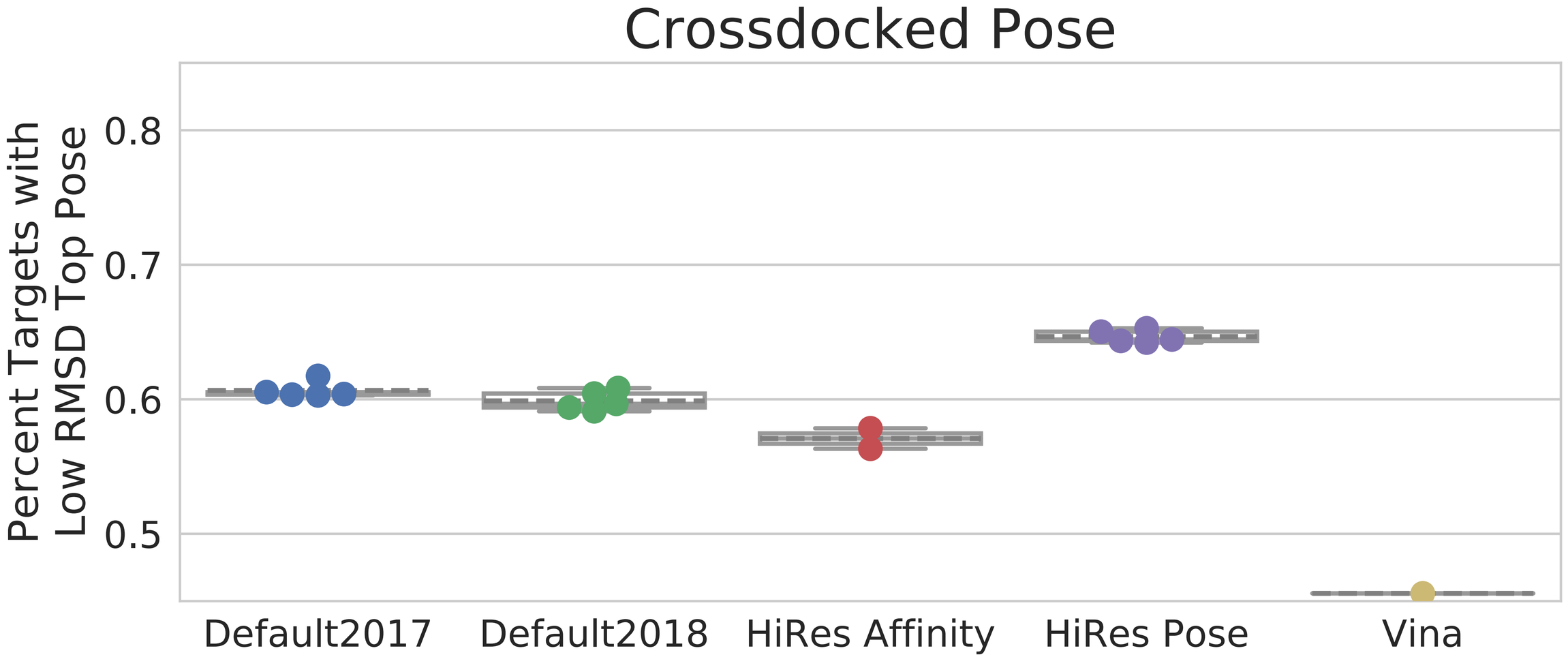
HiRes Affinity



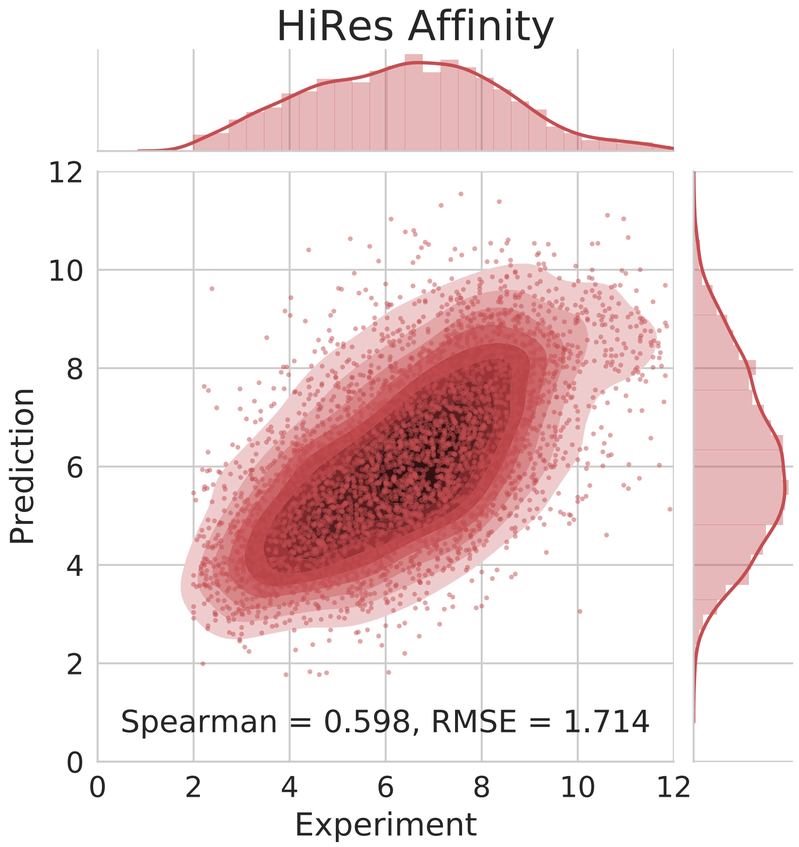
Pose Results

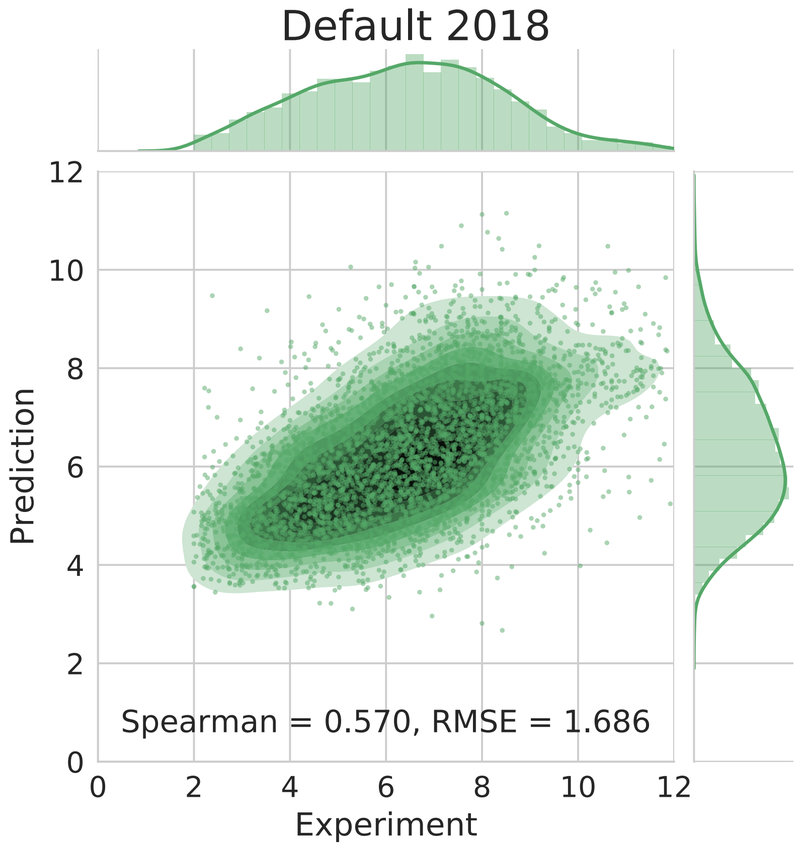


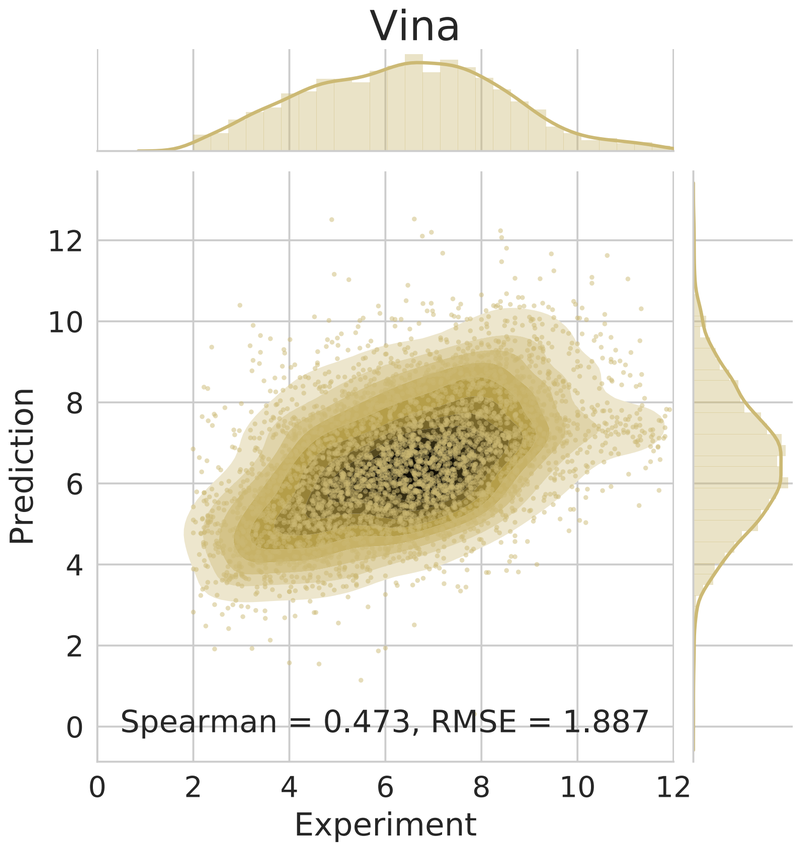
Pose Results



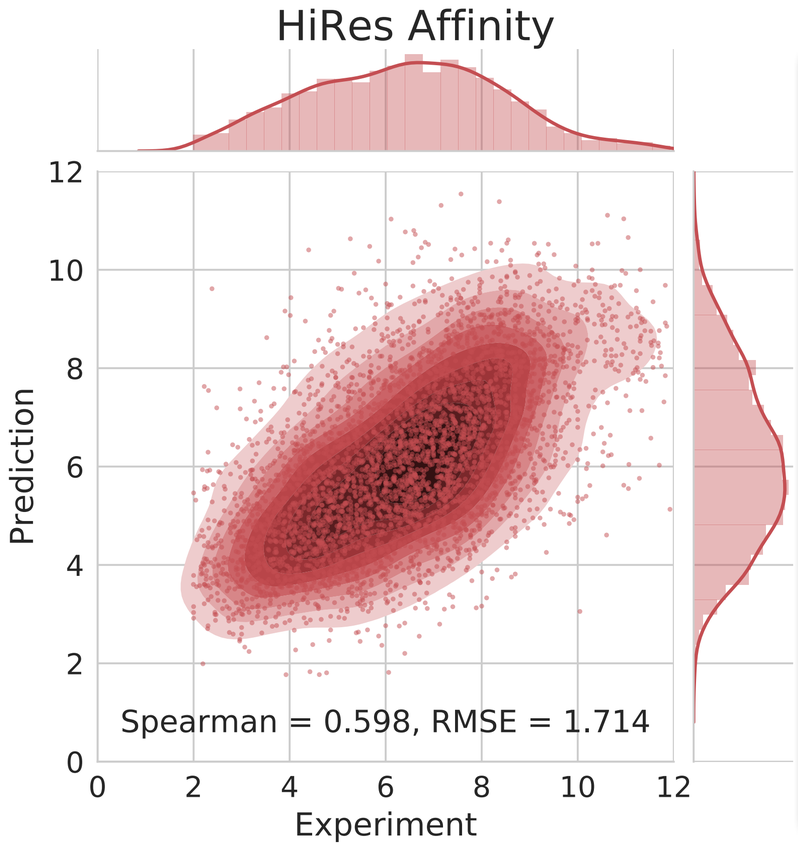
Affinity Results

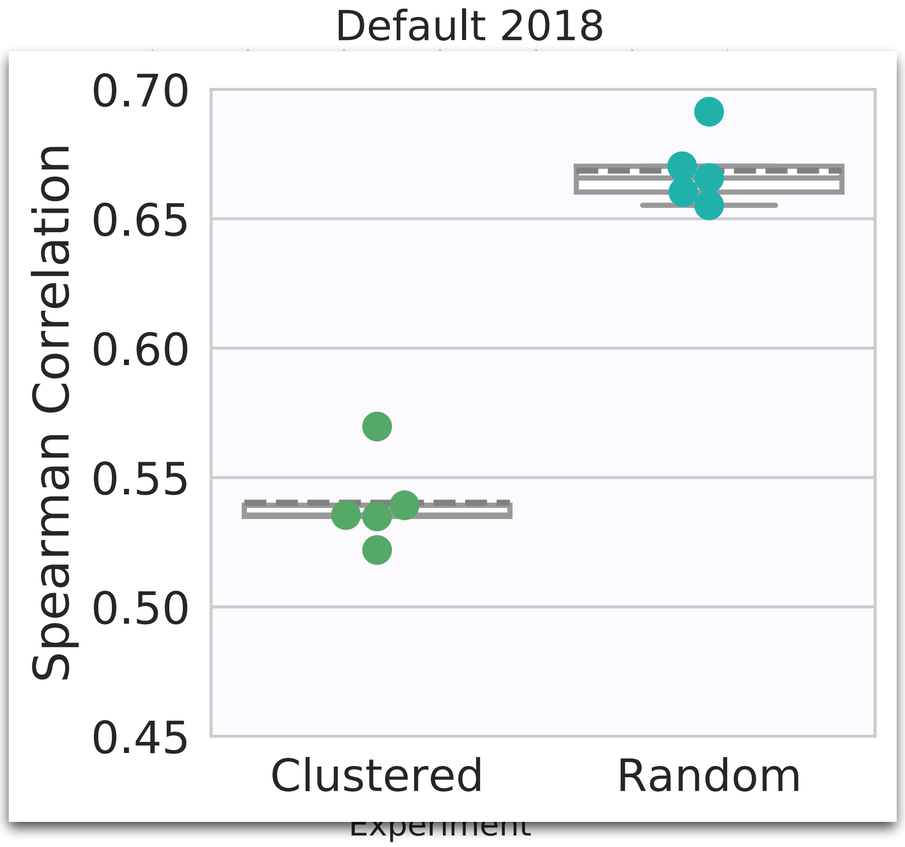


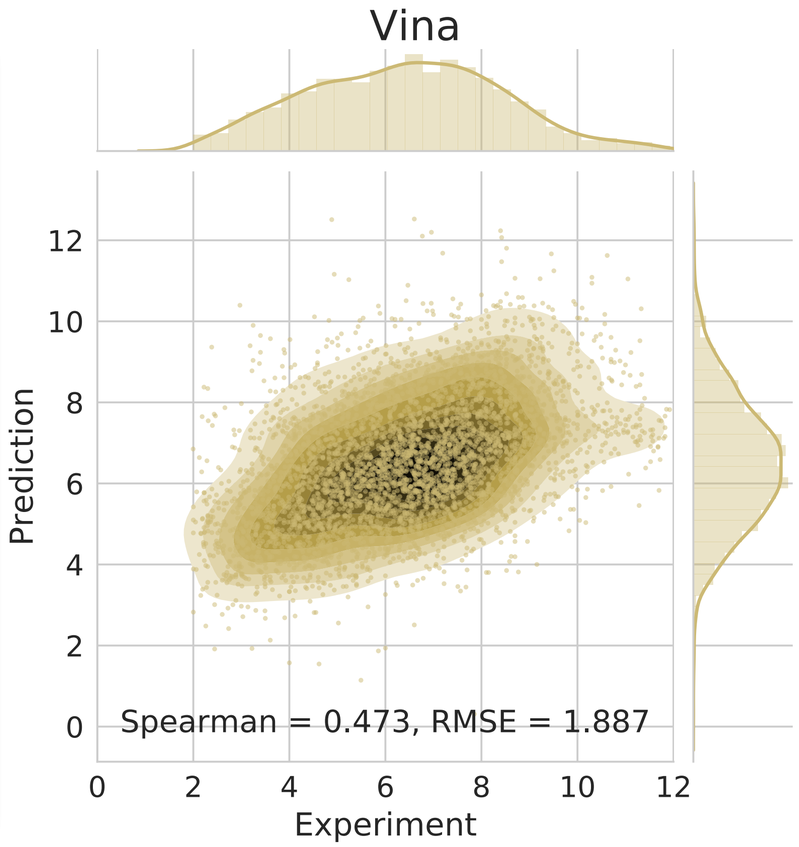




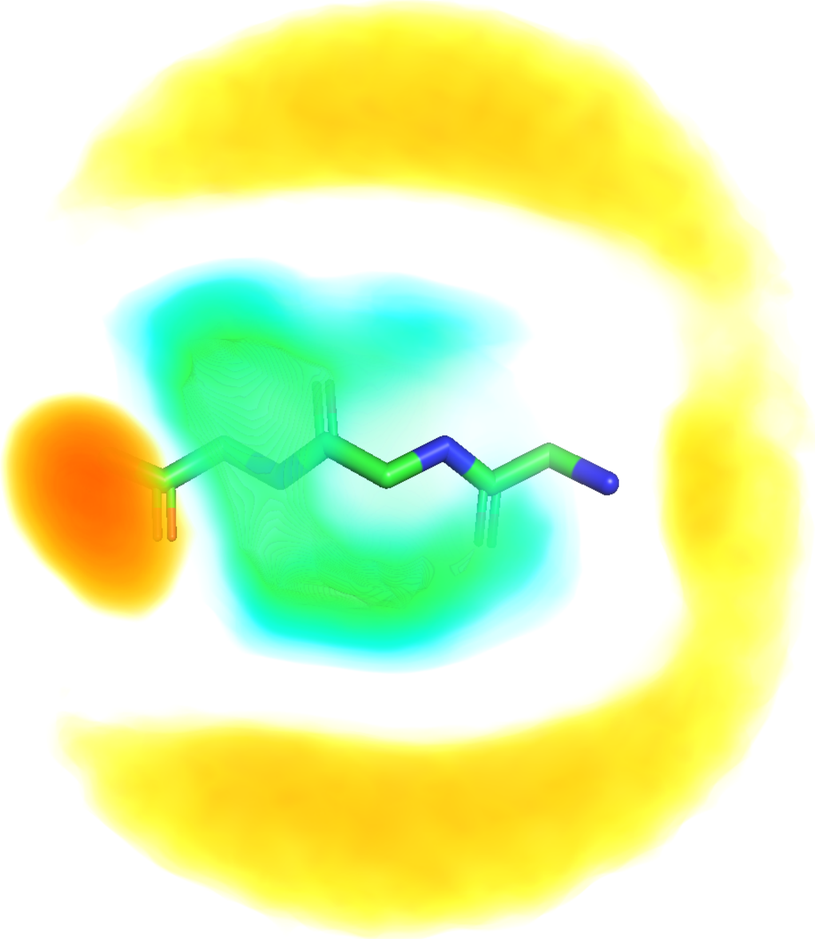
Affinity Results



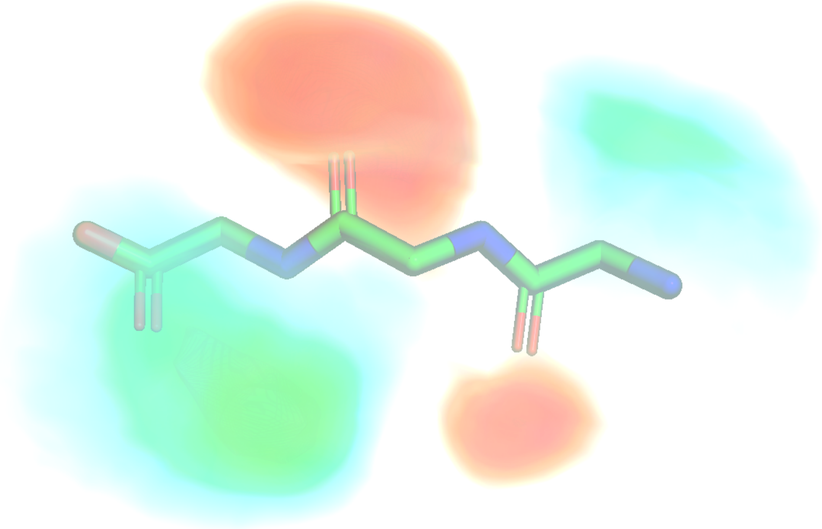




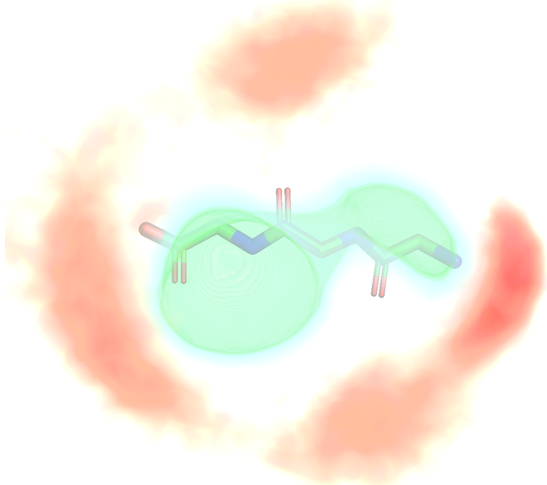
But what is it learning?



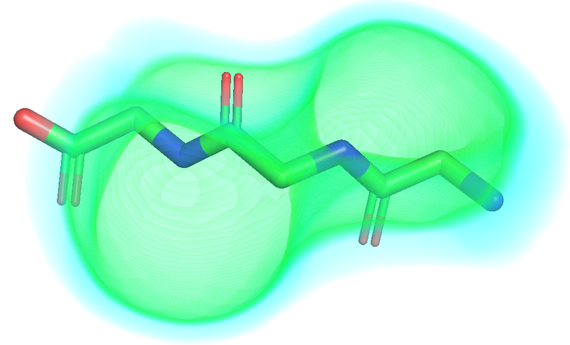
AliphaticCarbon



NitrogenDonor

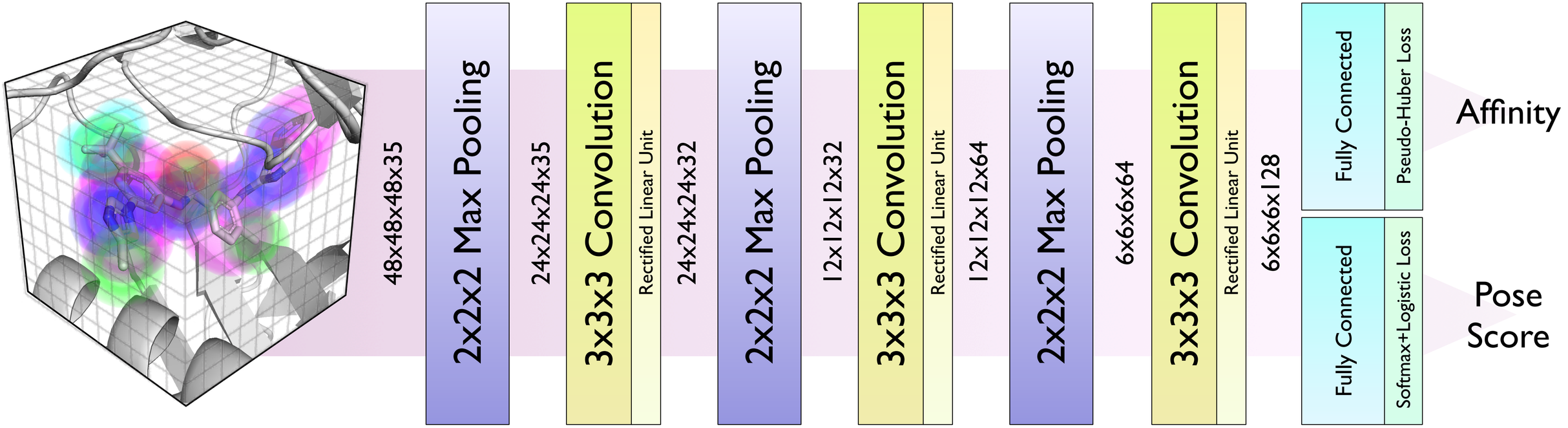


OxygenDonorAcceptor

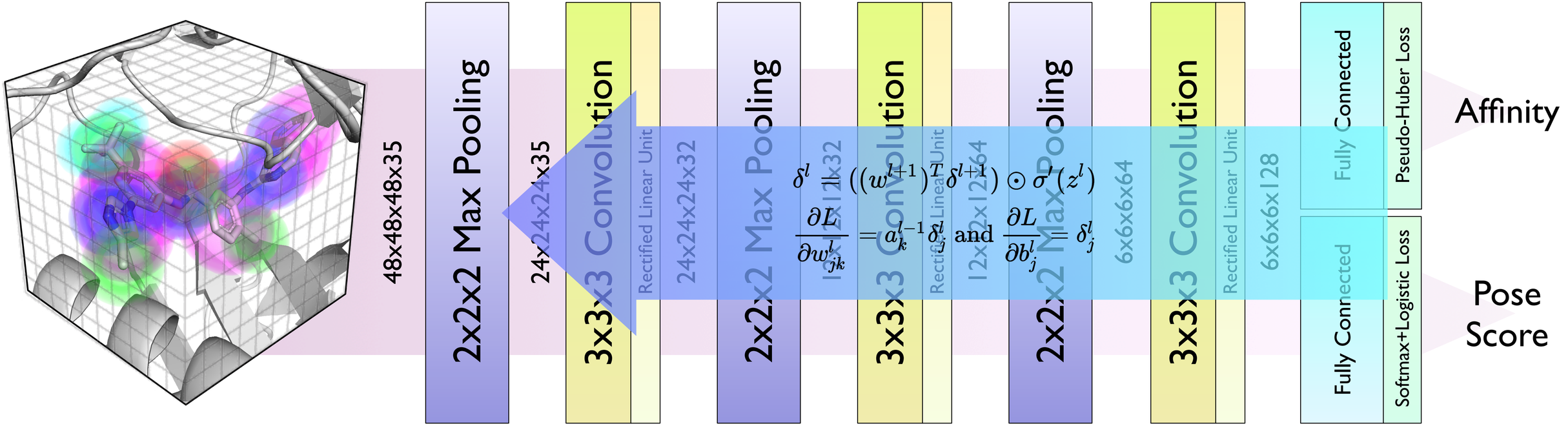


OxygenAcceptor

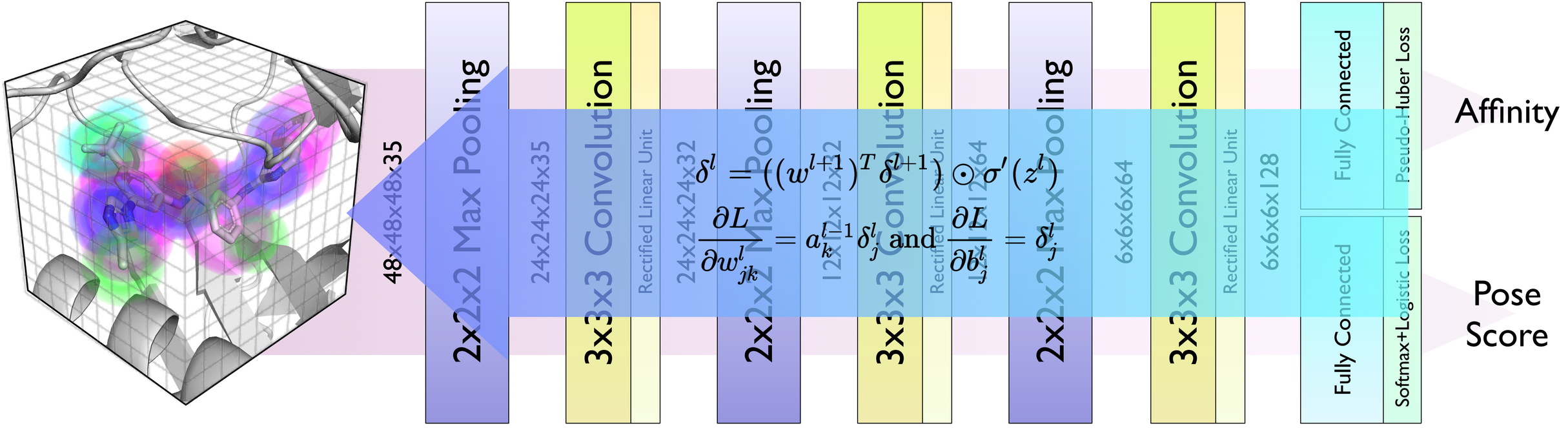
Beyond Scoring



Beyond Scoring



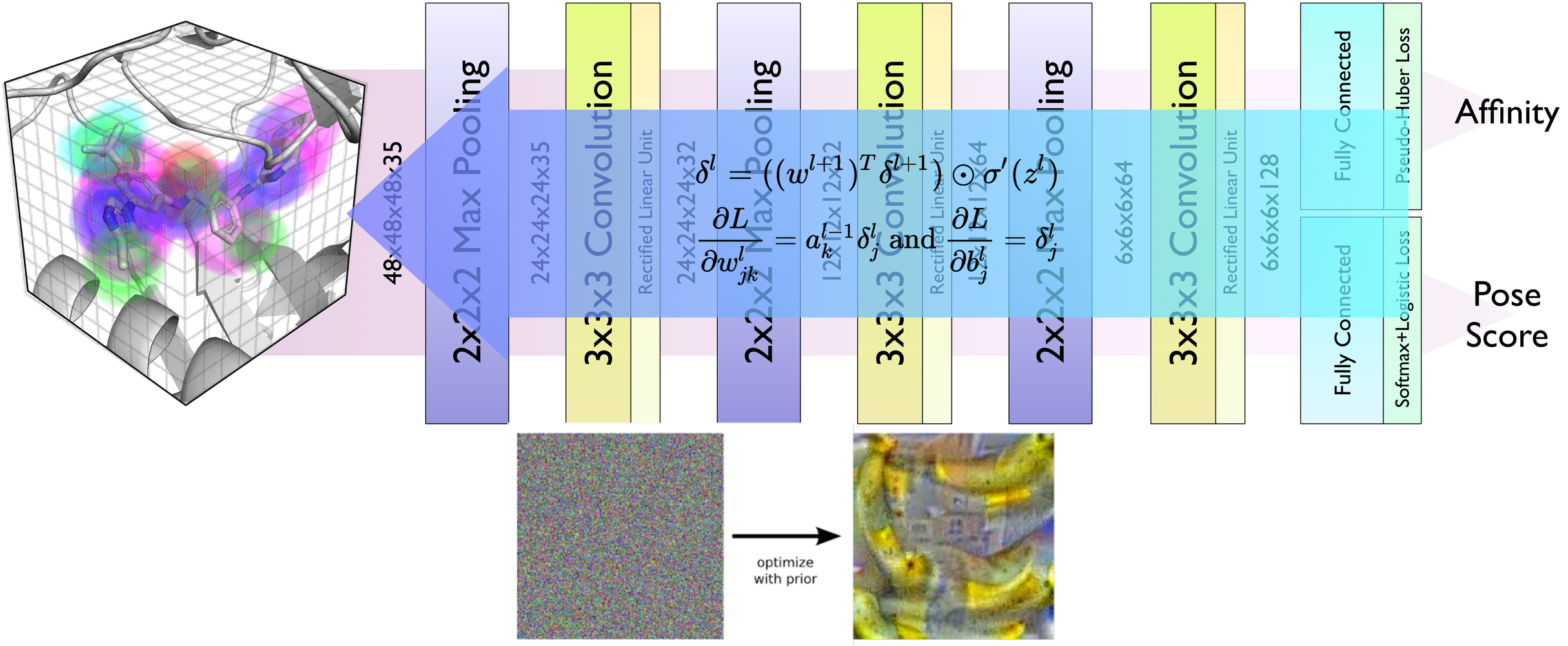
Beyond Scoring



$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$$

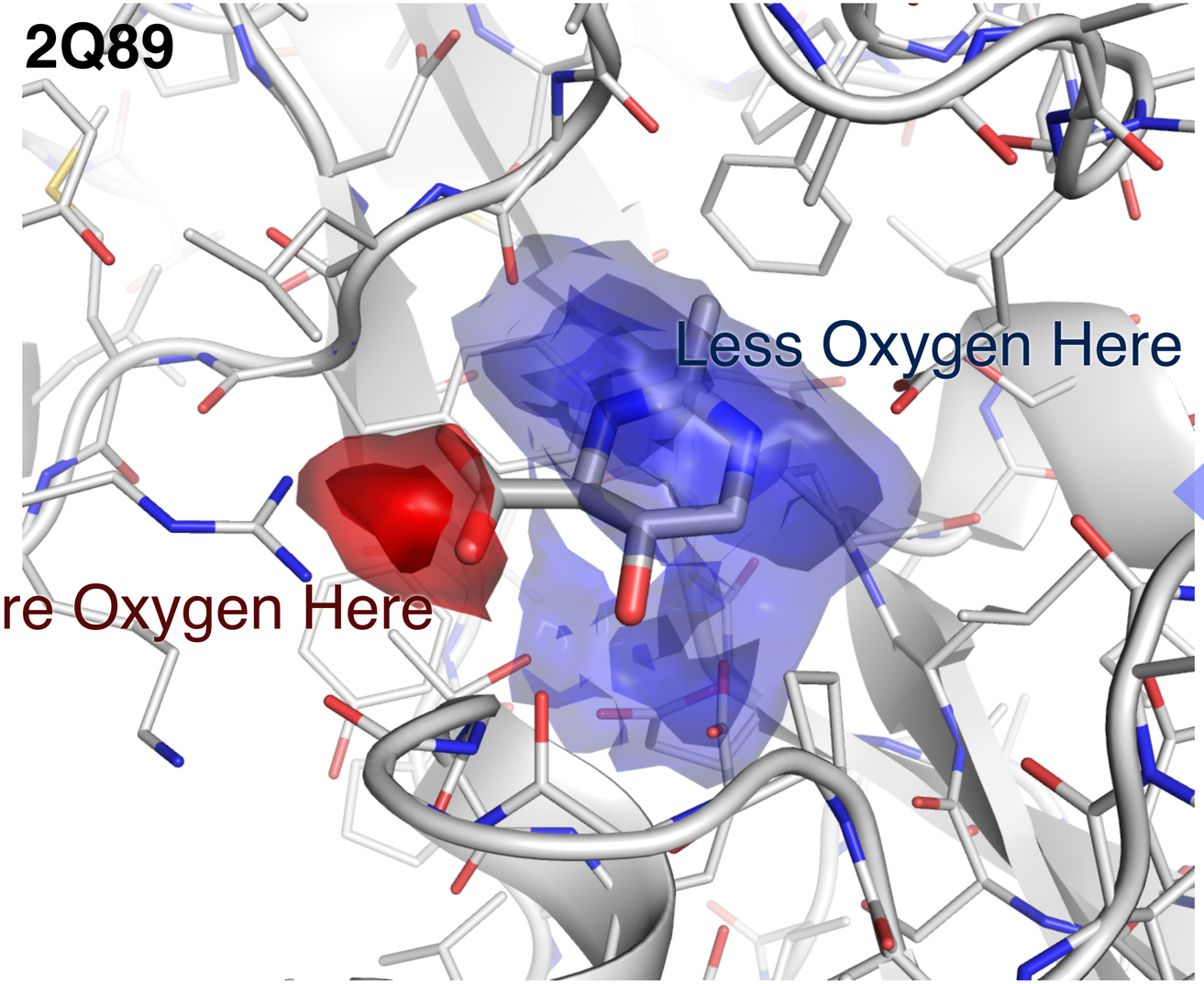
$$\frac{\partial L}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j \text{ and } \frac{\partial L}{\partial b^l_j} = \delta^l_j$$

Beyond Scoring



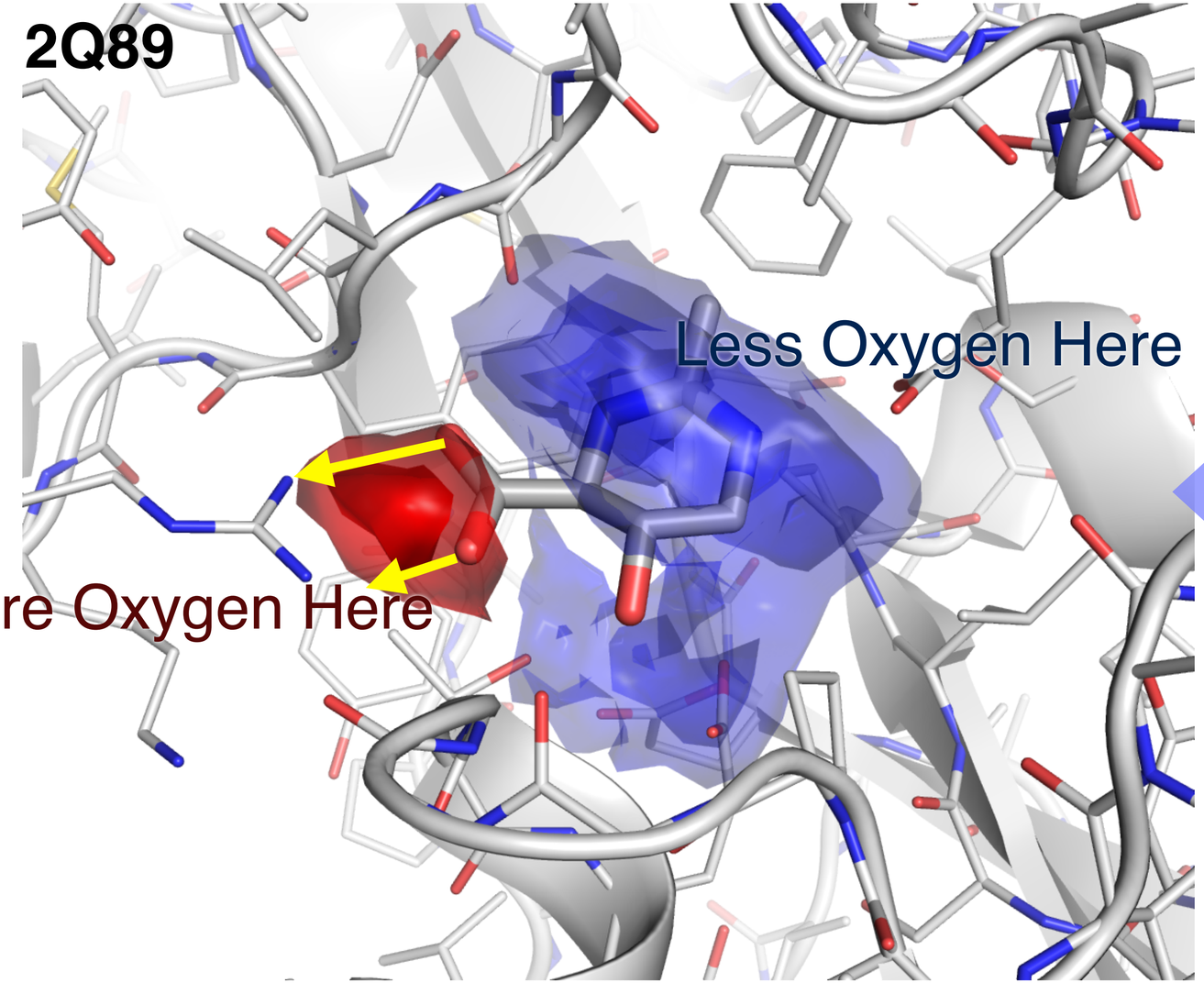
https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

Beyond Scoring

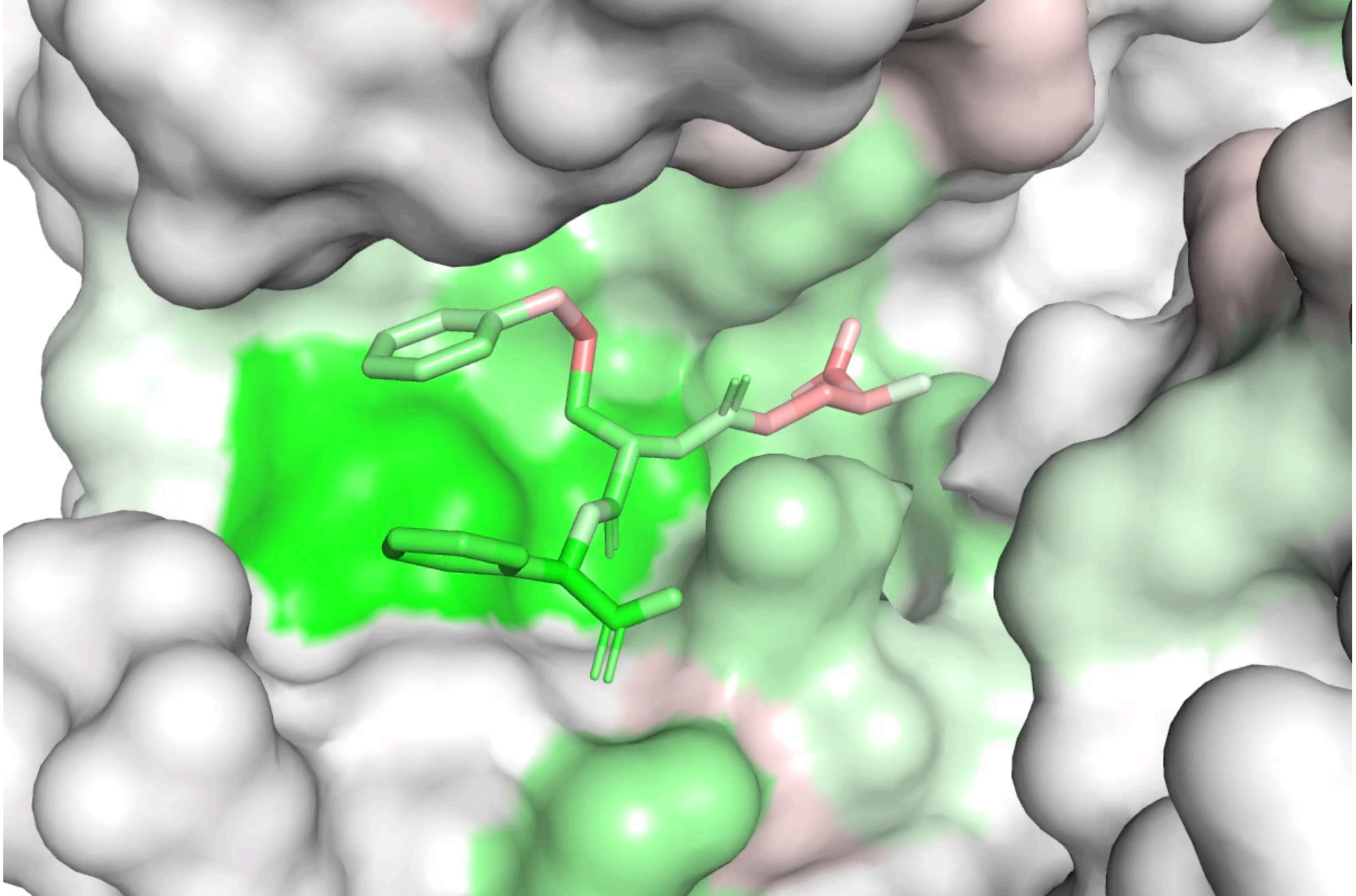


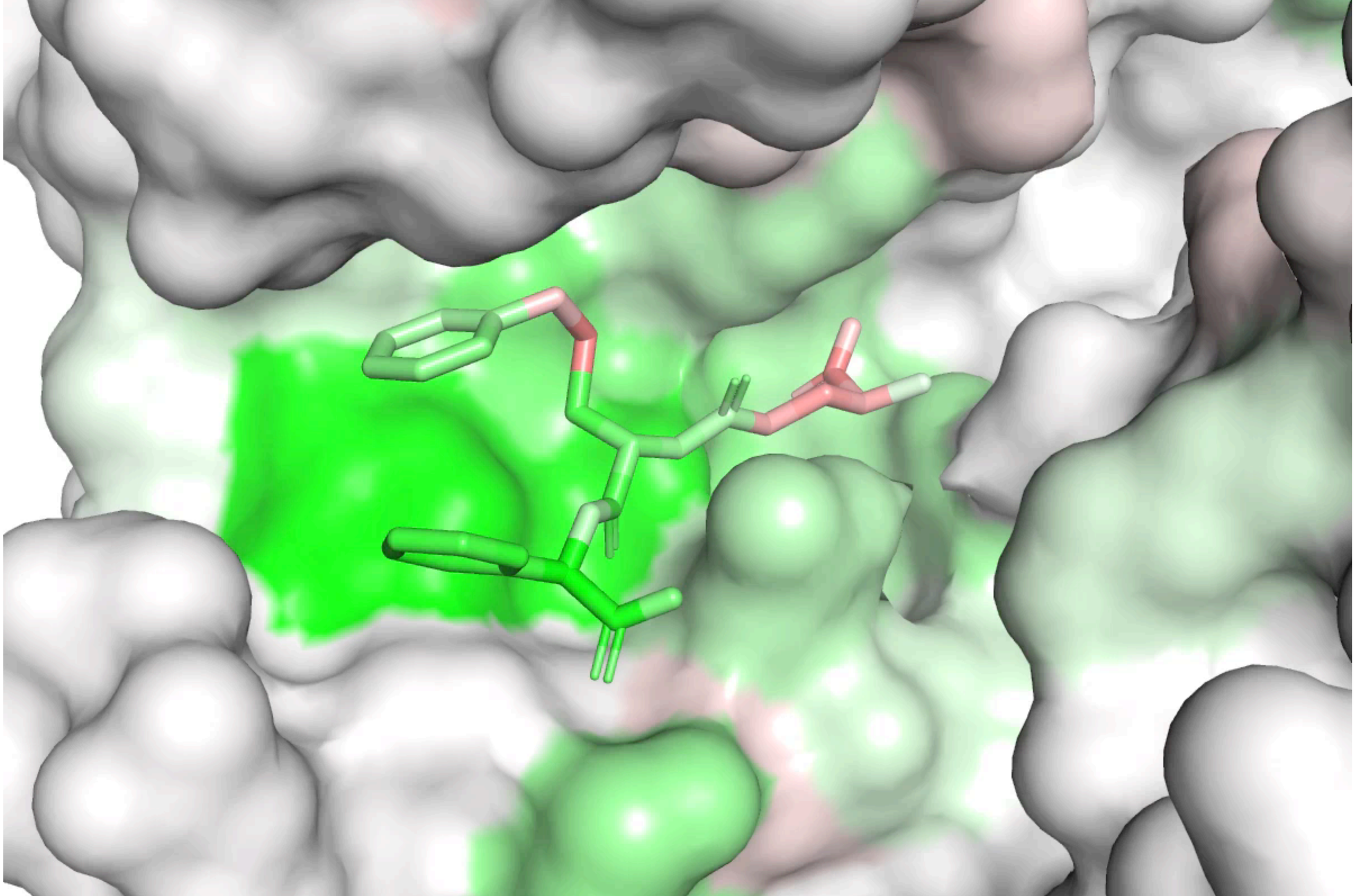
$$\frac{\partial L}{\partial A} = \sum_{i \in G_A} \frac{\partial L}{\partial G_i} \frac{\partial G_i}{\partial D} \frac{\partial D}{\partial A}$$

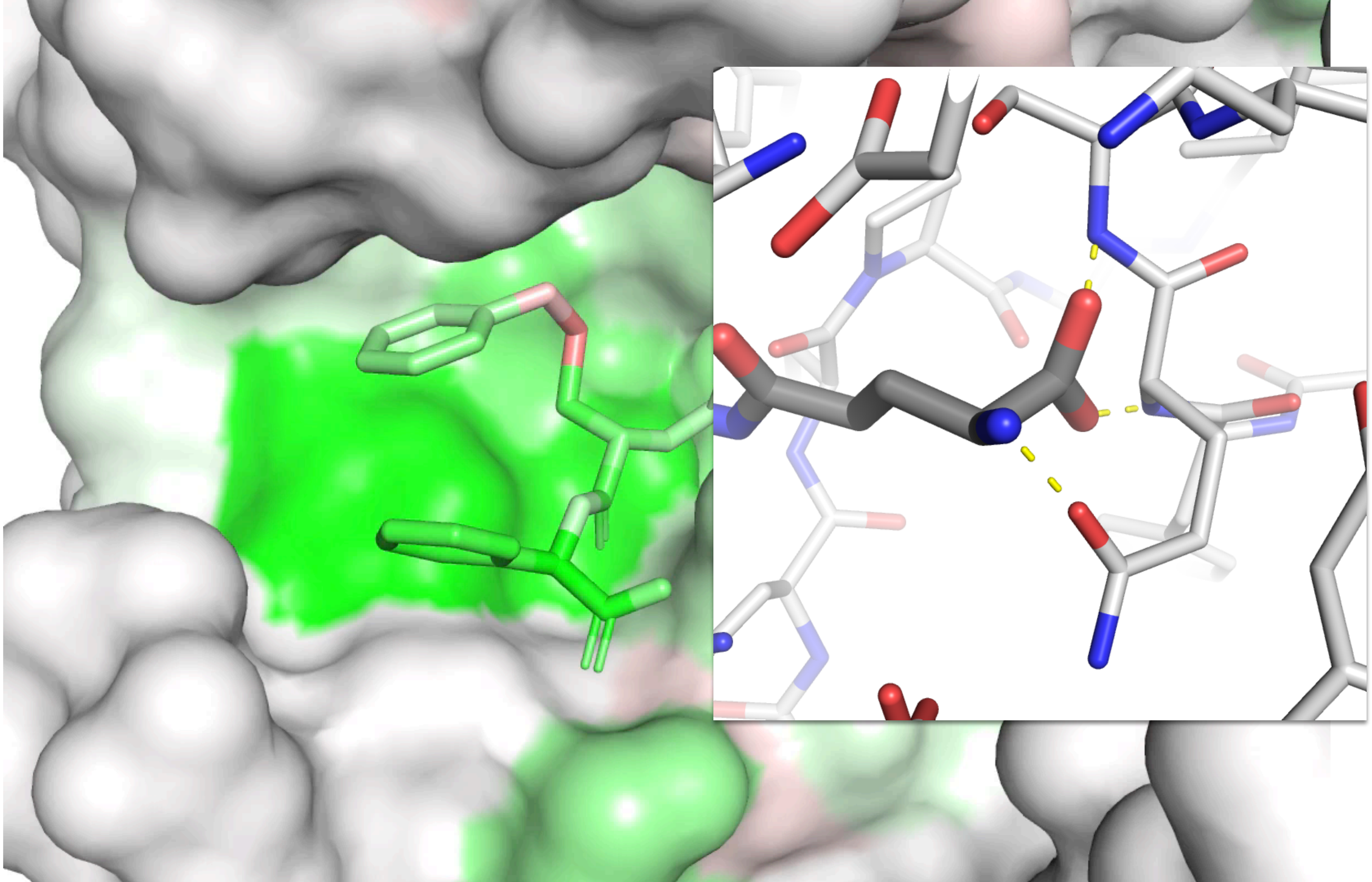
Beyond Scoring

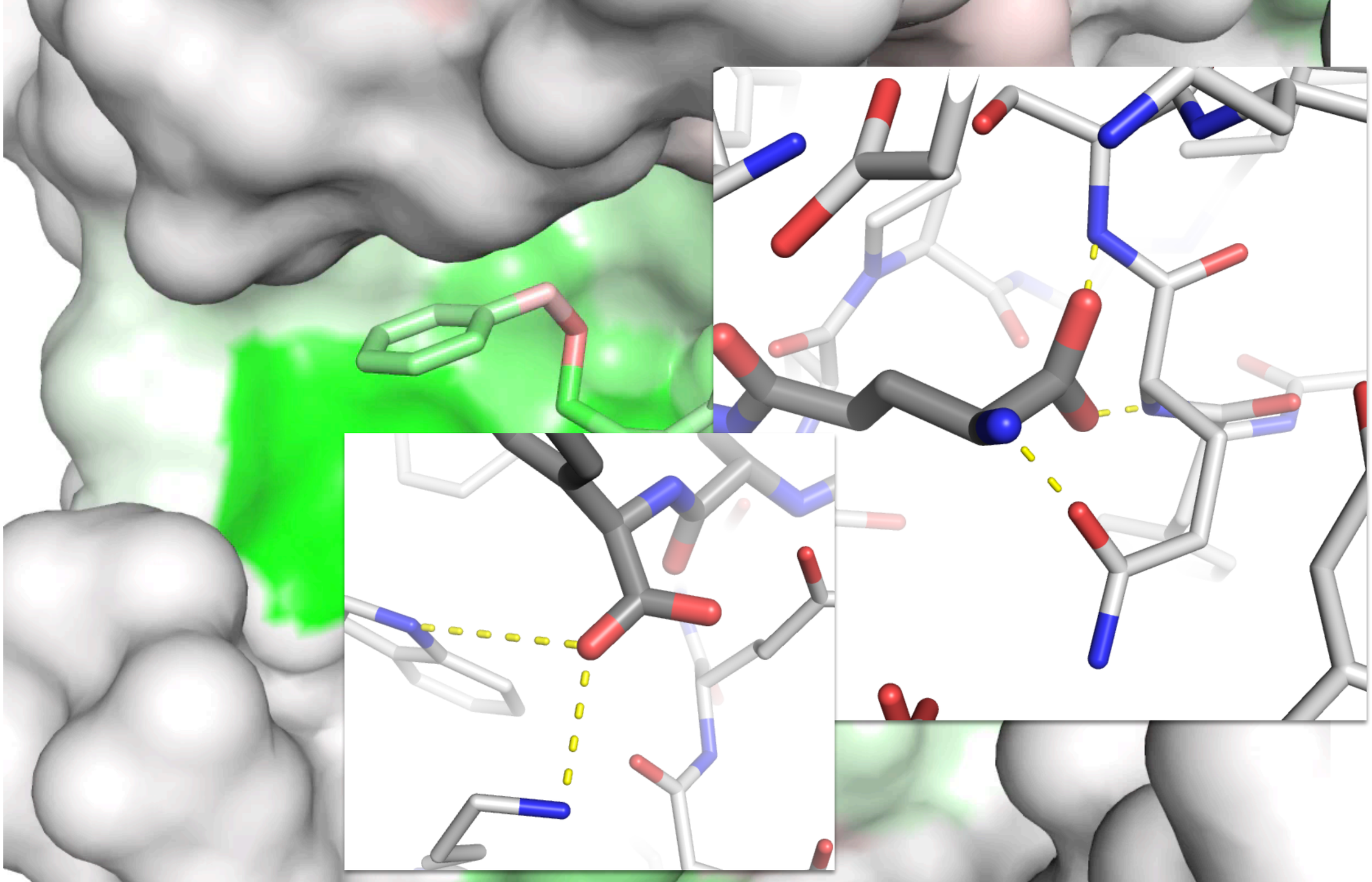


$$\frac{\partial L}{\partial A} = \sum_{i \in G_A} \frac{\partial L}{\partial G_i} \frac{\partial G_i}{\partial D} \frac{\partial D}{\partial A}$$

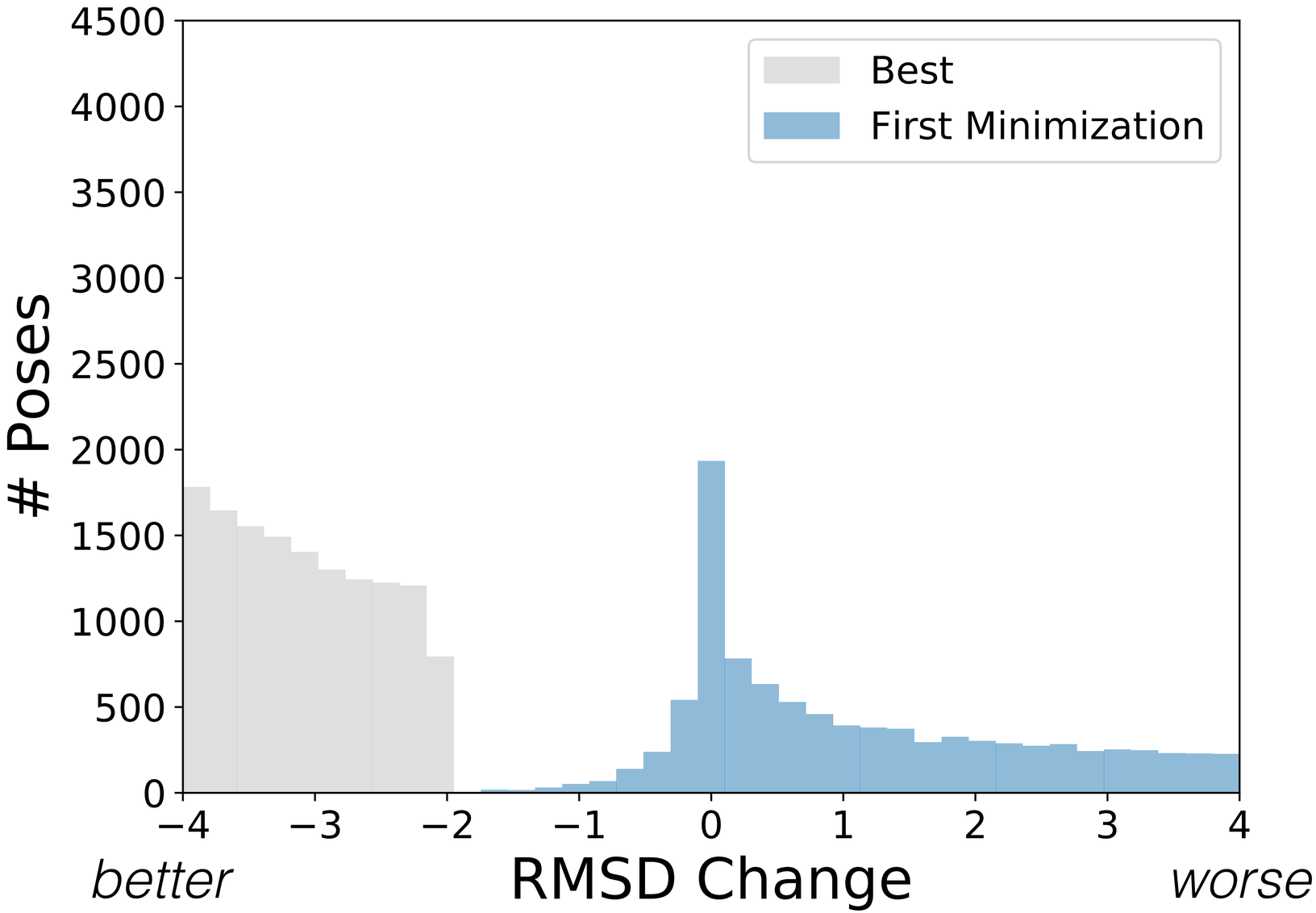




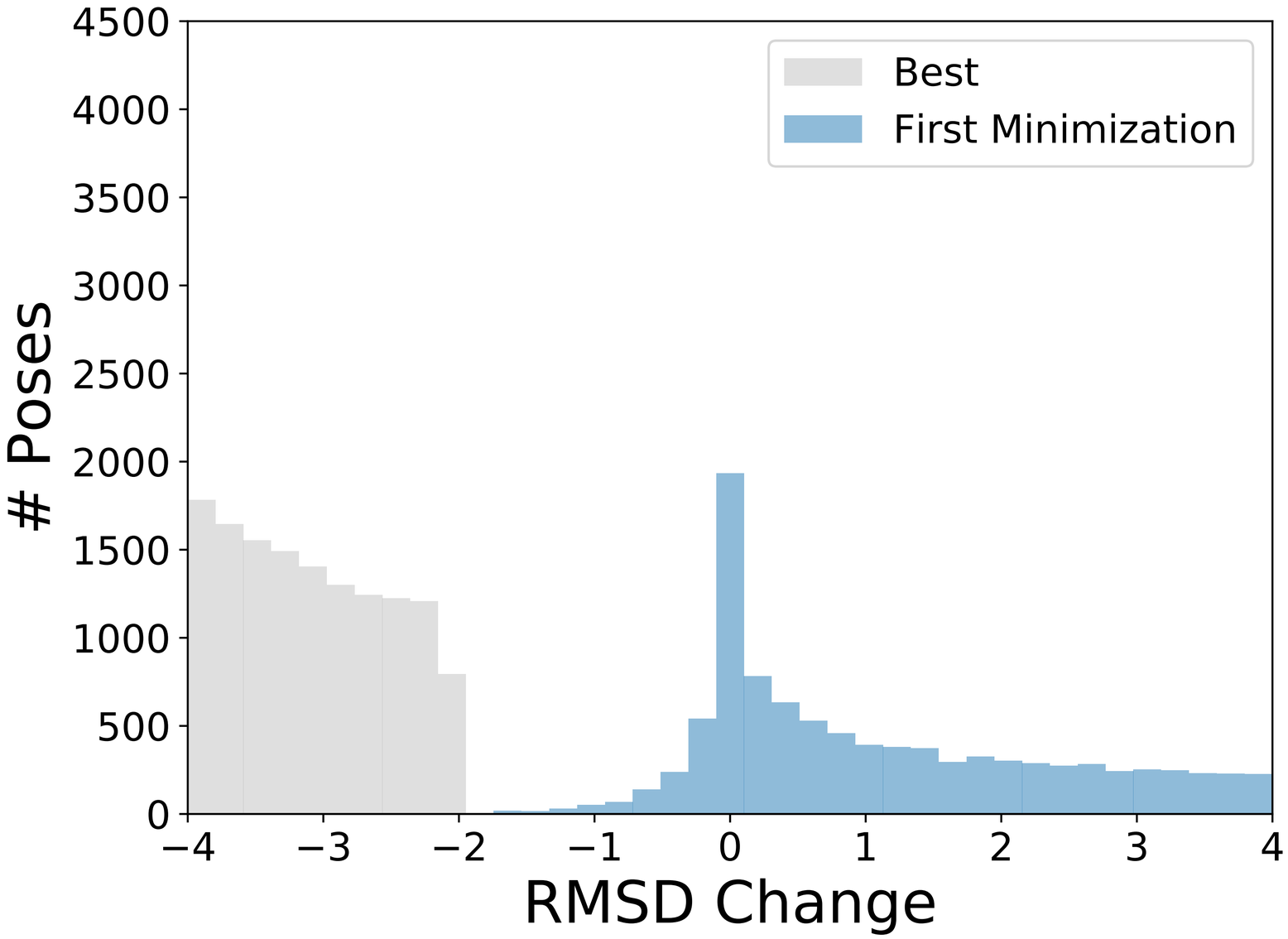




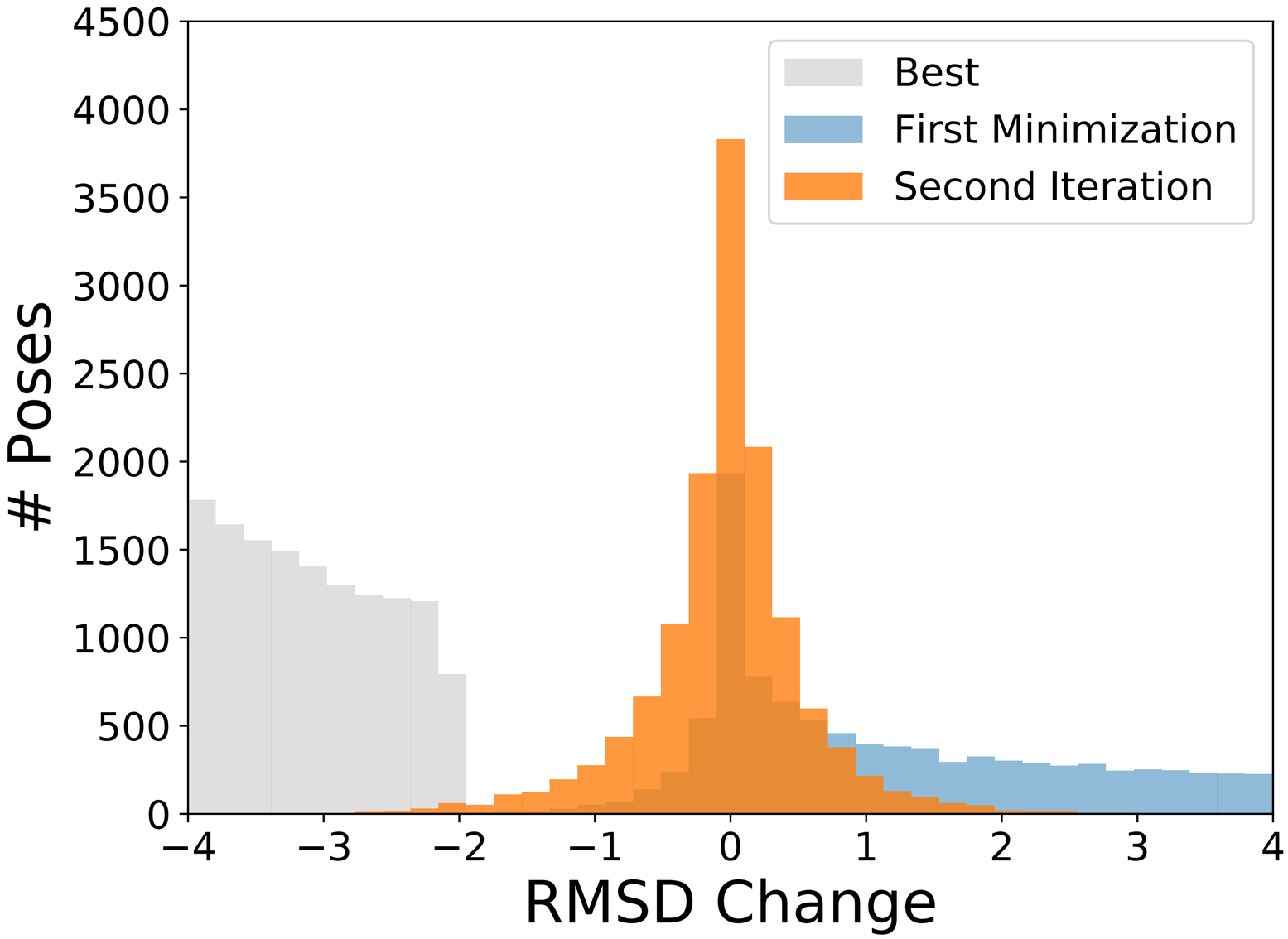
Minimizing Low RMSD Poses



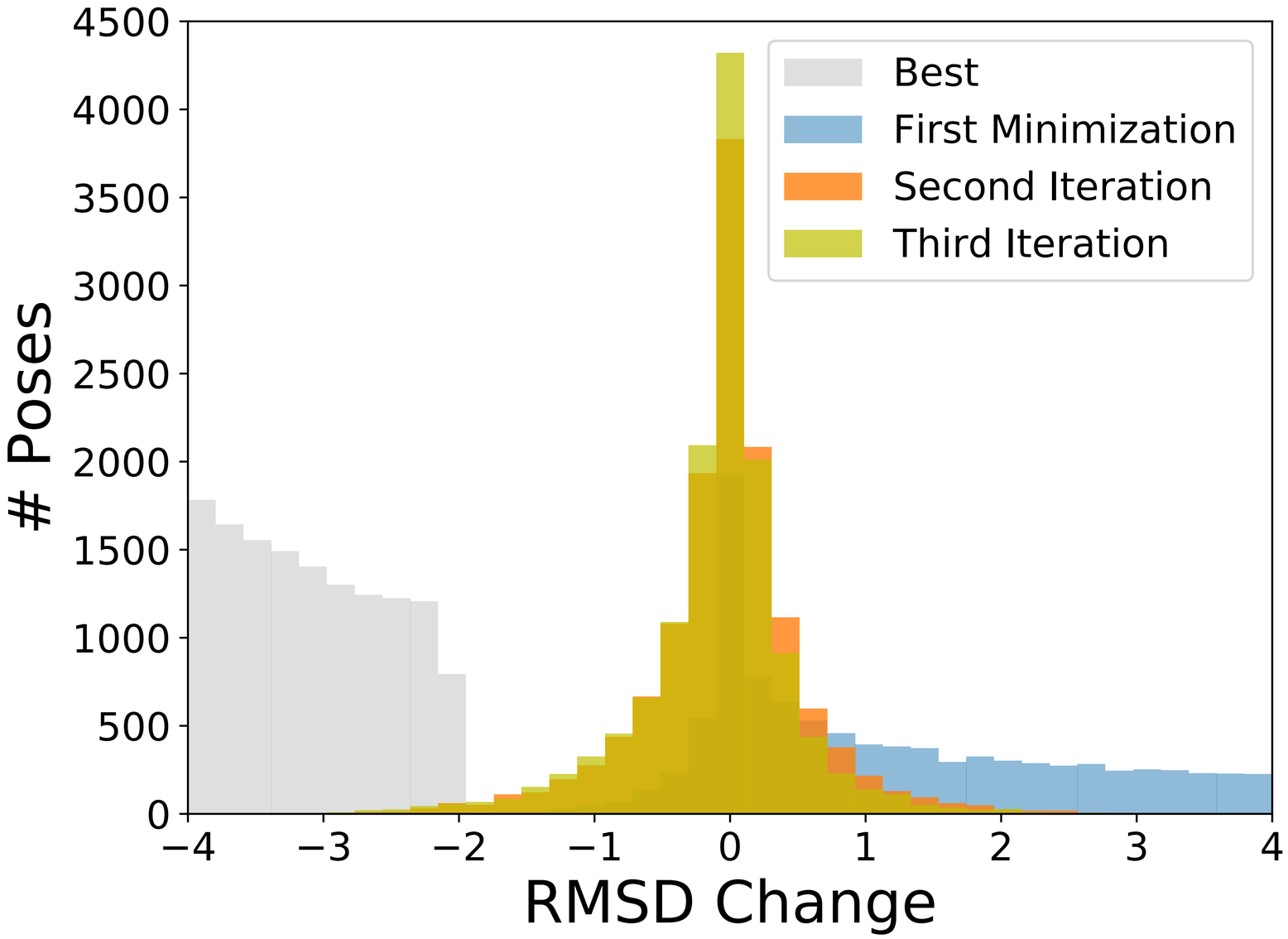
Iterative Refinement

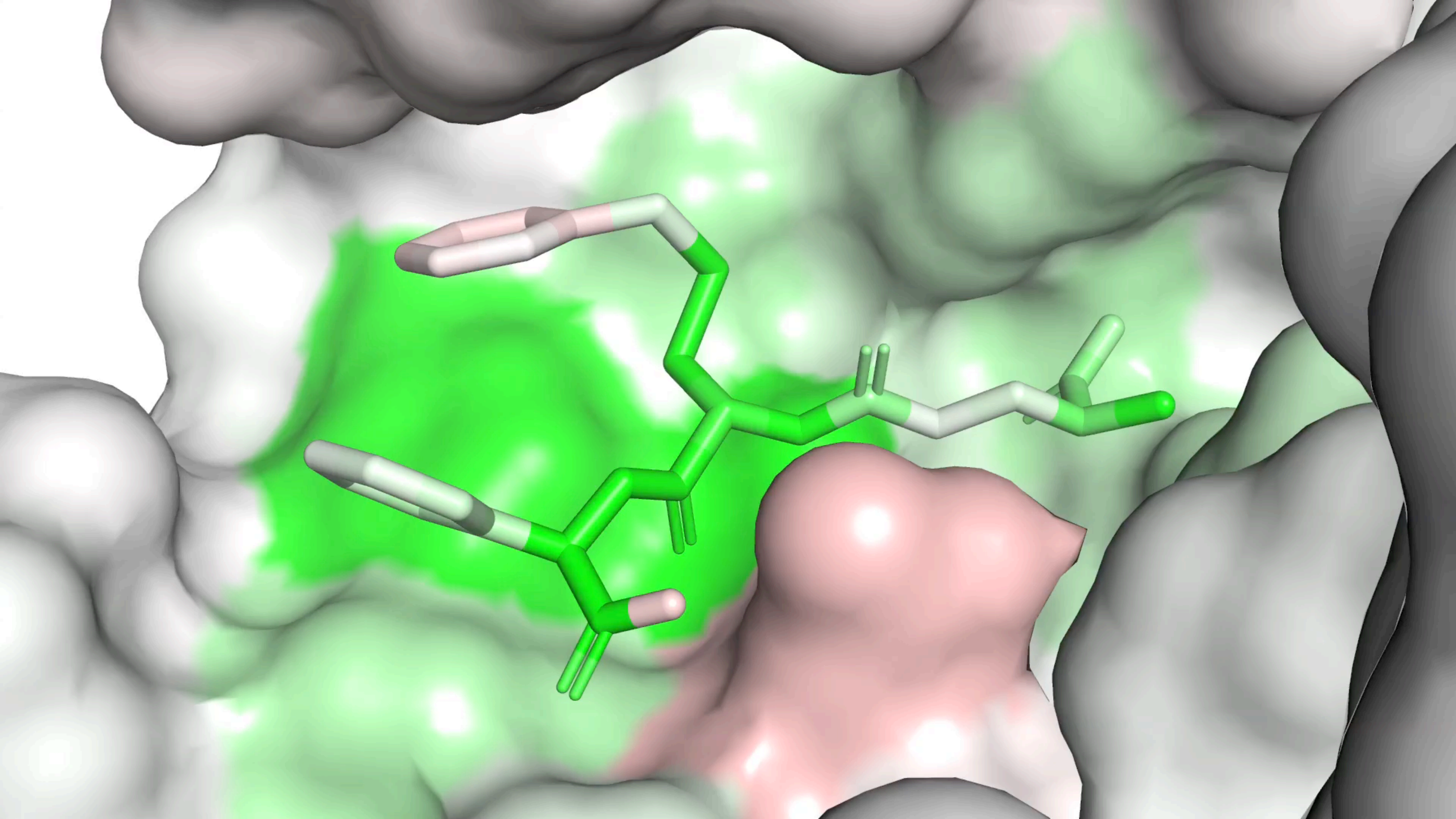


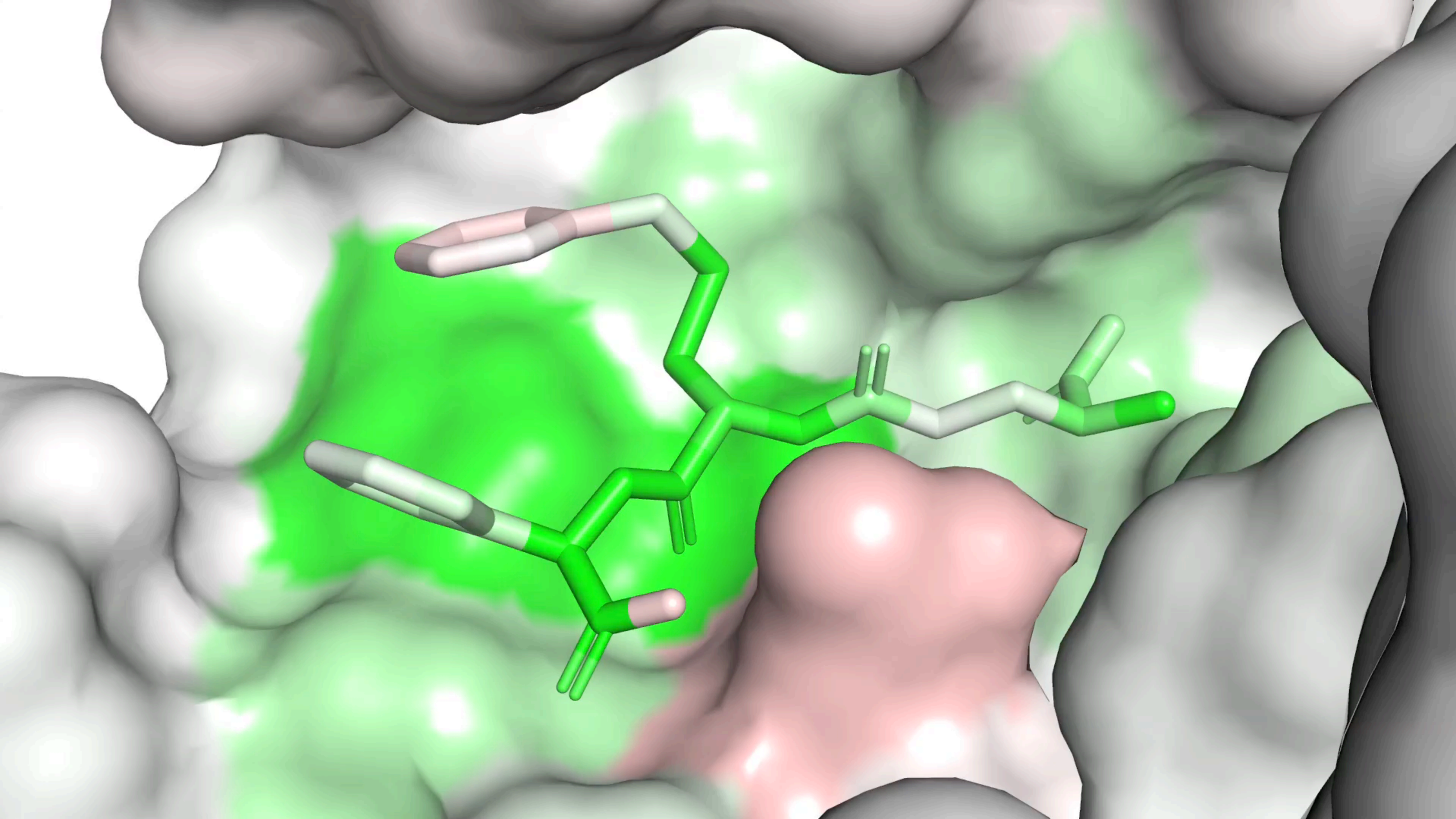
Iterative Refinement



Iterative Refinement

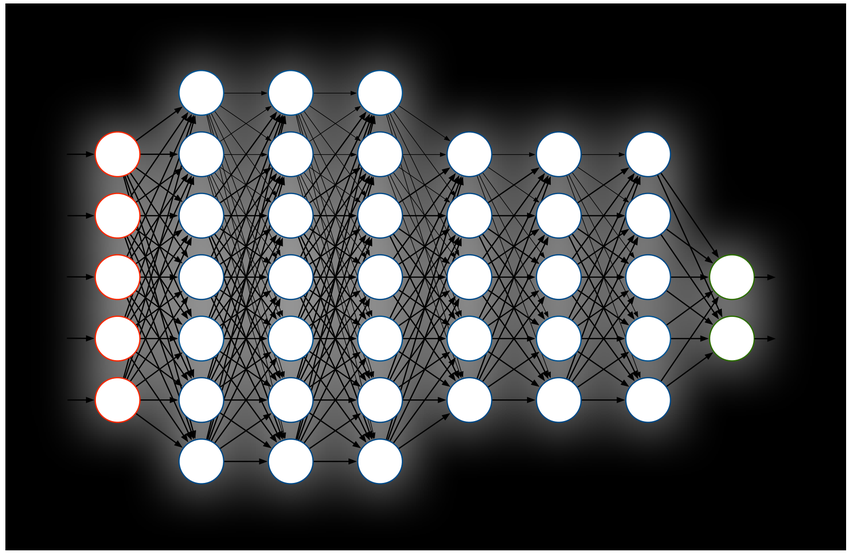




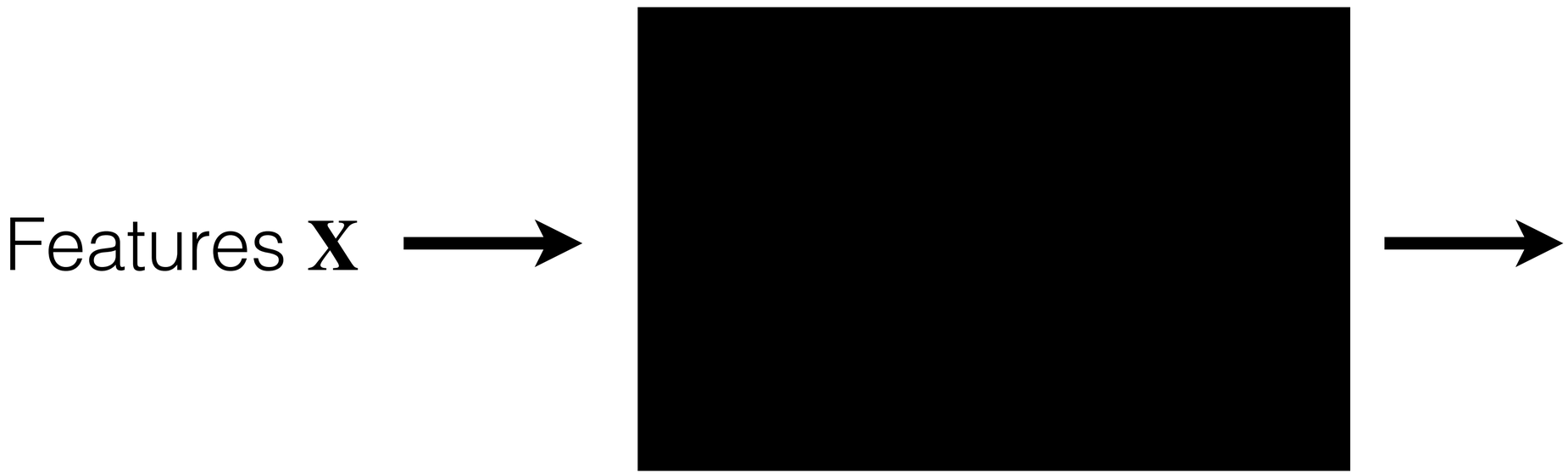


Generative Modeling

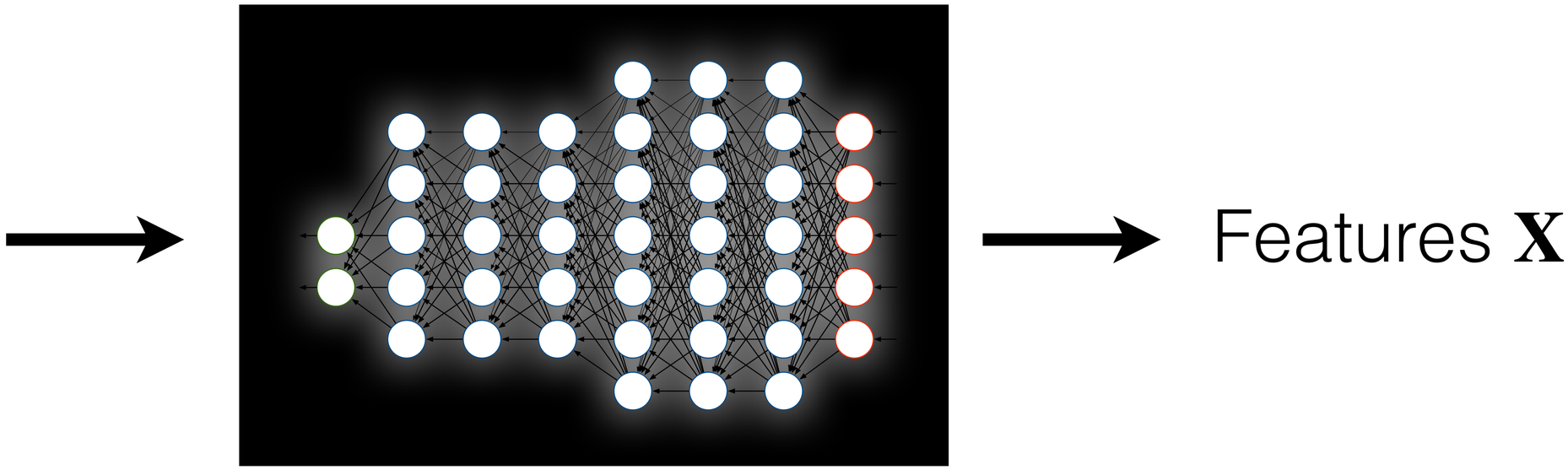
Discriminative Model

Features $\mathbf{x}$  Prediction $\mathbf{y}$

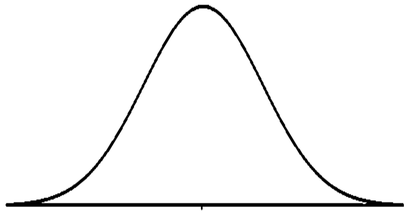
Generative Model



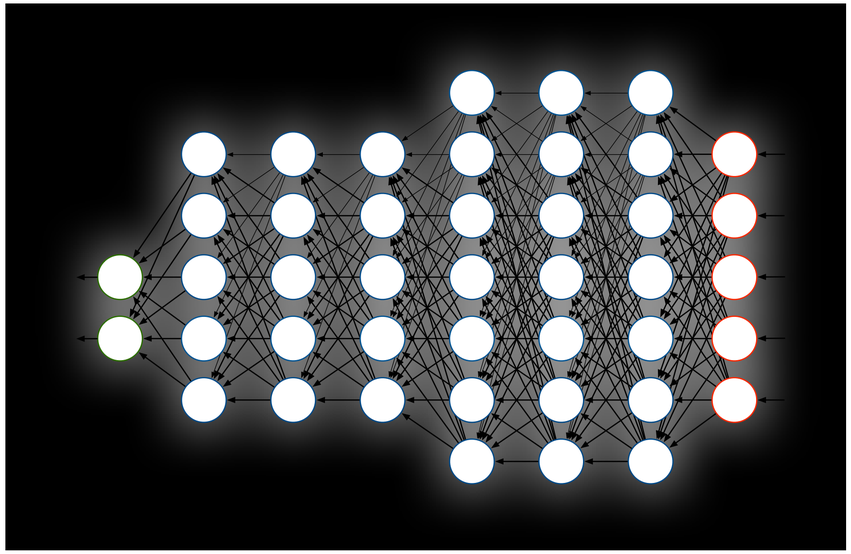
Generative Model



Generative Model

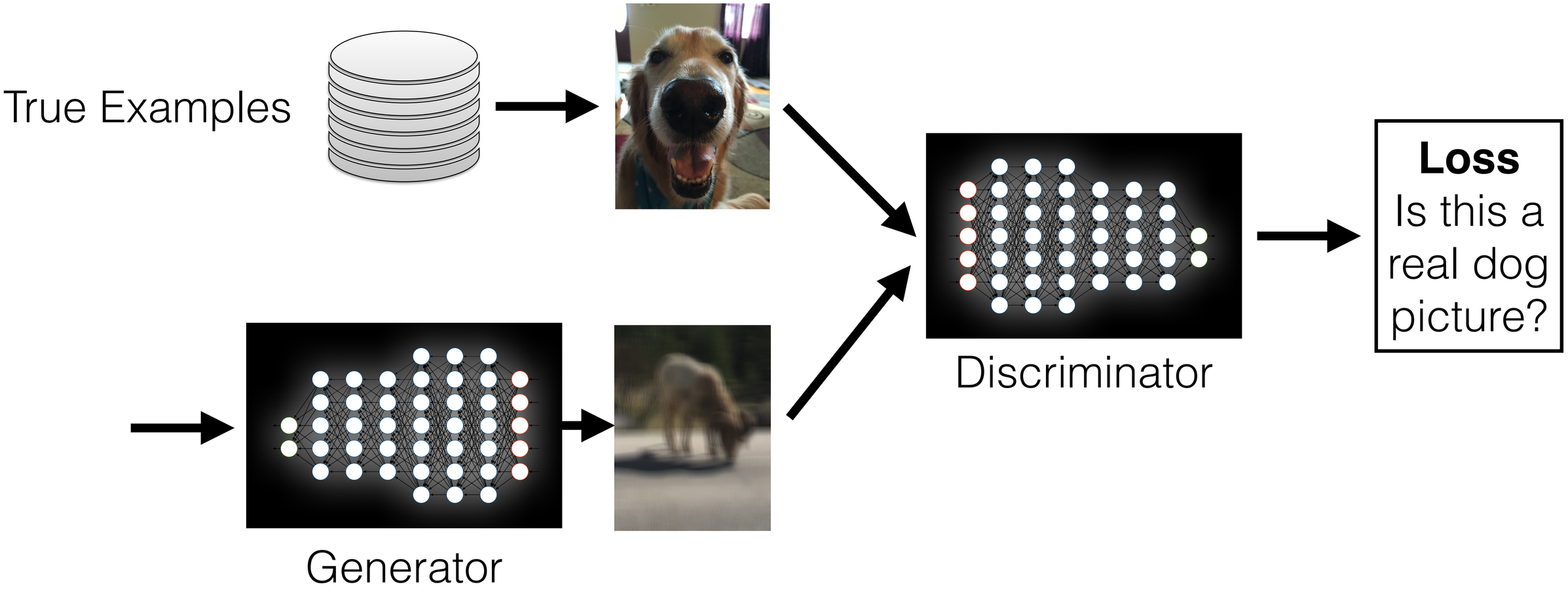


y?

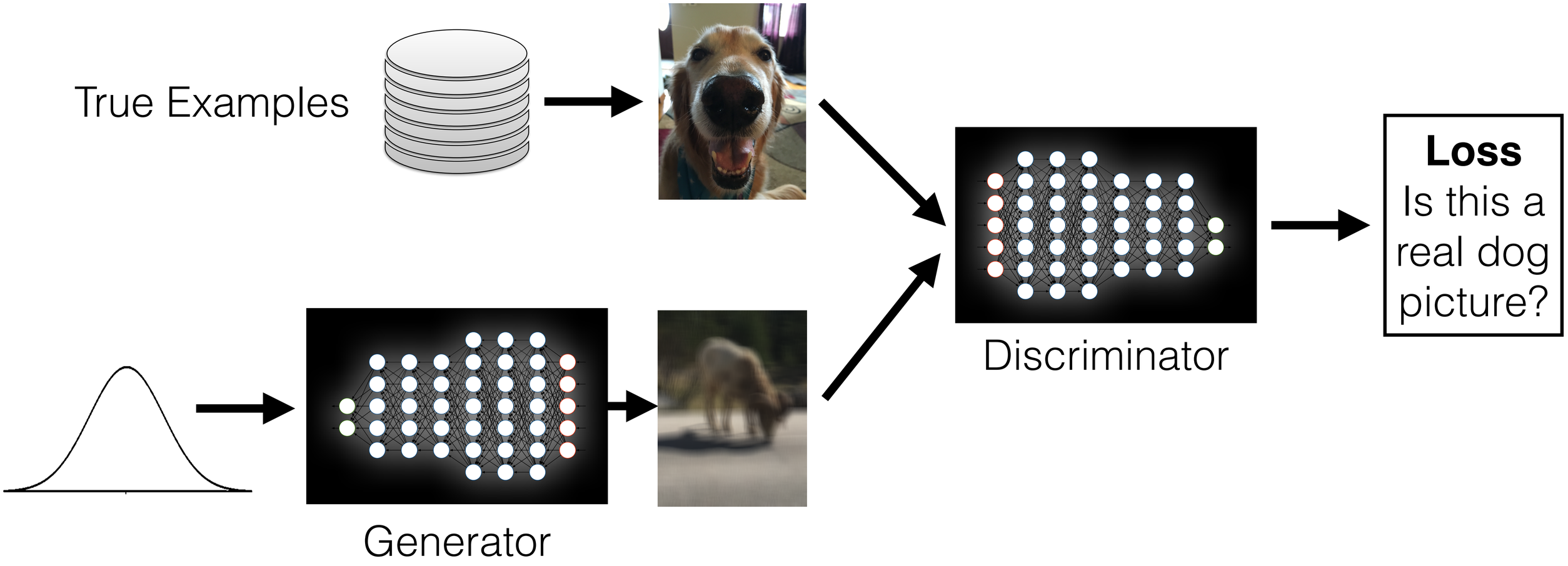


Features **X**

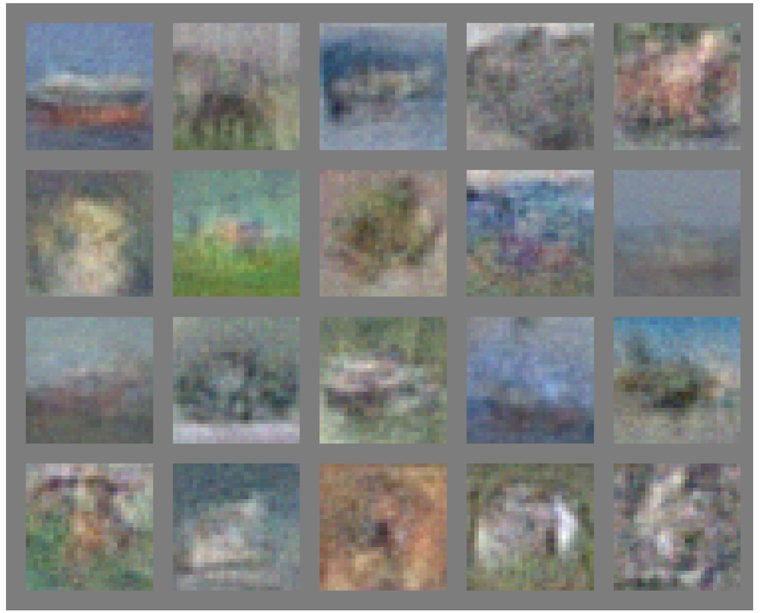
Generative Adversarial Networks



Generative Adversarial Networks



Generative Adversarial Networks



Generative Adversarial Networks

https://arxiv.org › stat ▾

by IJ Goodfellow - 2014 - Cited by 4339 - Related articles

Jun 10, 2014 - Submission history. From: Ian **Goodfellow** [view email] [v1] Tue, 10 Jun **2014** 18:58:17 GMT (1257kb,D). Which authors of this paper are ...

Generative Adversarial Networks

http://torch.ch/blog/2015/11/13/gan.html

Generative Adversarial Networks

https://arxiv.org › stat ▾

by IJ Goodfellow - 2014 - Cited by 4339 - Related articles

Jun 10, 2014 - Submission history. From: Ian **Goodfellow** [view email] [v1] Tue, 10 Jun **2014** 18:58:17 GMT (1257kb,D). Which authors of this paper are ...

Progressive Growing of GANs for Improved Quality, Stability, and Variation

Tero Karras
NVIDIA

Timo Aila
NVIDIA

Samuli Laine
NVIDIA

Jaakko Lehtinen
NVIDIA
Aalto University

https://youtu.be/G06dEcZ-QTg

Progressive Growing of GANs for Improved Quality, Stability, and Variation

Tero Karras
NVIDIA

Timo Aila
NVIDIA

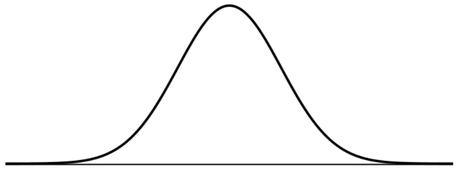
Samuli Laine
NVIDIA

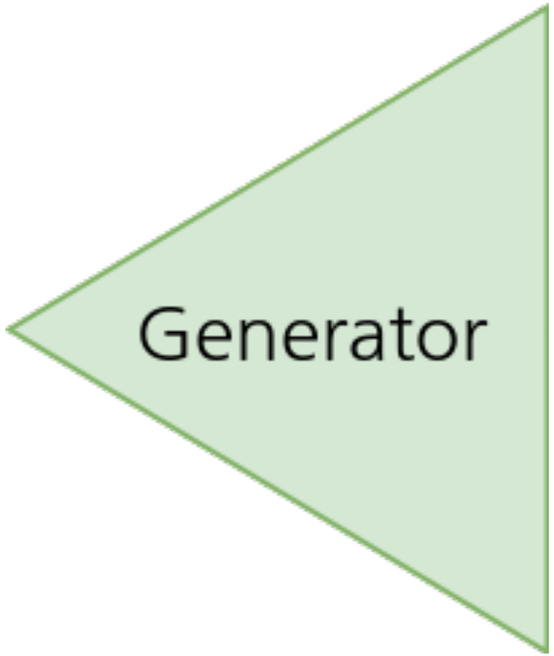
Jaakko Lehtinen
NVIDIA
Aalto University

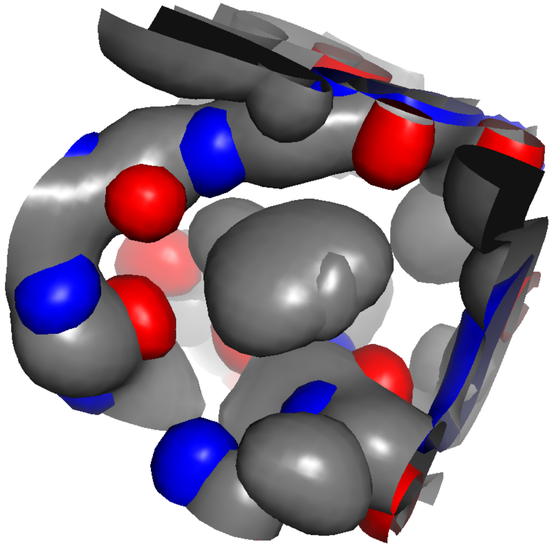
https://youtu.be/G06dEcZ-QTg

Generative Models

Generative models approximate a data distribution directly. They can map samples from one distribution (noise or input data) to realistic samples from an output distribution of interest.



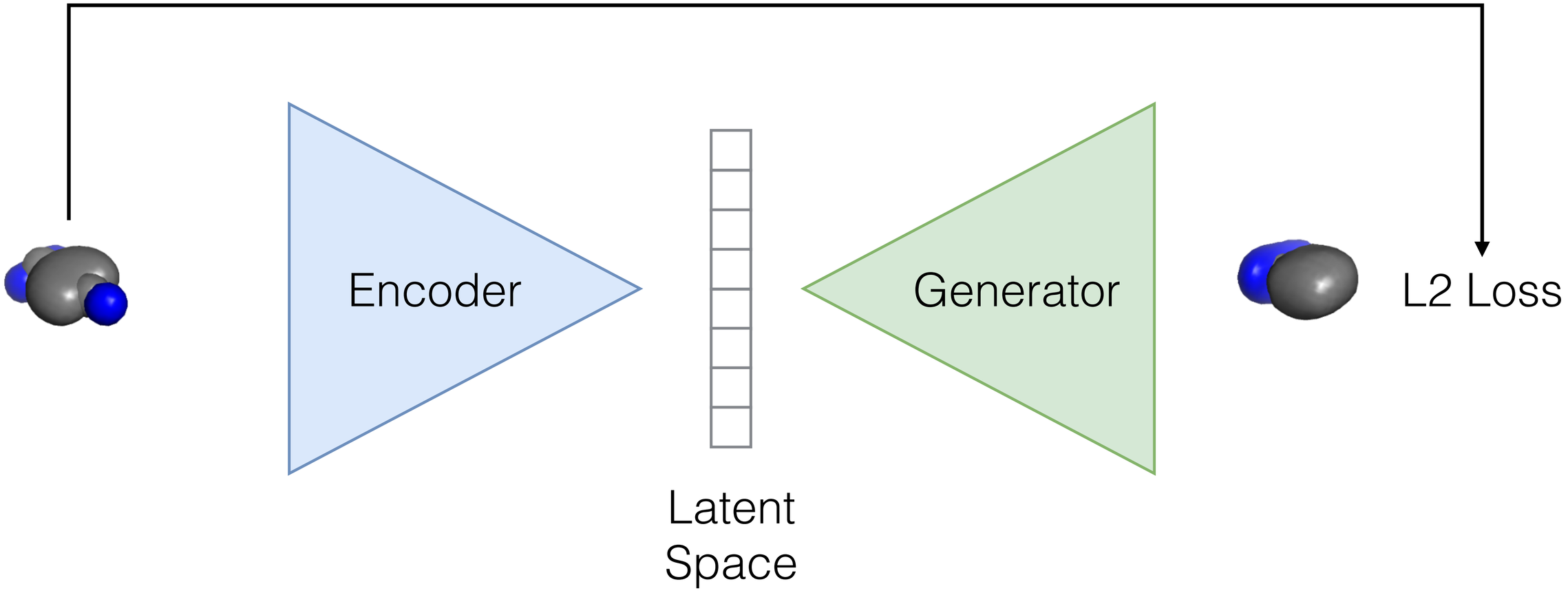




noise sample

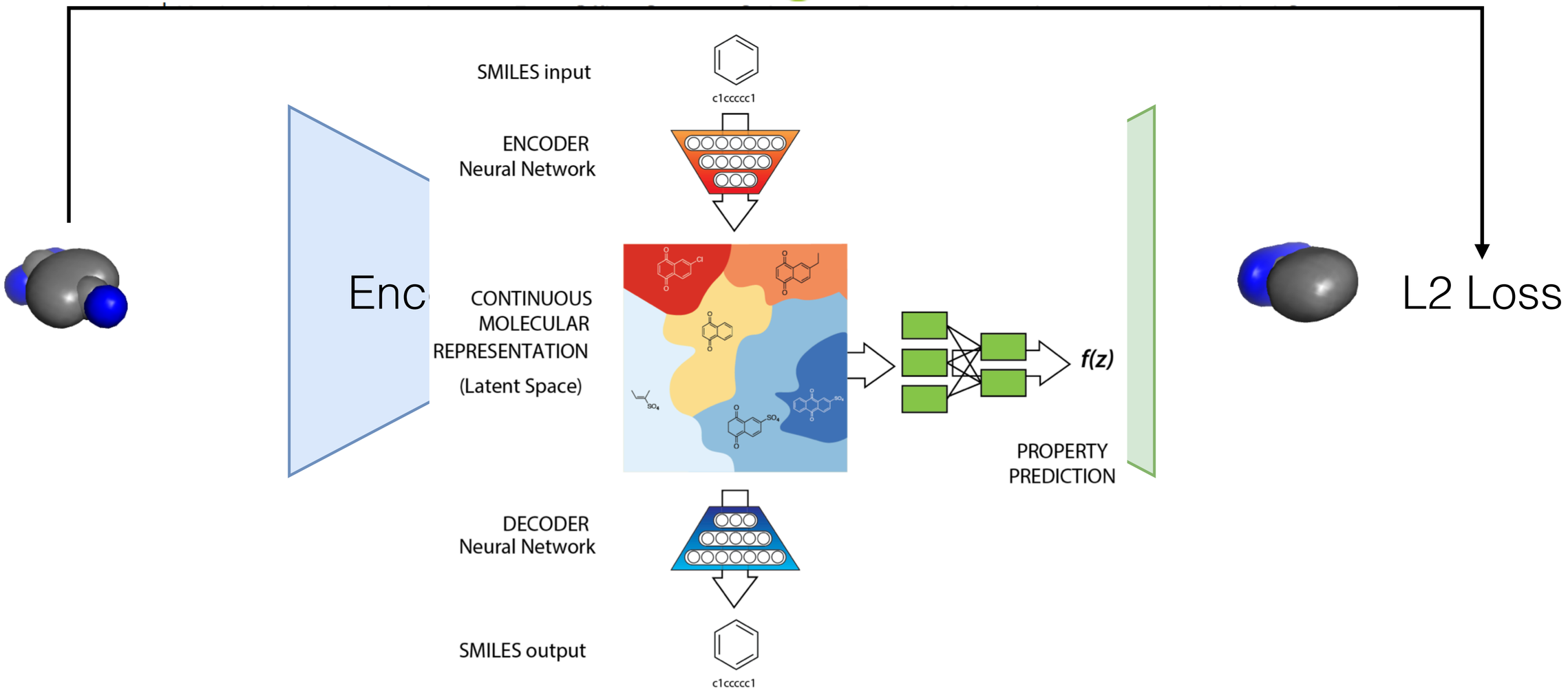
generated receptor & ligand grid

Autoencoding

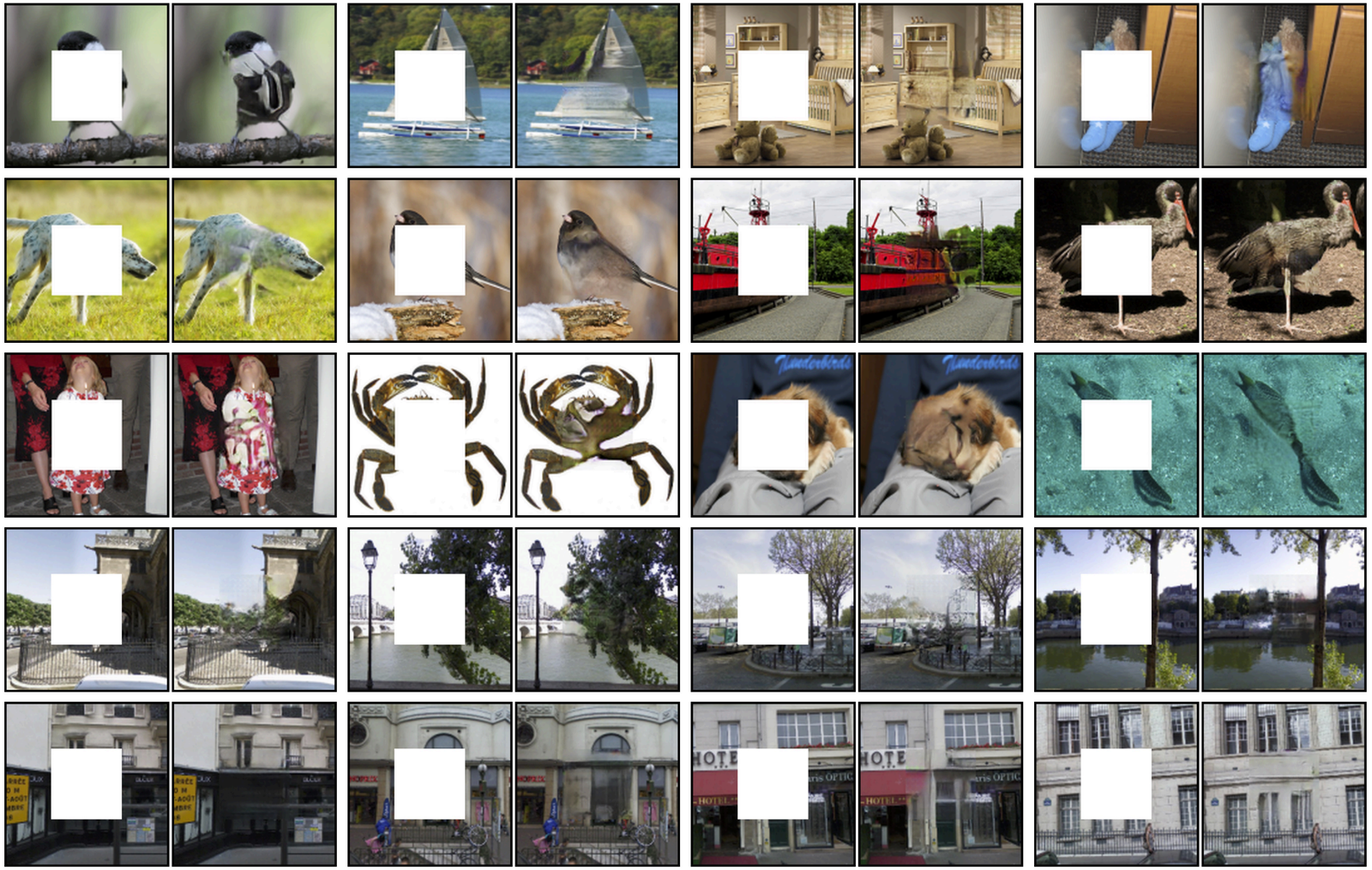


Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules

Rafael Gómez-Bombarelli[†#], Jennifer N. Wei[‡#], David Duvenaud[¶#], José Miguel Hernández-Lobato[§#], Benjamín Sánchez-Lengeling[‡], Dennis Sheberla[‡], Jorge Aguilera-Iparraguirre[†], Timothy D. Hirzel[†], Ryan P. Adams[∇⊥], and Alán Aspuru-Guzik[*‡⊥]

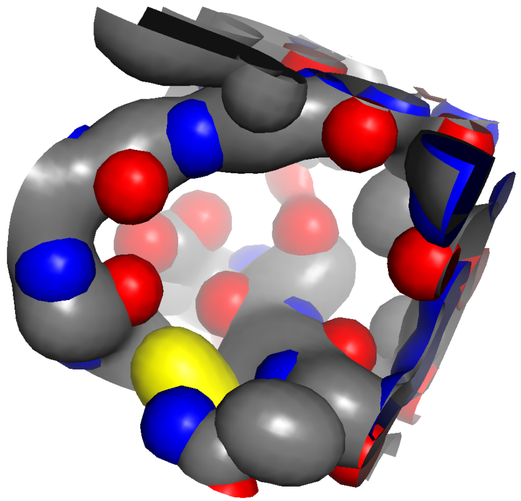


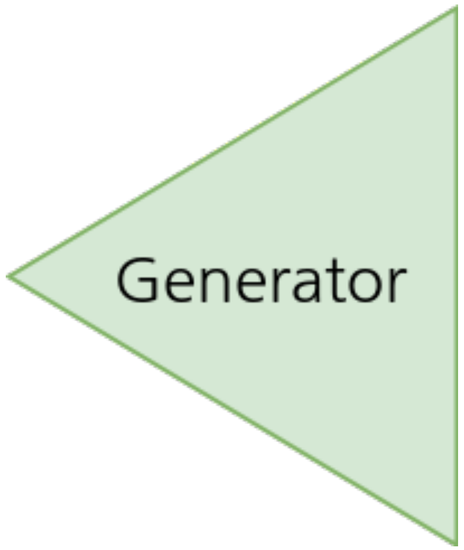
Context Encoding

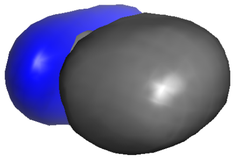


http://people.eecs.berkeley.edu/~pathak/context_encoder/

Context Encoding



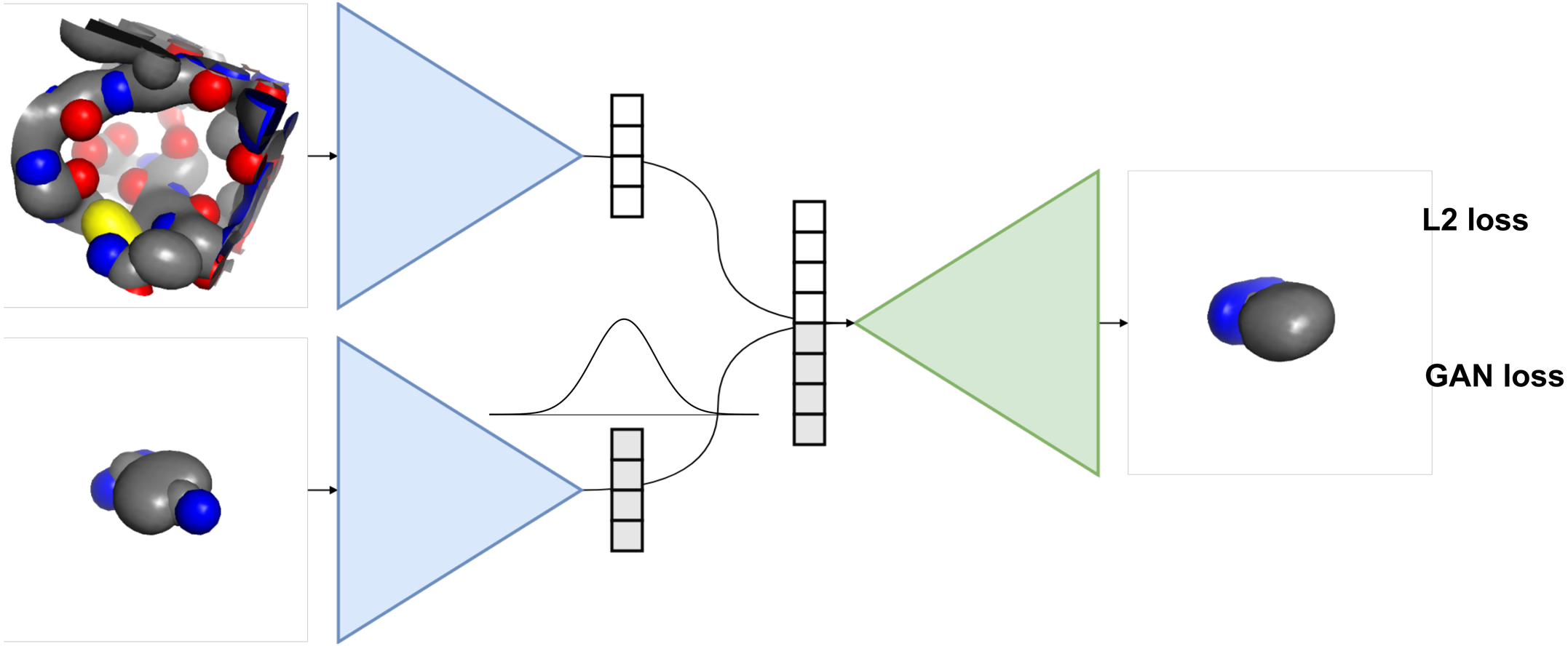




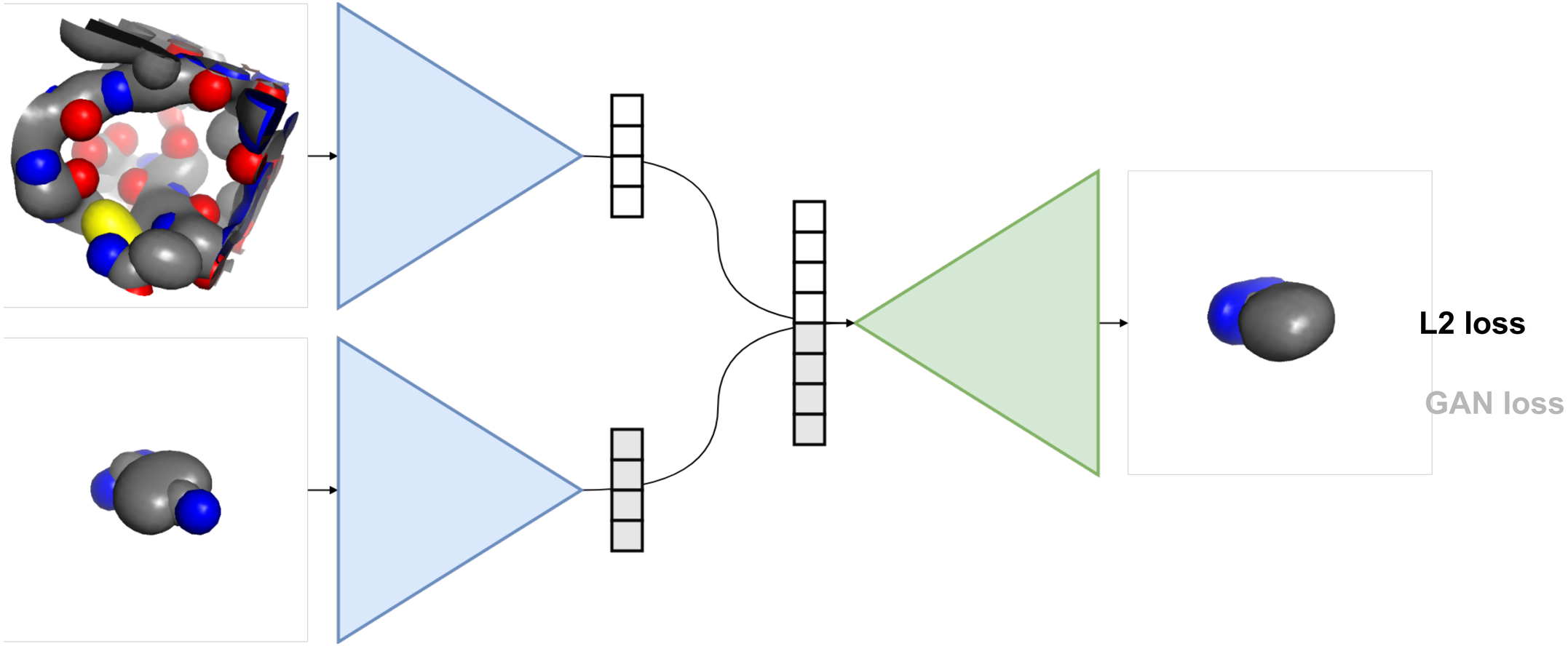
receptor grid

generated ligand grid

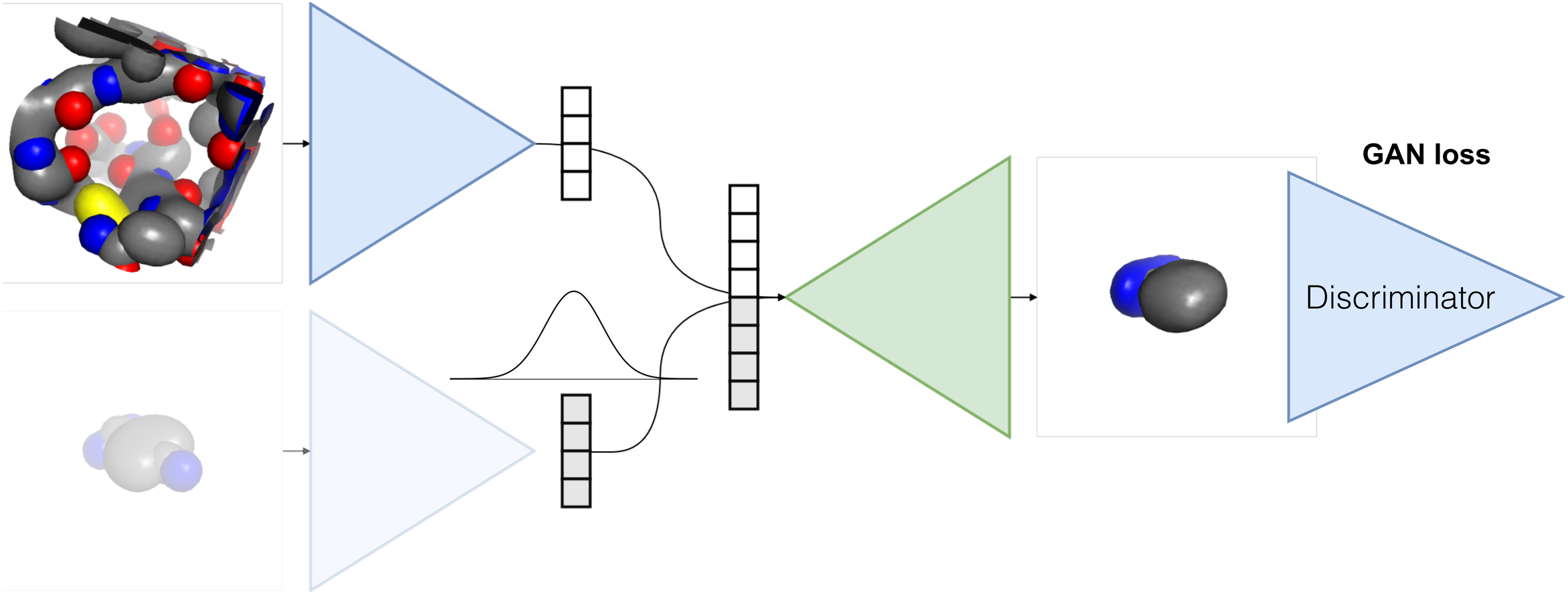
Receptor-Conditional Ligand-Variational Model



Receptor-Conditional Ligand-Variational Model

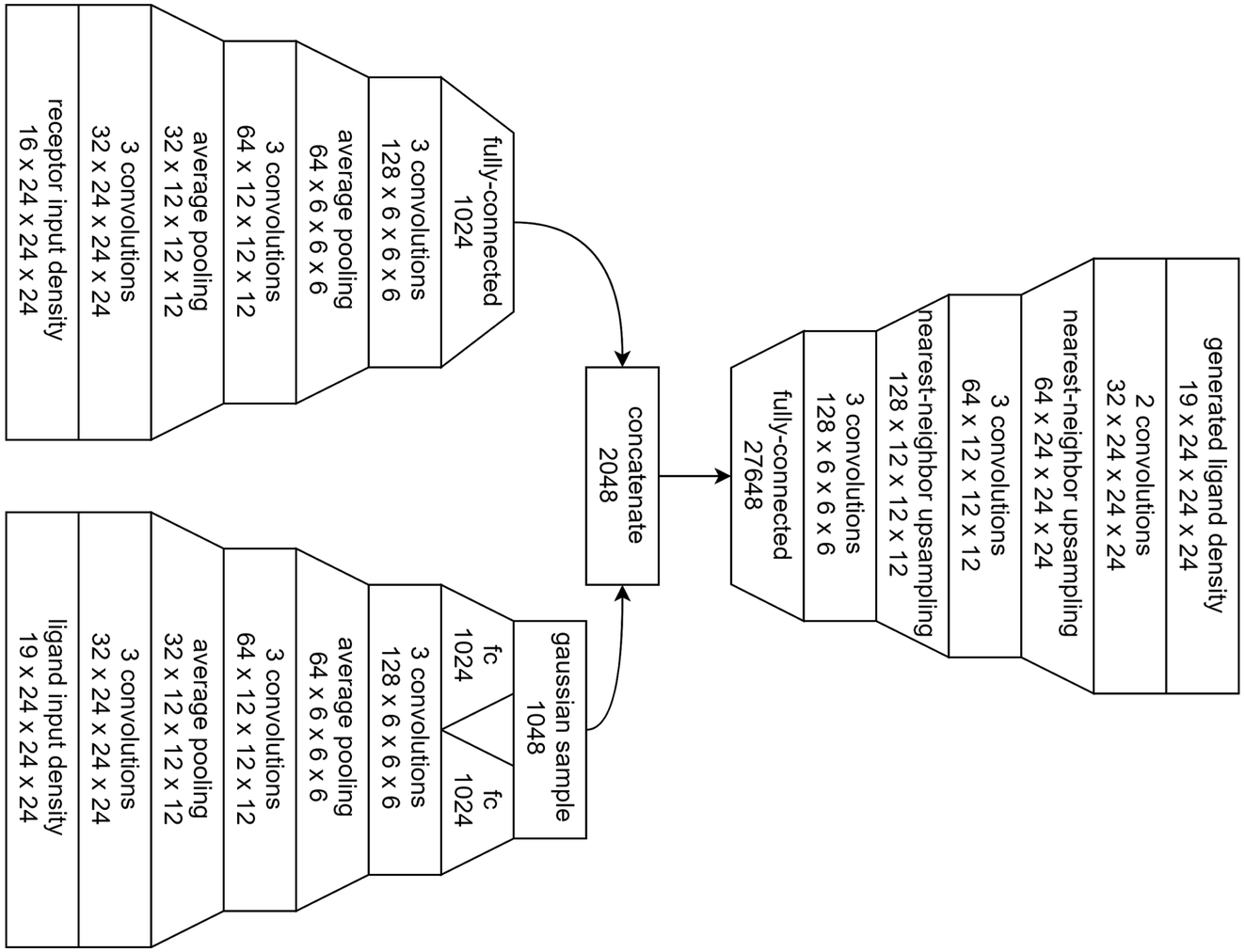


Receptor-Conditional Ligand-Variational Model

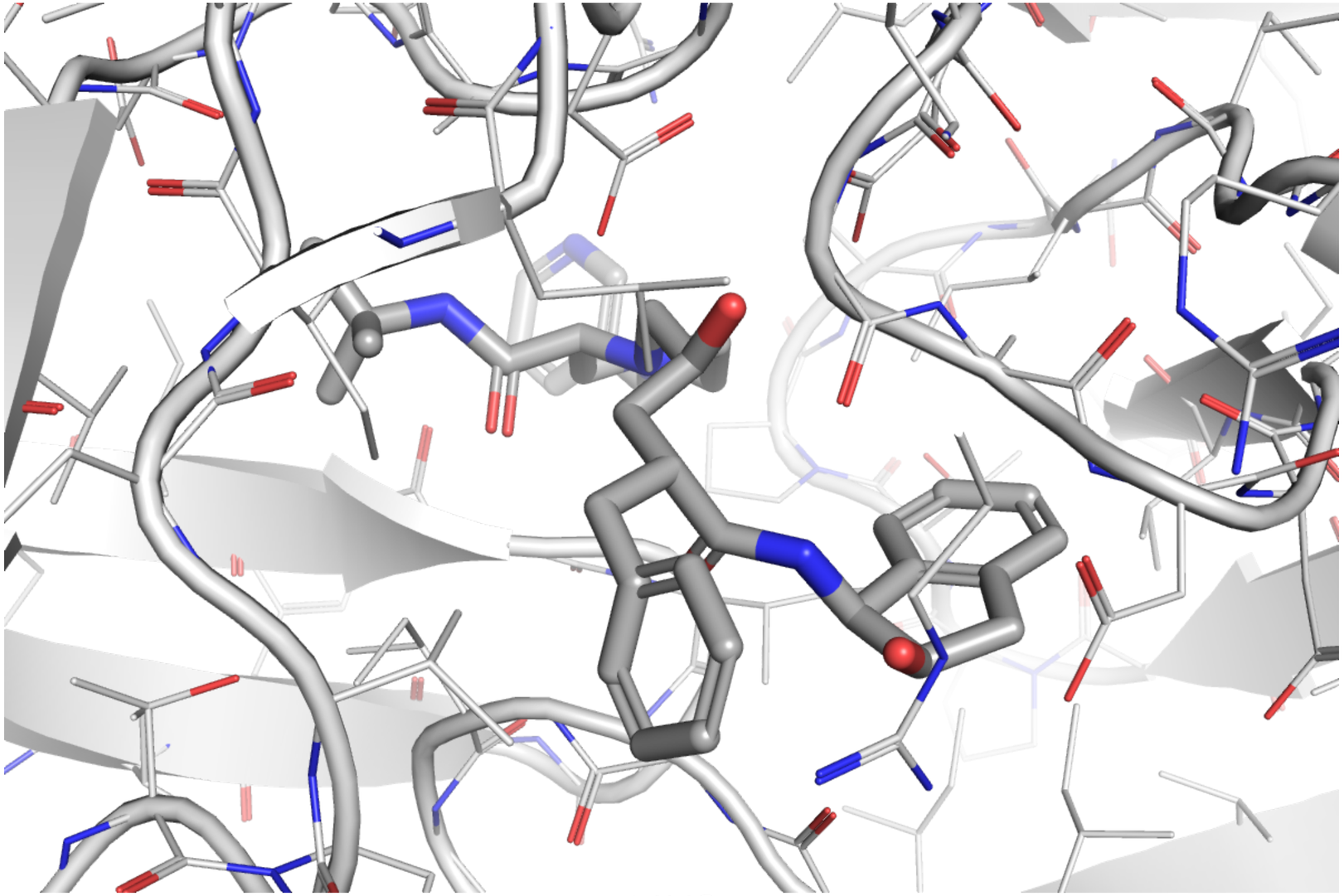


Model Architecture

n_levels = 3
conv_per_level = 3
n_filters = 32
width_factor = 2
n_latent = 1024

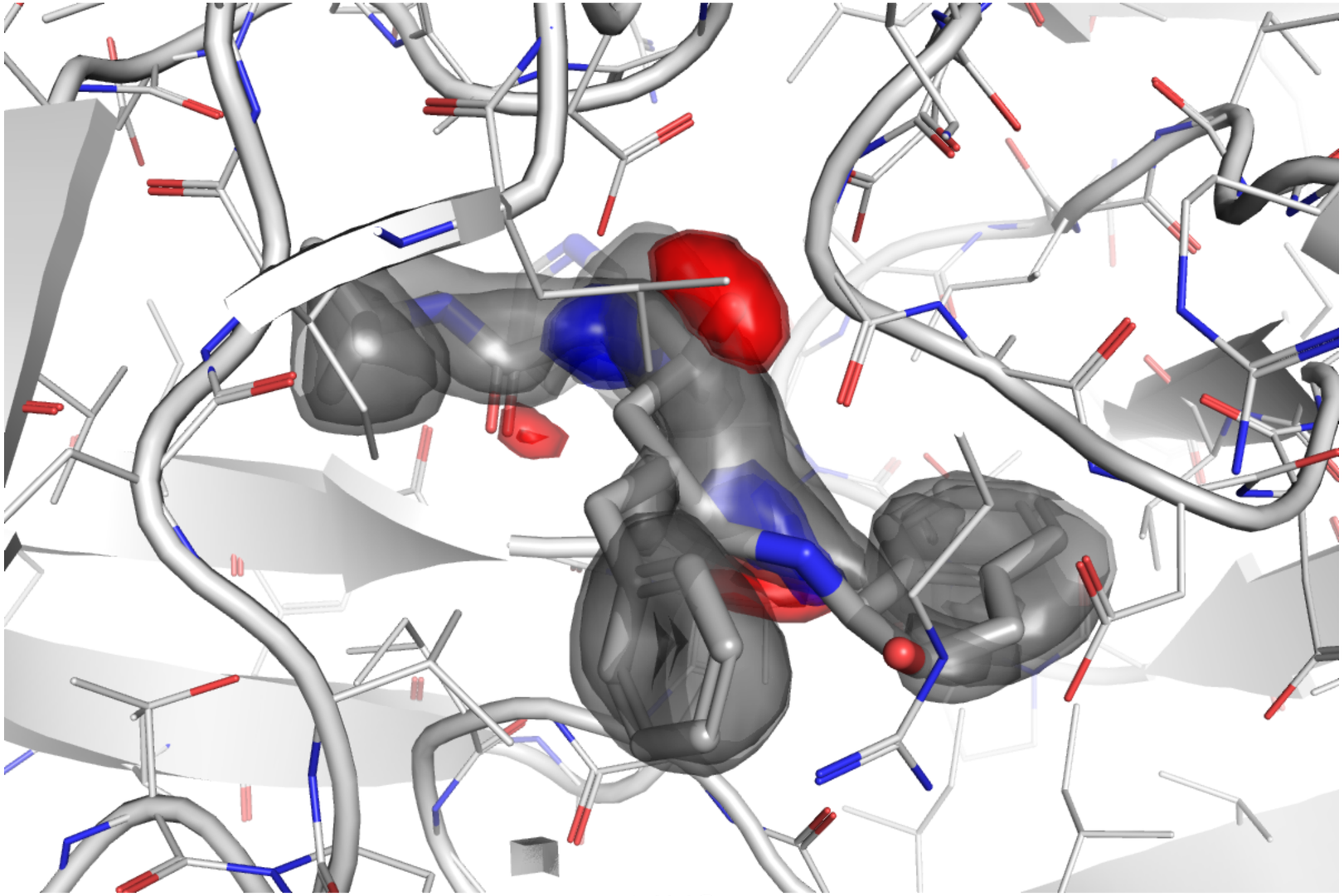


Autoencoding Examples



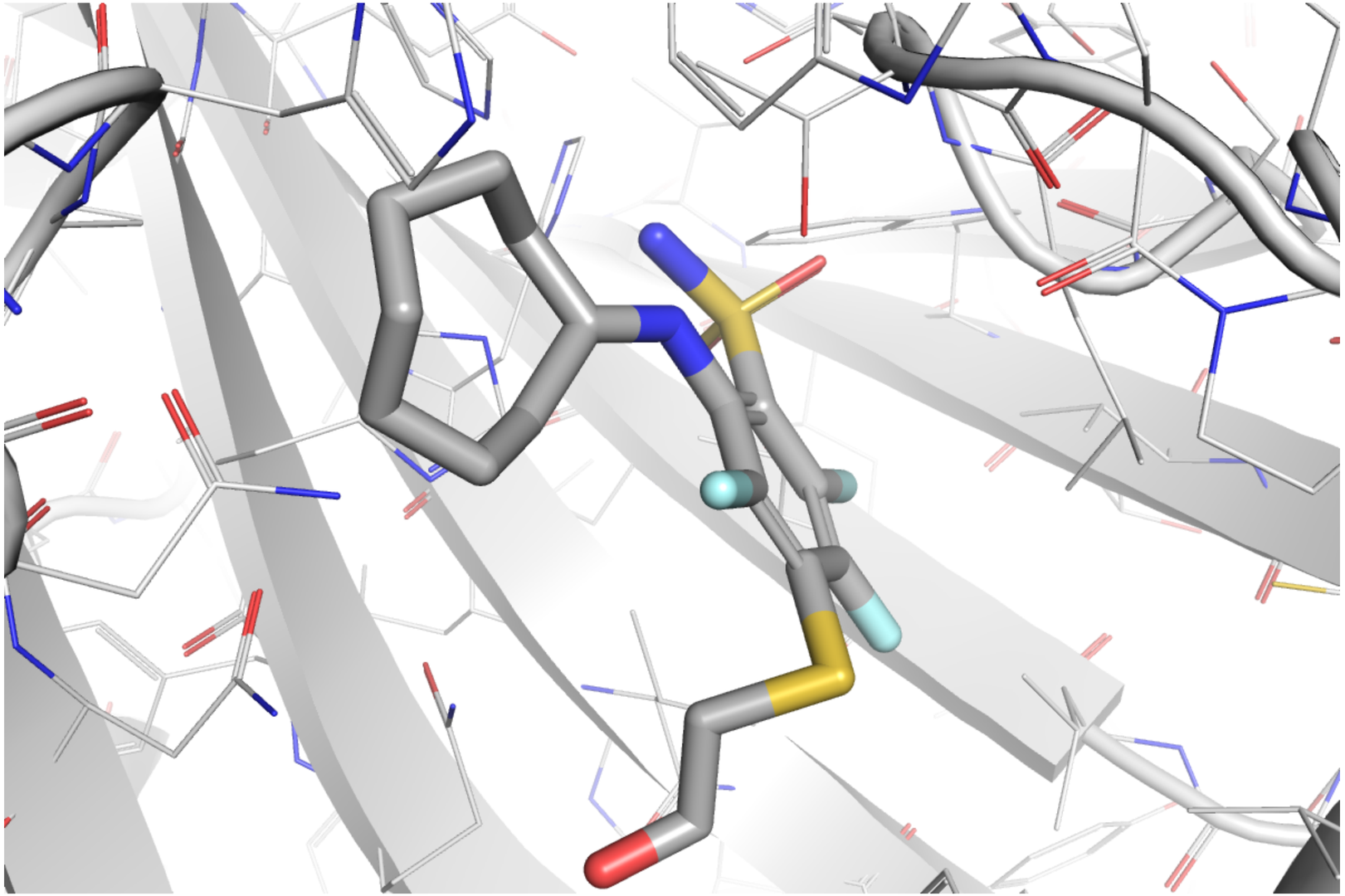
2AVO

Autoencoding Examples



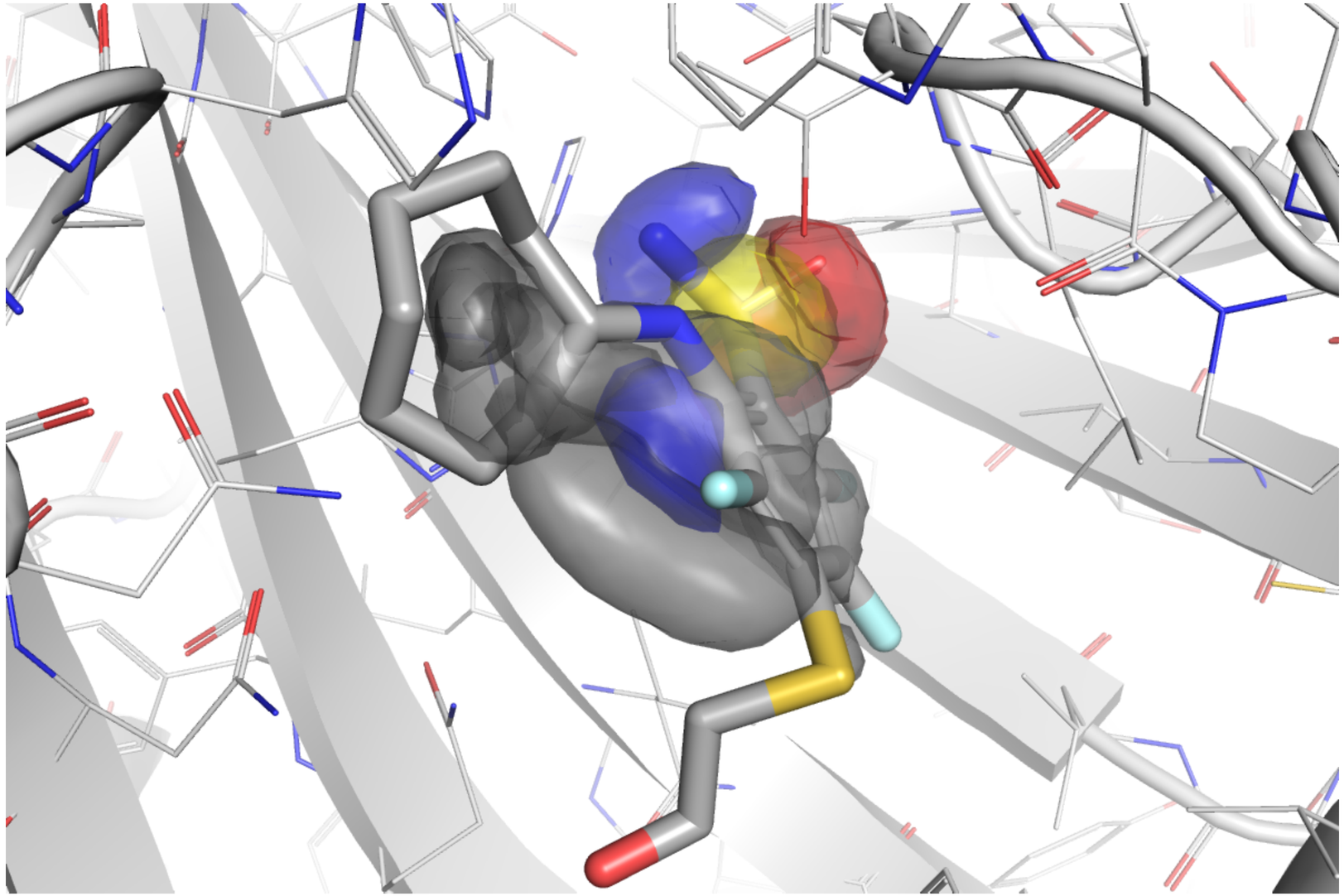
2AVO

Autoencoding Examples



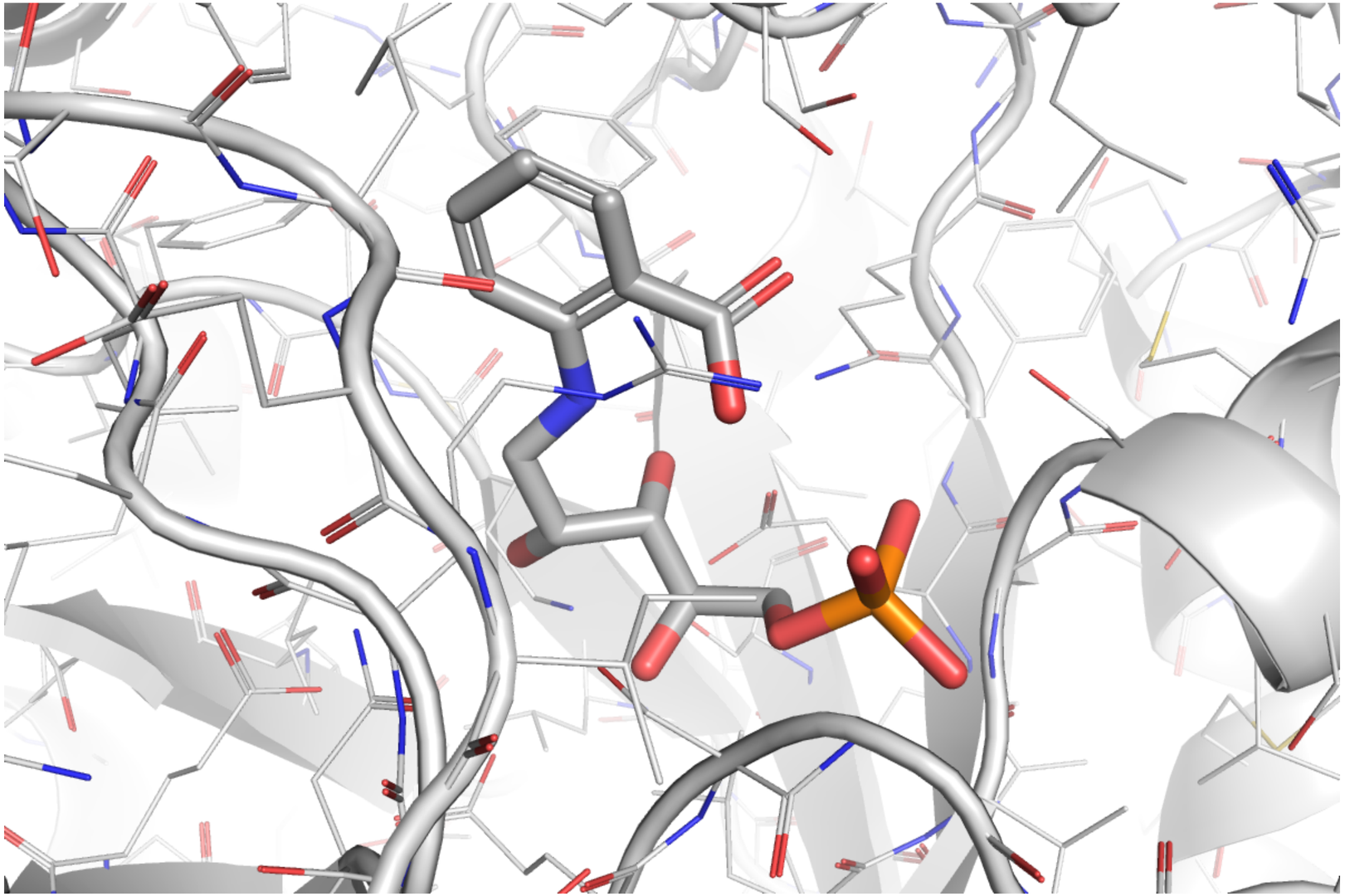
4PYX

Autoencoding Examples



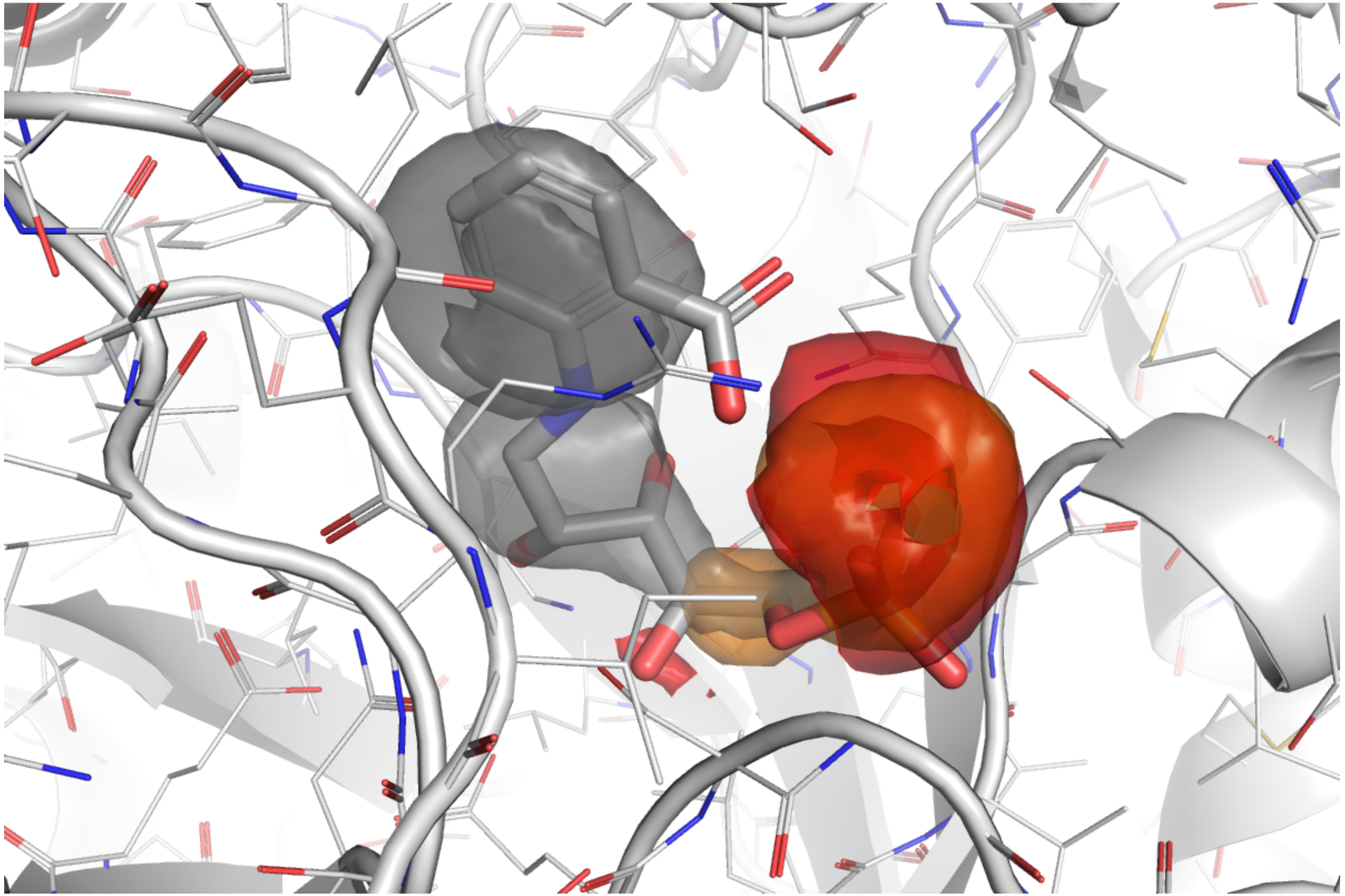
4PYX

Autoencoding Examples



1LBF

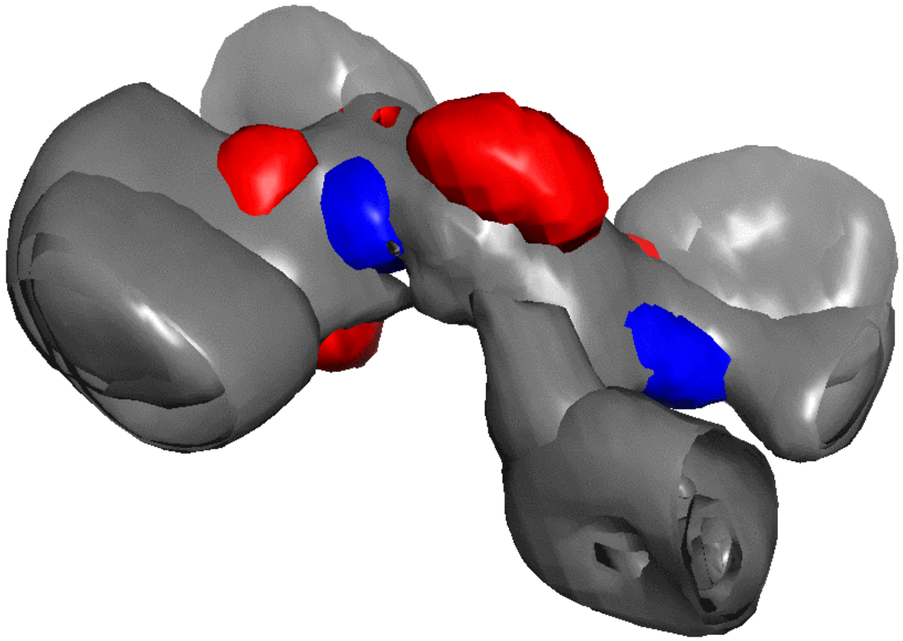
Autoencoding Examples



1LBF

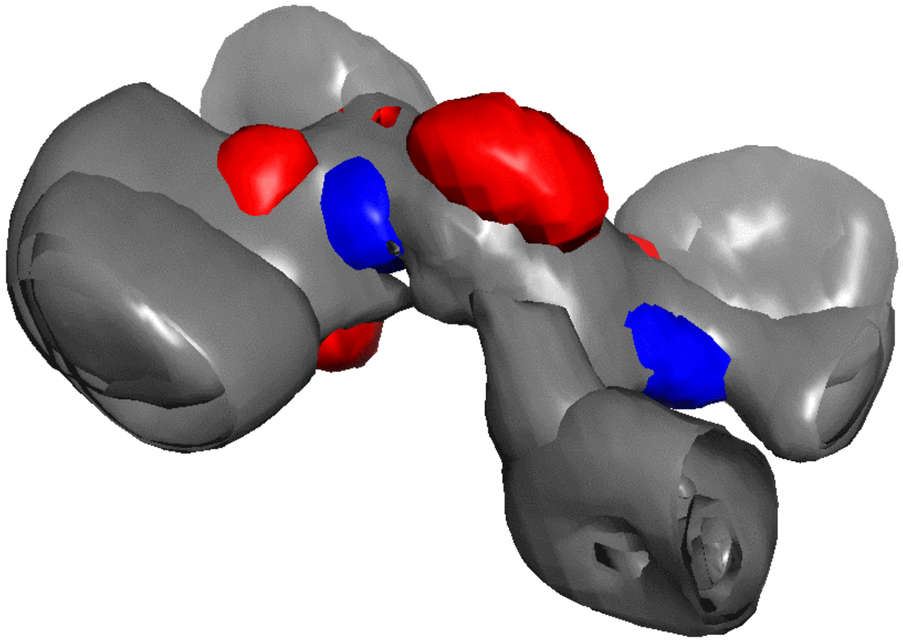
Atom Fitting

$$a^* = \underset{a}{\operatorname{argmin}} \|d - D(a)\|_2^2 + \lambda E(a)$$

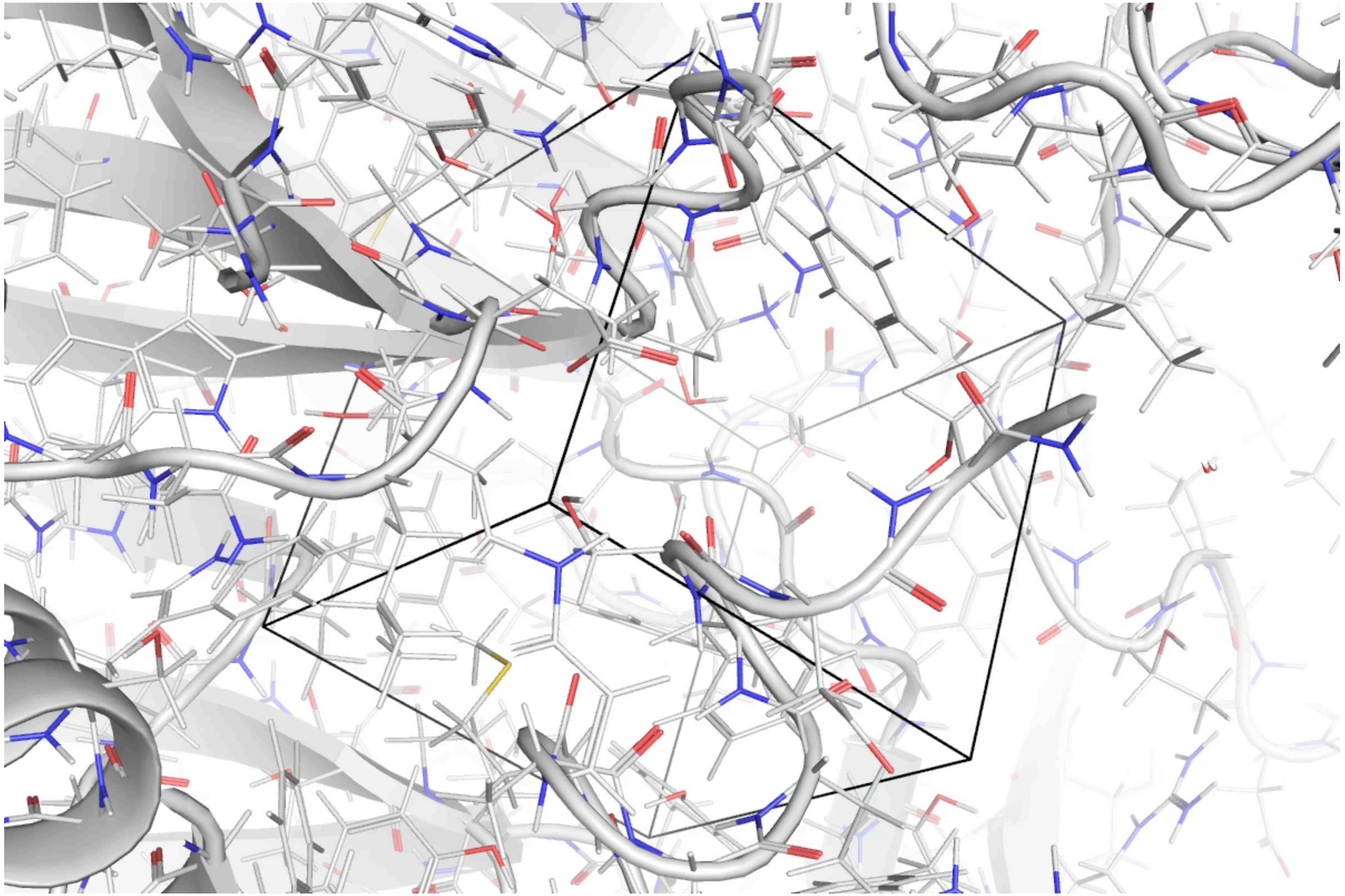


Atom Fitting

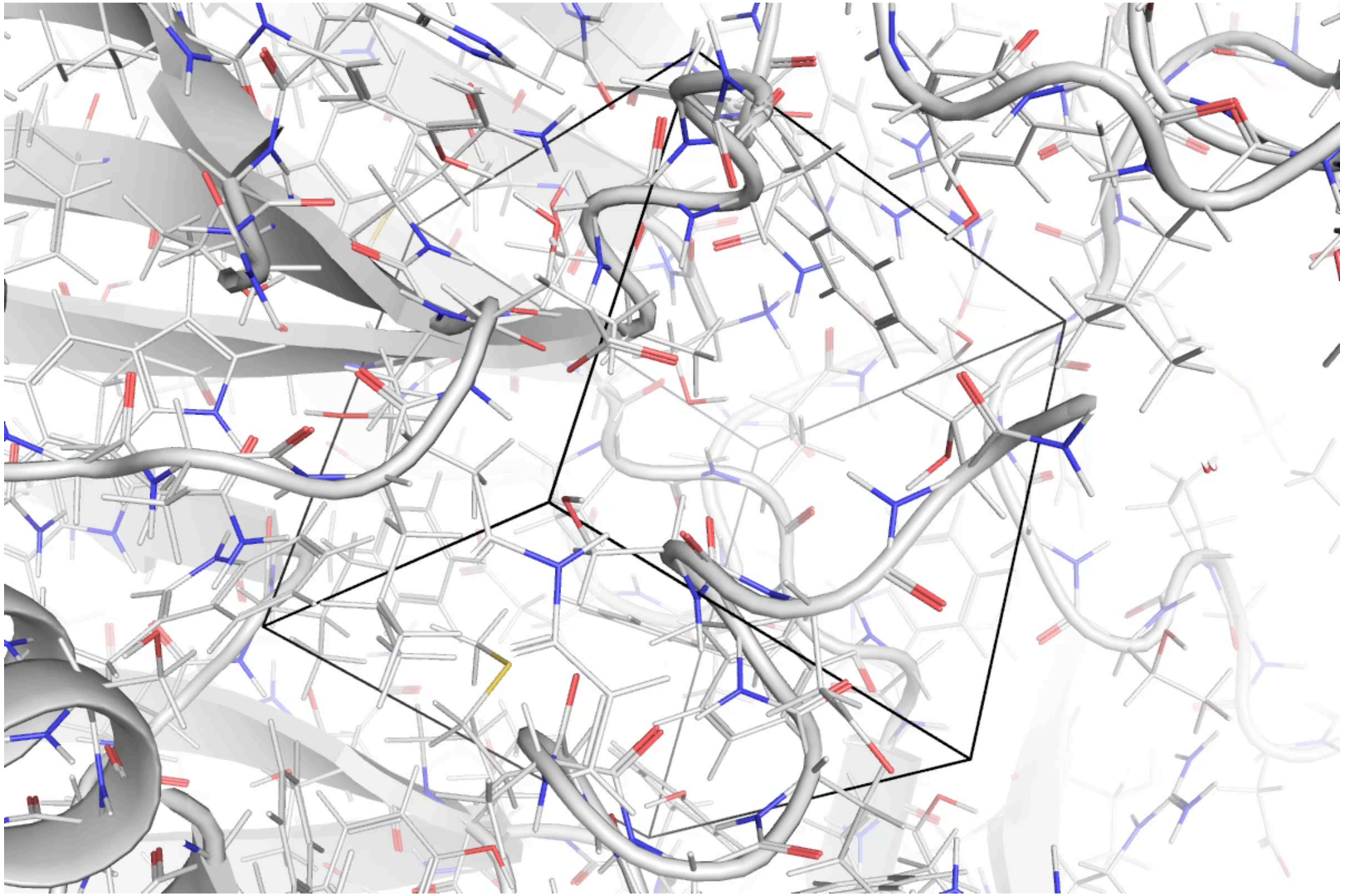
$$a^* = \underset{a}{\operatorname{argmin}} \|d - D(a)\|_2^2 + \lambda E(a)$$



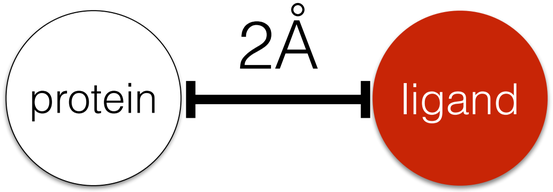
Conditioning on the Receptor



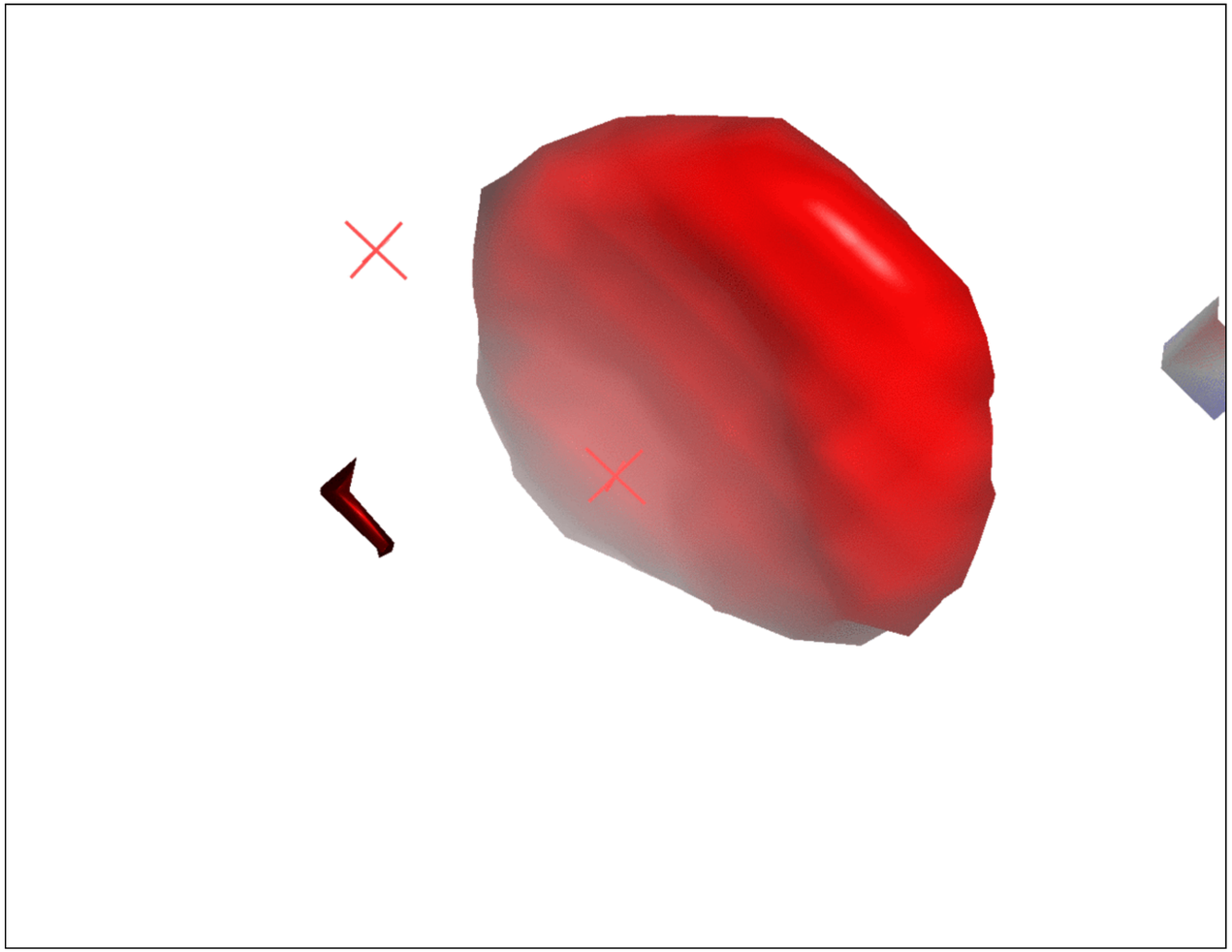
Conditioning on the Receptor



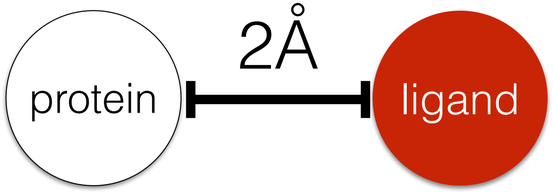
Iterpolating



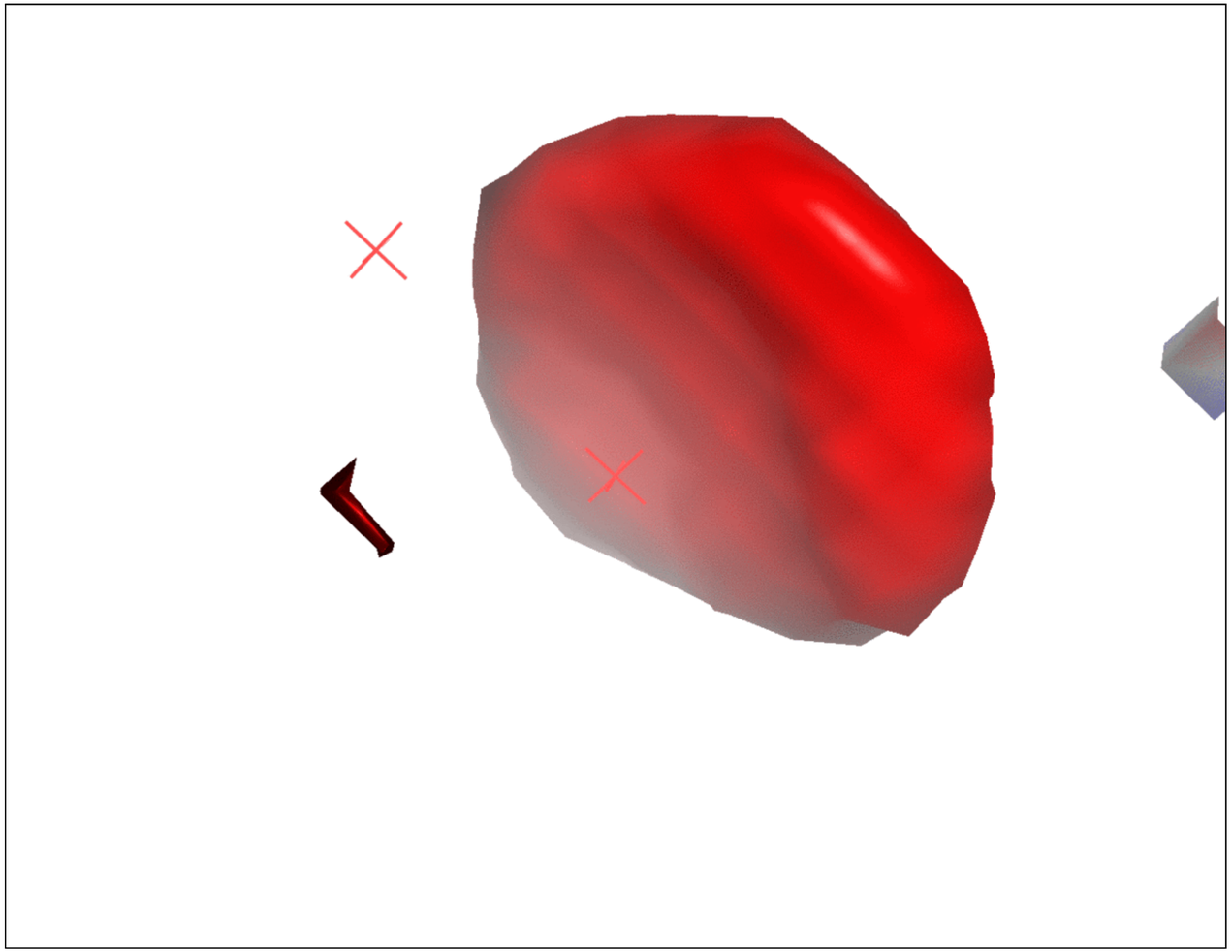
Two atom toy system



Iterpolating



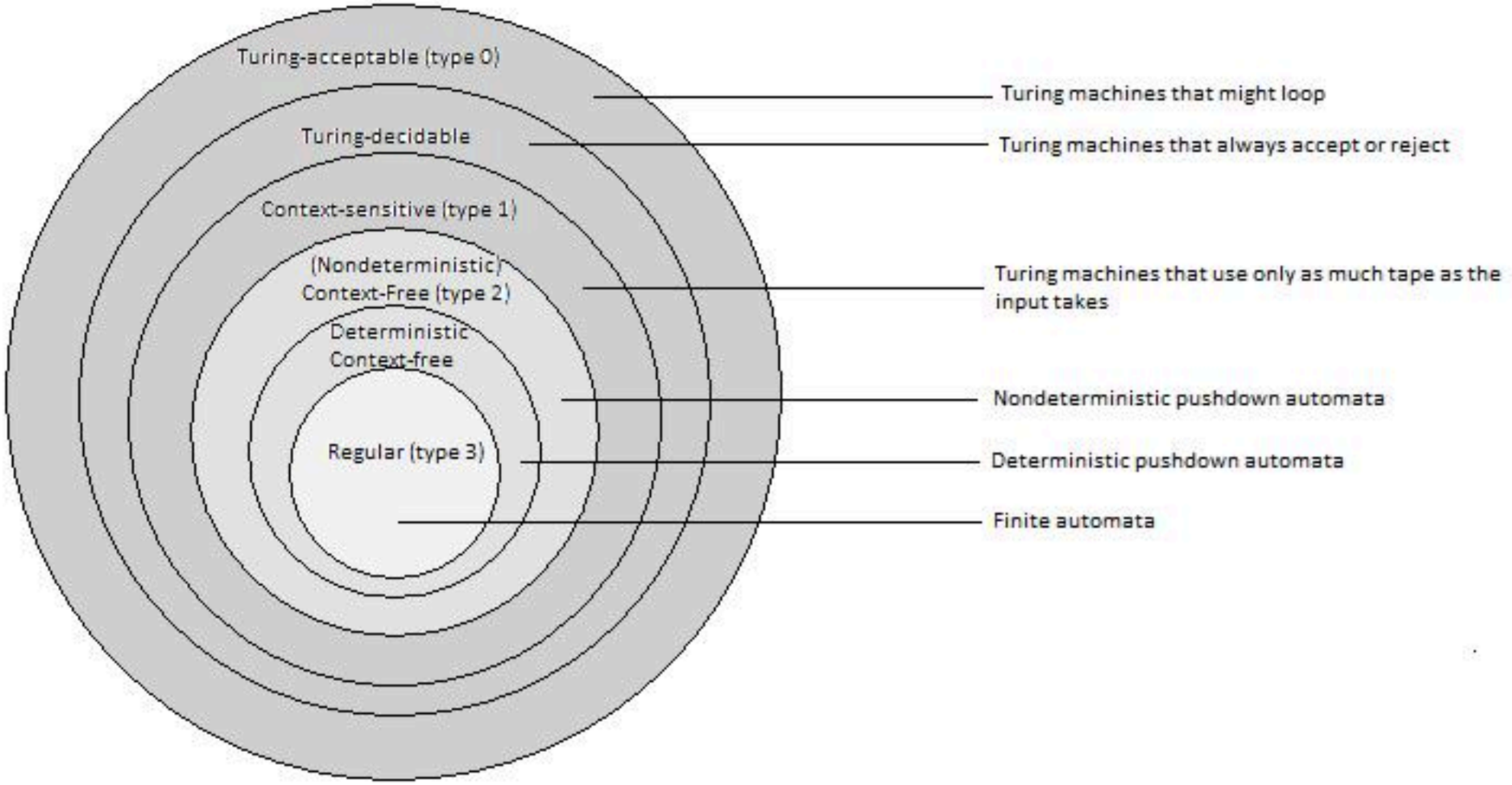
Two atom toy system

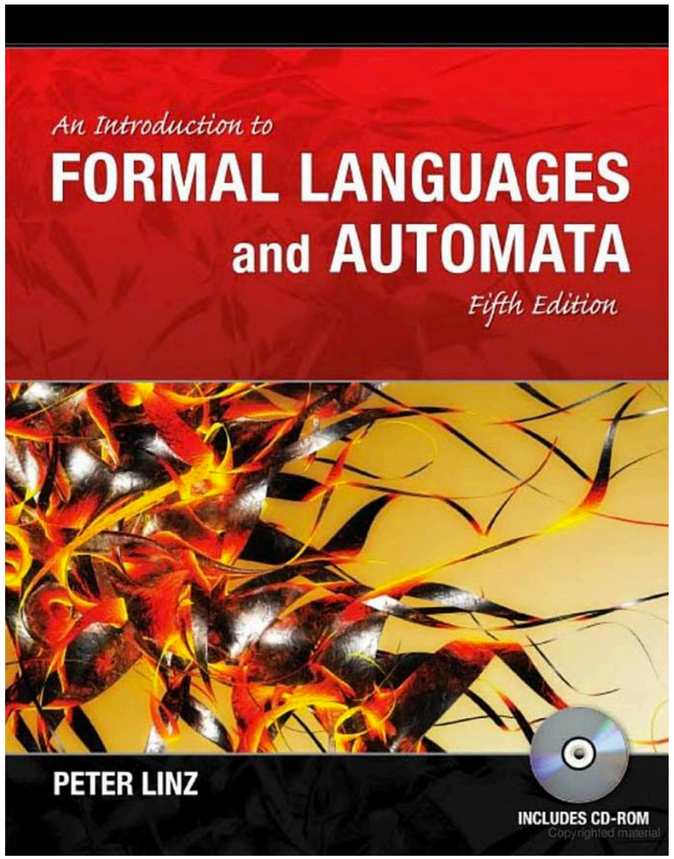


LALRNN

Removing the third dimension

Chomsky Hierarchy





http://www.cs.appstate.edu/~dap/classes/2490/chapter11print.html

Grammars

Balanced Parentheses

$S \rightarrow \varepsilon$

$S \rightarrow (S)$

$S \rightarrow SS$

```
()
(((())))()(())
()()()()()
```

Palindromes

$S \rightarrow \varepsilon$

$S \rightarrow aSa$

$S \rightarrow bSb$

```
aa
babbab
abbaabba
```

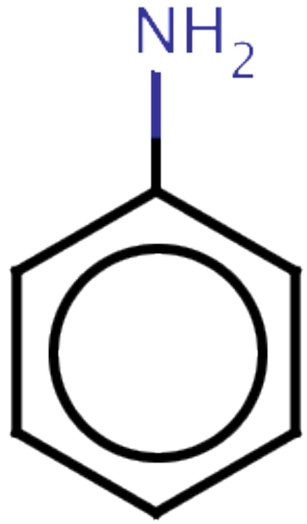
Arithmetic

```
E ::= id
    | num
    | E + E
    | E * E
    | ( E )
```

```
3 + 4 * 5
(3 + 4) * 5
```

SMILES

Section	Formal Grammar
	ATOMS
3.1	atom ::= bracket_atom | aliphatic_organic | aromatic_organic | '*'
	ORGANIC SUBSET ATOMS
3.1.5	aliphatic_organic ::= 'B' | 'C' | 'N' | 'O' | 'S' | 'P' | 'F' | 'Cl' | 'Br' | 'I'
3.5	aromatic_organic ::= 'b' | 'c' | 'n' | 'o' | 's' | 'p'
	BRACKET ATOMS
3.1.1	bracket_atom ::= '[' isotope? symbol chiral? hcount? charge? class? ']'
3.1.1	symbol := element_symbols | aromatic_symbols | '*'
3.1.4	isotope ::= NUMBER
3.1.1	element_symbols ::=
	'H' | 'He'
	|'Li'|'Be'| 'B' |'C' |'N' |'O' |'F' |'Ne'
	|'Na'|'Mg'| 'Al'|'Si'|'P' |'S' |'Cl'|'Ar'
	|'K' |'Ca'|'Sc'|'Ti'|'V' |'Cr'|'Mn'|'Fe'|'Co'|'Ni'|'Cu'|'Zn'|'Ga'|'Ge'|'As'|'Se'|'Br'|'Kr'
	|'Rb'|'Sr'|'Y' |'Zr'|'Nb'|'Mo'|'Tc'|'Ru'|'Rh'|'Pd'|'Ag'|'Cd'|'In'|'Sn'|'Sb'|'Te'|'I' |'Xe'
	|'Cs'|'Ba'| 'Hf'|'Ta'|'W' |'Re'|'Os'|'Ir'|'Pt'|'Au'|'Hg'|'Tl'|'Pb'|'Bi'|'Po'|'At'|'Rn'
	|'Fr'|'Ra'| 'Rf'|'Db'|'Sg'|'Bh'|'Hs'|'Mt'|'Ds'|'Rg'
	|'La'|'Ce'|'Pr'|'Nd'|'Pm'|'Sm'|'Eu'|'Gd'|'Tb'|'Dy'|'Ho'|'Er'|'Tm'|'Yb'|'Lu'
	|'Ac'|'Th'|'Pa'|'U' |'Np'|'Pu'|'Am'|'Cm'|'Bk'|'Cf'|'Es'|'Fm'|'Md'|'No'|'Lr'
3.5	aromatic_symbols ::= 'c' | 'n' | 'o' | 'p' | 's' | 'se' | 'as'
	CHIRALITY
3.9	chiral ::= '@'
	| '@@'
	| '@TH1' | '@TH2'
	| '@AL1' | '@AL2'
	| '@SP1' | '@SP2' | '@SP3'
	| '@TB1' | '@TB2' | '@TB3' | ... | '@TB29' | '@TB30'
	| '@OH1' | '@OH2' | '@OH3' | ... | '@OH29' | '@OH30'
	HYDROGENS
3.1.2	hcount ::= 'H'
	| 'H' DIGIT
	CHARGE
3.1.3	charge ::= '-'
	| '-' DIGIT
	| '+'
	| '+' DIGIT
	| '--' *deprecated*
	| '++' *deprecated*
	ATOM CLASS
3.1.7	class ::= ':' NUMBER
	BONDS AND CHAINS
3.2, 3.9.3	bond ::= '-' | '=' | '#' | '$' | ':' | '/' | '\'
3.4	ringbond ::= bond? DIGIT
	| bond? '%' DIGIT DIGIT
3.3	branched_atom ::= atom ringbond* branch*
	branch ::= '(' chain ')'
	| '(' bond chain ')'
	| '(' dot chain ')'
	chain ::= branched_atom
	| chain branched_atom
	| chain bond branched_atom
	| chain dot branched_atom
3.7	dot ::= '.'
	SMILES STRINGS
3.10	smiles ::= chain terminator
	terminator ::= SPACE TAB | LINEFEED | CARRIAGE_RETURN | END_OF_STRING

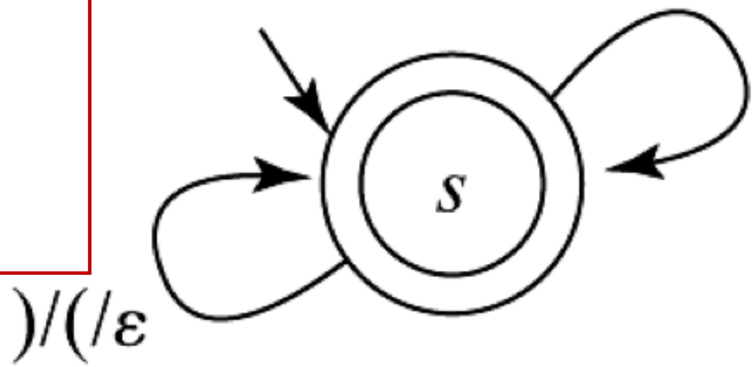


`c1ccccc1N`

Push Down Automata

Balanced Parentheses

If input is) and (is on stack top, then (is popped and nothing is pushed to stack.



If input is (, what's on stack top doesn't matter and a (is pushed to stack.

Bottom Up Parsing

A PDA can be implemented with a *parse table*

state	action			goto	
	ident	+	$	E	T
0	**s3**			**g1**	**g2**
1			**a**		
2		**s4**	**r2**		
3		**r3**	**r3**		
4	**s3**			**g5**	**g2**
5			**r1**		

S → E$
E → T + E
E → T
T → *identifier*

states != rules

x + y$

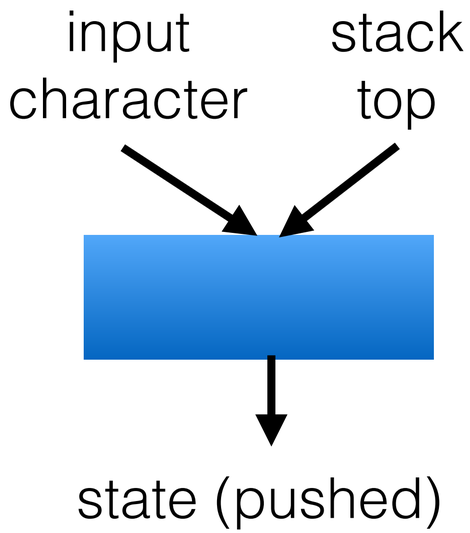
```
while(true)
  s = state on top of stack
  a = current input token
  if(action[s][a] == sN)                shift
    push N
    a = next input token
  else if(action[s][a] == rR)           reduce
    remove rhs of rule R from stack
    X = lhs of rule R
    N = state on top of stack
    push goto[N][X]
  else if(action[s][a] == a)            accept :-)
    return success
  else                                  error
    return failure
```

The NN Part

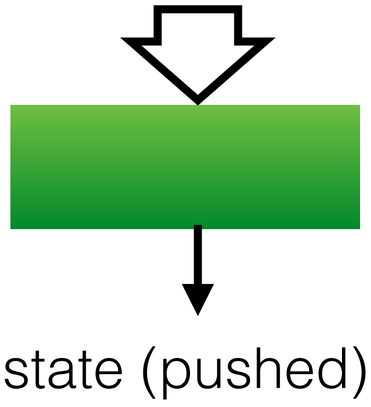
Implement every state as its own neural network that calculates a function of the input in the context of the parse (encoder) or outputs a syntactically correct string according to the rules of the grammar (generator)

Encoder

Shift State

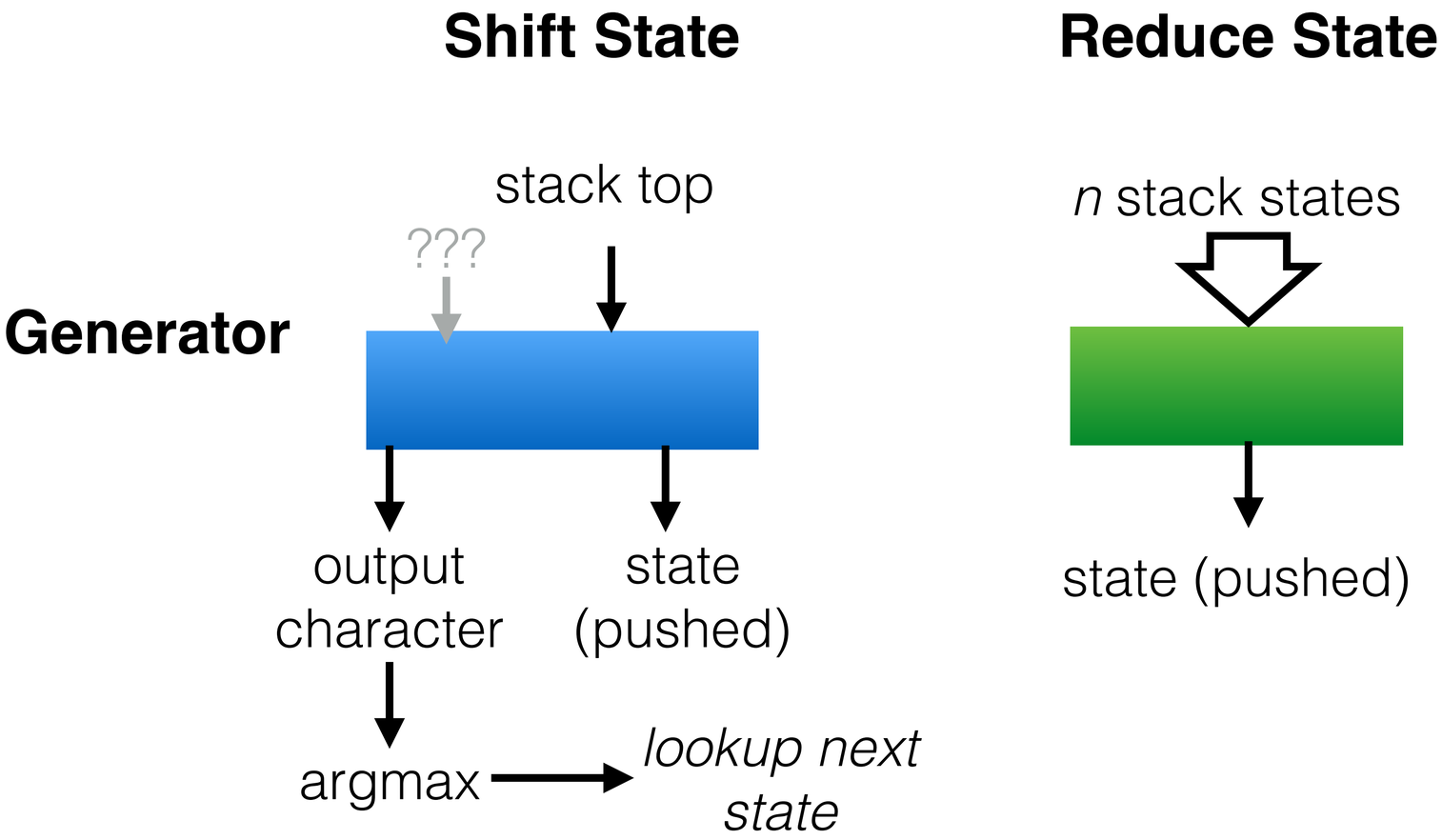


Reduce State

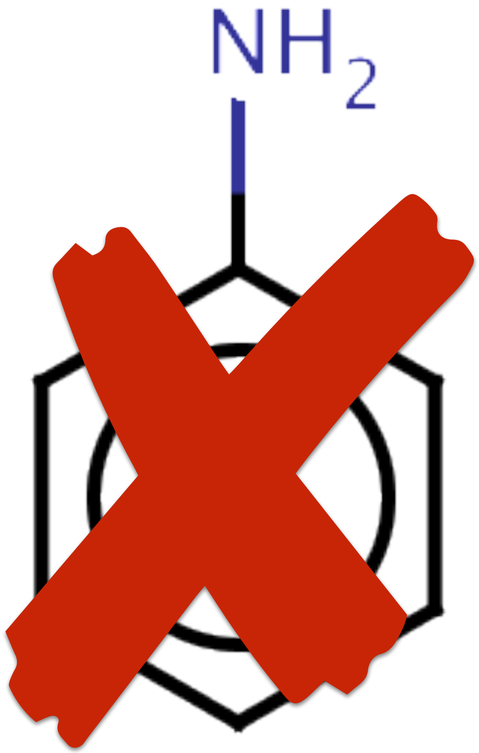


The NN Part

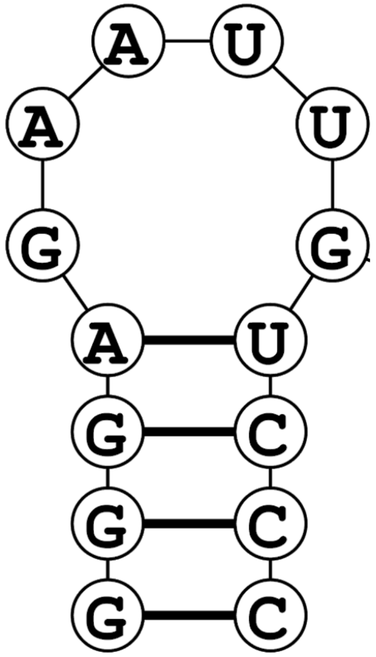
Implement every state as its own neural network that calculates a function of the input in the context of the parse (encoder) or outputs a syntactically correct string according to the rules of the grammar (generator)



Does it work???



c1ccccc1N



GGGAGAAUUGUCCC

((((.....))))

$S \rightarrow .$

$S \rightarrow ()$

$S \rightarrow (S)$

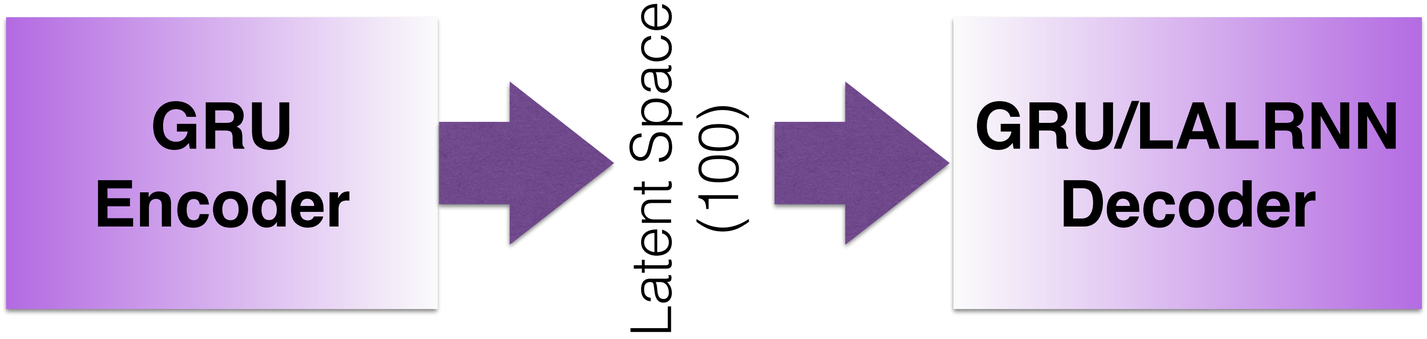
$S \rightarrow S(S)$

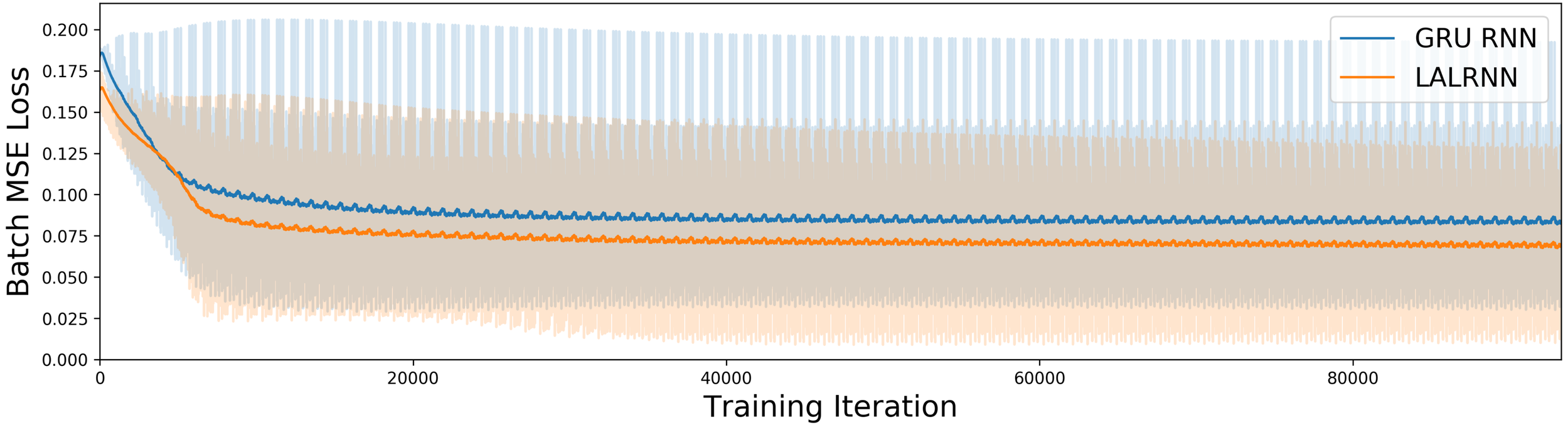
$S \rightarrow S.$

$S \rightarrow S()$

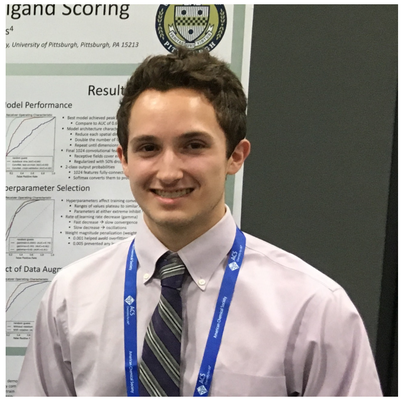
state	.	(	)	S
0	s6	s1		g5
1	s6	s1	s7	g4
2	s6	s1	s11	g3
3	s10	s2	s9	
4	s10	s2	s8	
5	s10	s2		
6	Reduce S $\rightarrow$.			
7	Reduce S $\rightarrow$ ()			
8	Reduce S $\rightarrow$ (S)			
9	Reduce S $\rightarrow$ S(S)			
10	Reduce S $\rightarrow$ S.			
11	Reduce S $\rightarrow$ S()			
12	END			

LALRNN vs GRU

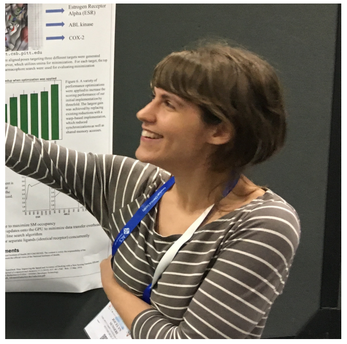




Acknowledgements



Matt Ragoza



Jocelyn Sunseri

Paul Francoeur

Department of Computational and Systems Biology

AI GRANT

National Institute of General Medical Sciences
R01GM108340

NVIDIA

Google Cloud

github.com/gnina

http://bits.csb.pitt.edu

@david_koes