Backstory:

One of the most defining aspects of myself is my love for music and playing piano. I love listening to and playing music, but I really wanted to listen to a piece that I could call my own. I dreamt about writing a song that was on the radio, or even on a small podcast channel. Over the summer, I wanted to tap into my creativity and compose music, but since I suck at music composition, the computer geek side of me thought, "what if I let a computer make music?" So I spent the entire week gathering a couple thousand piano pieces by my favorite composers, (Chopin, Beethoven, Bach, Mozart, Brahms, Liszt, Schubert, Handel, Haydn, Tchaikovsky, Debussy, and the list goes on) from a small German site I had spent 20 euros on, and I trained a transformer neural network to generate new music. I had just learned about this type of model from the recent research paper, *Attention Is All You Need* ([1706.03762] Attention Is All You Need), that came out in 2017, and I thought I would try to implement it. All it did was look at a sequence of notes and chords and predict which note/chord should be next. Doing this thousands of times, I would be able to generate my own song in the style of all the composers I loved.

The song was absolute garbage, so in researching how to improve the model to create better music, I came across a Medium post about how someone was able to generate protein sequences and turn them into music (Turning a Protein into Music (Part 3) — Recursive Neural Network | by Ford Combs). It was amazing to see how two completely unrelated things could be brought together by machine learning and its ability to identify patterns in two disparate types of data, but what was even more beautiful to me was how something as abstract and random as nature could create music which would typically need some sort of structure to sound pleasant. To me, this sparked an idea of a relationship between all things and made me realize that everything has structure, and everything has patterns, from the patterns the neural network tried so hard to extract from piano pieces, to the patterns in protein sequences. Machine learning is simply a way to analyze these patterns in an almost superhuman way, to help create, identify, and discover.

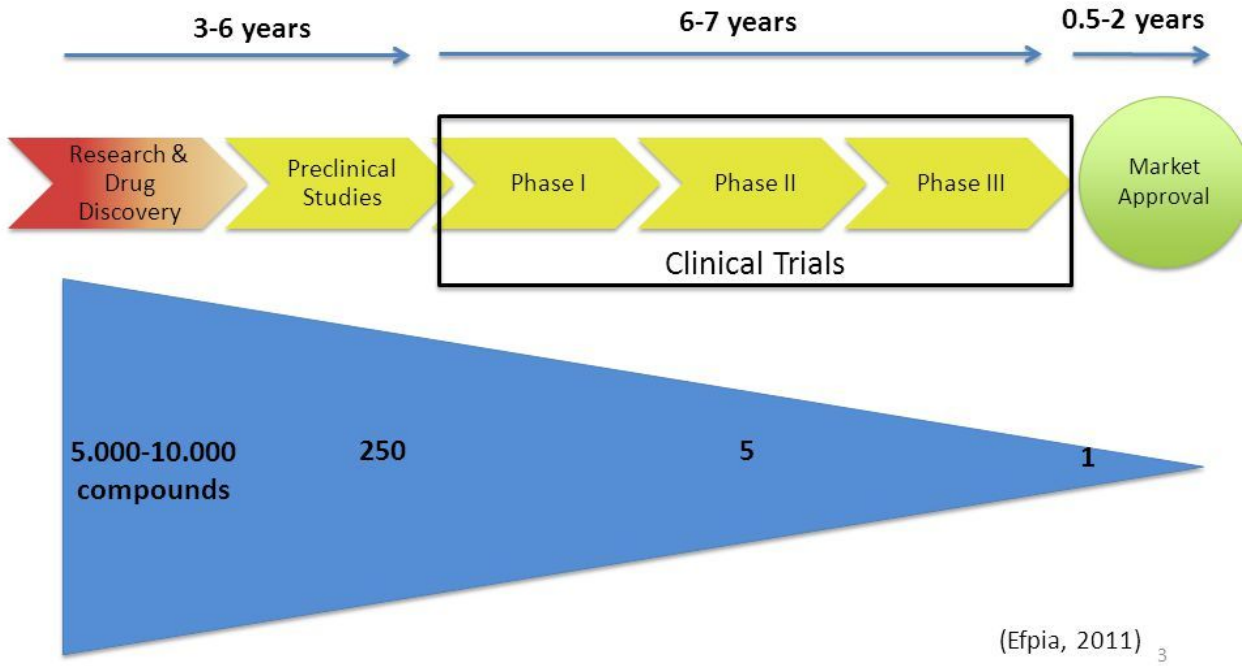
Being in quarantine during coronavirus then inspired me further to use machine learning to understand the patterns of different proteins and the drugs that inhibit them to help people overcome such diseases. Besides being bored at home with nothing but my hobbies to pursue, I was genuinely fearful of how the virus had affected so many people. My mom is a doctor who goes out into the field everyday; some of her patients have even been infected. My grandparents live in India, which has terrible medical infrastructure, and if they were infected, they would surely not survive. I wanted to help in any way I could and use my skills and hobbies to my advantage. I found out that these drugs can be written as 1-D sequences using a format called SMILES (simplified molecular-input line-entry system) like how I read the notes from piano pieces as sequences. I used nearly the same techniques to generate novel drug-like compounds.

But given that the generated music was so awful, how could I know that these generated drugs were not the same? I thought to myself, “what makes a drug ‘good’ in the first place?” After intensive research, I found many objectives scientists have to balance and optimize in making a good drug. The drug needs to have high binding affinity (which means it needs to bind to the target protein very well), a high selectivity (which means it needs to bind *only* to the target protein and not to any other “off targets”), a low toxicity (which means it should not have a high risk of poisoning the client like compounds such as cyanide), as well as many other criteria. I figured I should try and start out with the one I thought was most important: binding affinity. If the drug does not bind well to the target protein, then it is essentially useless, and every other criterion does not even matter. The next question became, “how do I make sure that the drug has a high affinity?” Affinity can be measured in different ways, but the one I chose was the one that had the most recorded data, which was the IC50 (inhibitory concentration) value. IC50 is a measure of the potency of a substance in inhibiting a specific biological or biochemical function. It basically asks how much of the drug will I need to inhibit the protein. The less of the drug needed, the better it binds to the protein. Therefore, IC50 and affinity are inversely related, so I must generate drugs with a low predicted IC50 value. The dataset I found was a simple CSV file with only three columns: the molecule in the 1-D sequence in SMILES format, the protein in a 1-D sequence of amino acids which is in FASTA (Fast Adaptive Shrinkage Threshold Algorithm) format, and the IC50 value. If patterns can be extracted from drug sequences and music can be created from protein sequences, patterns could definitely be extracted from protein sequences, so I created a multimodal model (model with multiple inputs) that takes both sequences and outputs a predicted IC50 value. The accuracy was very underwhelming, even after cleaning the data and changing model architecture. After talking to an AI researcher, I realized that I had accidentally stumbled into one of the hardest ongoing problems; protein folding (predicting the 3-D structure of proteins). The interaction of a molecule binding to a protein depends on both of their 3-D structures, and I forced my model to do this using 1-D sequences. There were many ways to tackle this problem, the best I thought would be by using a 3-D convolutional neural network or by using a graph convolutional neural network, but this approach would be limited to known protein-ligand complex structures, and only 25,000 ligands are reported in PDB. I decided to work with what I had and continue with the project.

I eventually came up with an iterative process, in which the first model uses whatever existing inhibitors there are for a target protein to create new compounds. The second model then scores these compounds by predicting the IC50 value of the compound to the target protein. The next iteration would then use the best compounds (the ones with the lowest IC50) to create even more compounds to be scored, and so on until the best drug is generated. Of course, had the second model been more accurate, it would have definitely helped in creating the best drug candidate.

Project:

Drug discovery is a very long and expensive process. The average time from FDA application to approval of drugs is 12 years, and the estimated average cost of taking a new drug from concept to market exceeds $1 billion. Of up to 10,000 compounds tested, only one may end up becoming a drug that reaches the market.



How can we make this process more time/cost effective?

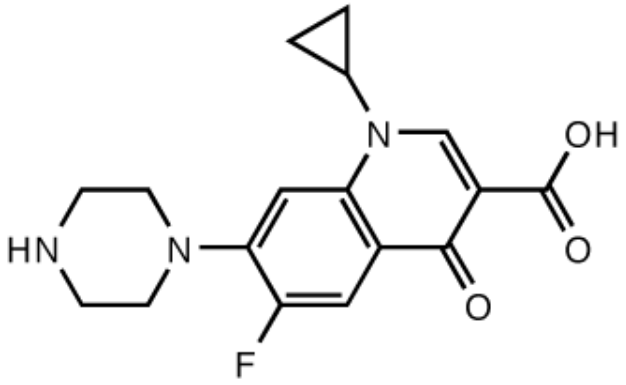
Model #1:

The first model is able to generate novel compounds based on molecules in SMILES format. It uses LSTM cells to look at the previous 100 elements of a SMILES sequence to predict what element should be next.

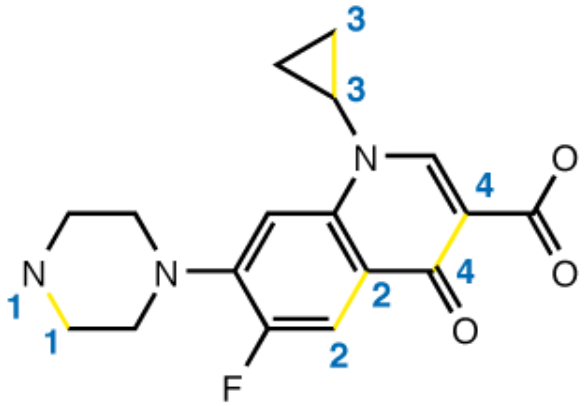
SMILES (Simplified Molecular-Input Line-Entry System)

* Atoms are shown by atomic symbols
* Hydrogen atoms are assumed to fill spare valencies
* Adjacent atoms are connected by single bonds
    * double bonds are shown by "="
    * triple bonds are shown by "#"
* Branching is indicated by parenthesis
* Ring closures are shown by pairs of matching digits

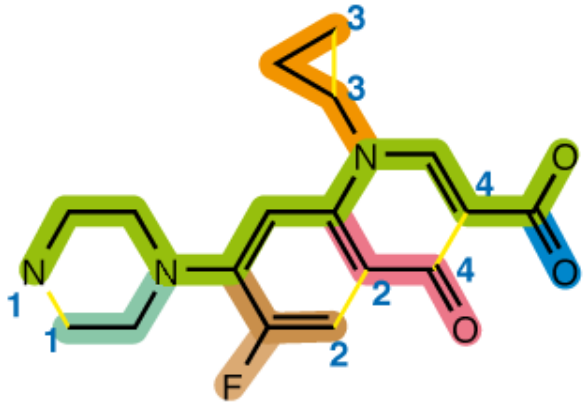
A



B



C

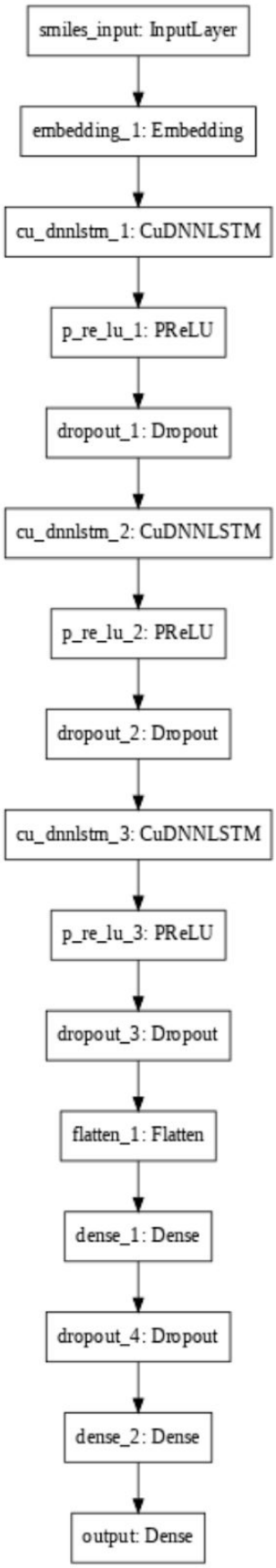


D

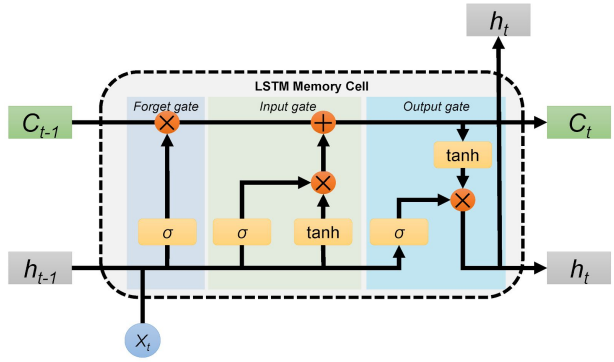
N1CCN(CC1)C(C(F)=C2)=CC(=C2C4=O)N(C3CC3)C=C4C(=O)O

Model #1 Architecture (>16 million parameters):

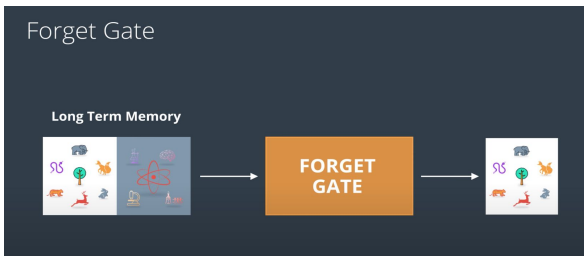
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
smiles_input (InputLayer)    (None, 100)               0
_________________________________________________________________
embedding_1 (Embedding)      (None, 100, 128)          5504
_________________________________________________________________
cu_dnnlstm_1 (CuDNNLSTM)     (None, 100, 256)          395264
_________________________________________________________________
p_re_lu_1 (PReLU)            (None, 100, 256)          25600
_________________________________________________________________
dropout_1 (Dropout)          (None, 100, 256)          0
_________________________________________________________________
cu_dnnlstm_2 (CuDNNLSTM)     (None, 100, 512)          1576960
_________________________________________________________________
p_re_lu_2 (PReLU)            (None, 100, 512)          51200
_________________________________________________________________
dropout_2 (Dropout)          (None, 100, 512)          0
_________________________________________________________________
cu_dnnlstm_3 (CuDNNLSTM)     (None, 100, 256)          788480
_________________________________________________________________
p_re_lu_3 (PReLU)            (None, 100, 256)          25600
_________________________________________________________________
dropout_3 (Dropout)          (None, 100, 256)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 25600)             0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               13107712
_________________________________________________________________
dropout_4 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
output (Dense)               (None, 43)                11051
=================================================================
Total params: 16,118,699
Trainable params: 16,118,699
Non-trainable params: 0
_________________________________________________________________
```


smiles_input: InputLayer
embedding_1: Embedding
cu_dnnlstm_1: CuDNNLSTM
p_re_lu_1: PReLU
dropout_1: Dropout
cu_dnnlstm_2: CuDNNLSTM
p_re_lu_2: PReLU
dropout_2: Dropout
cu_dnnlstm_3: CuDNNLSTM
p_re_lu_3: PReLU
dropout_3: Dropout
flatten_1: Flatten
dense_1: Dense
dropout_4: Dropout
dense_2: Dense
output: Dense

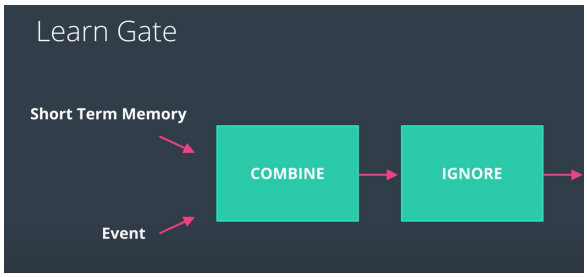
How do LSTM (Long-Short Term Memory) cells work?

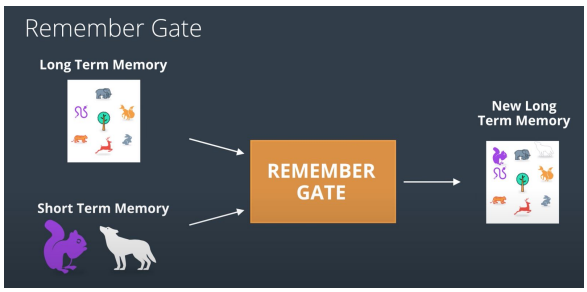


Forget Gate:

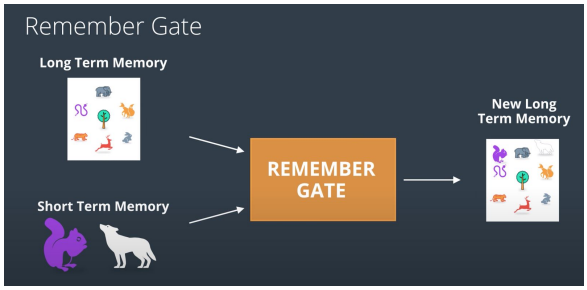


Input Gate (includes the Learn Gate and Remember Gate):





Output Gate (AKA the Use Gate):



Model #1 was trained on 1.55 million molecules (~75,000,000 samples) and validated on 41,378 molecules (~2,000,000 samples). All compounds come from the ChEMBL database (a manually curated chemical database of bioactive molecules with drug-like properties).

There are 3 different generation techniques:

1. Generate Randomly
    a. This technique takes a randomly sampled sequence and builds off of it, outputting a random drug-like compound
2. Generate Methodically
    a. This technique takes a list of molecules, creates sequences from them, and builds a molecule using each sequence
3. Augment
    a. This technique replaces random parts of molecules with model predictions
        i. Makes sure the molecules are at least 20% similar

Before outputting any molecule the model generates, it makes sure:

1. The molecule is valid
    a. It follows the laws of chemistry and SMILES format
2. The molecule is drug-like
    a. It passes Lipinski's Rule of Five (a set of constraints in order to maintain drug-like character within the compounds - otherwise the drug may have poor permeation/absorption, faster rate of metabolism and excretion, unfavorable distribution, and might be toxic)
        i. H bond donors (OH and NH) <= 5
        ii. Molecular weight <= 500 Daltons or g/mol (same thing)
        iii. Log P <= 5
        iv. H bond acceptors (N and O) <= 10
3. The molecule is unique
    a. It is not in the list of compounds it has already generated

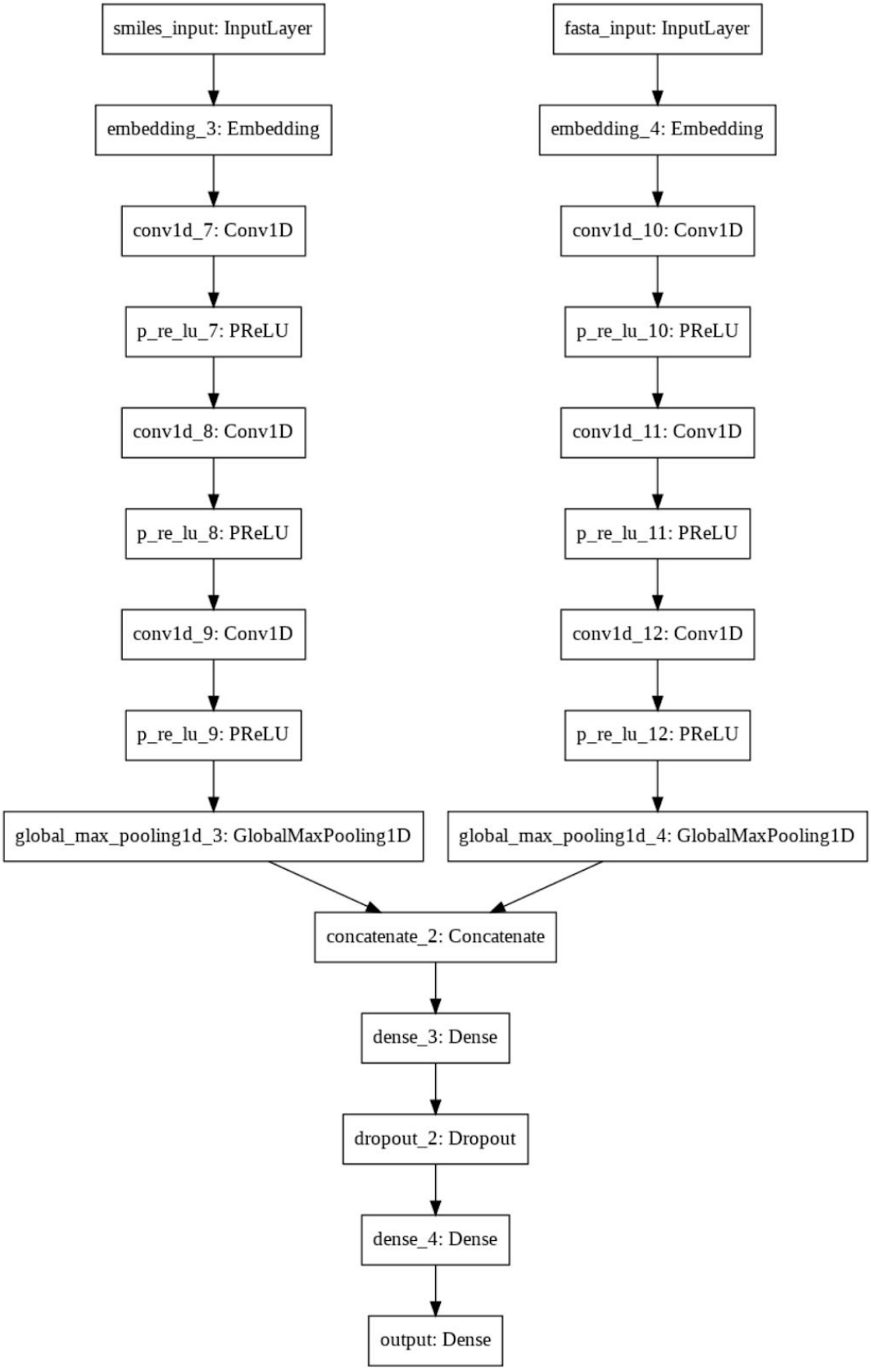
Model #2:

The second model predicts how well a molecule will bind to a target protein. It uses 1-D convolutional filters to extract information from both the protein sequence and molecule in SMILES format, to predict the IC50 (inhibitory concentration) value of the molecule to the protein. IC50 is a measure of the potency of a substance in inhibiting a specific biological or biochemical function. It essentially asks how much of a certain drug is needed to inhibit a certain protein. The less of the drug needed, the better it must bind to the protein. Predicting the IC50 value, will help understand the binding affinity because they are inversely related. The IC50 value must be minimized in order to maximize affinity.

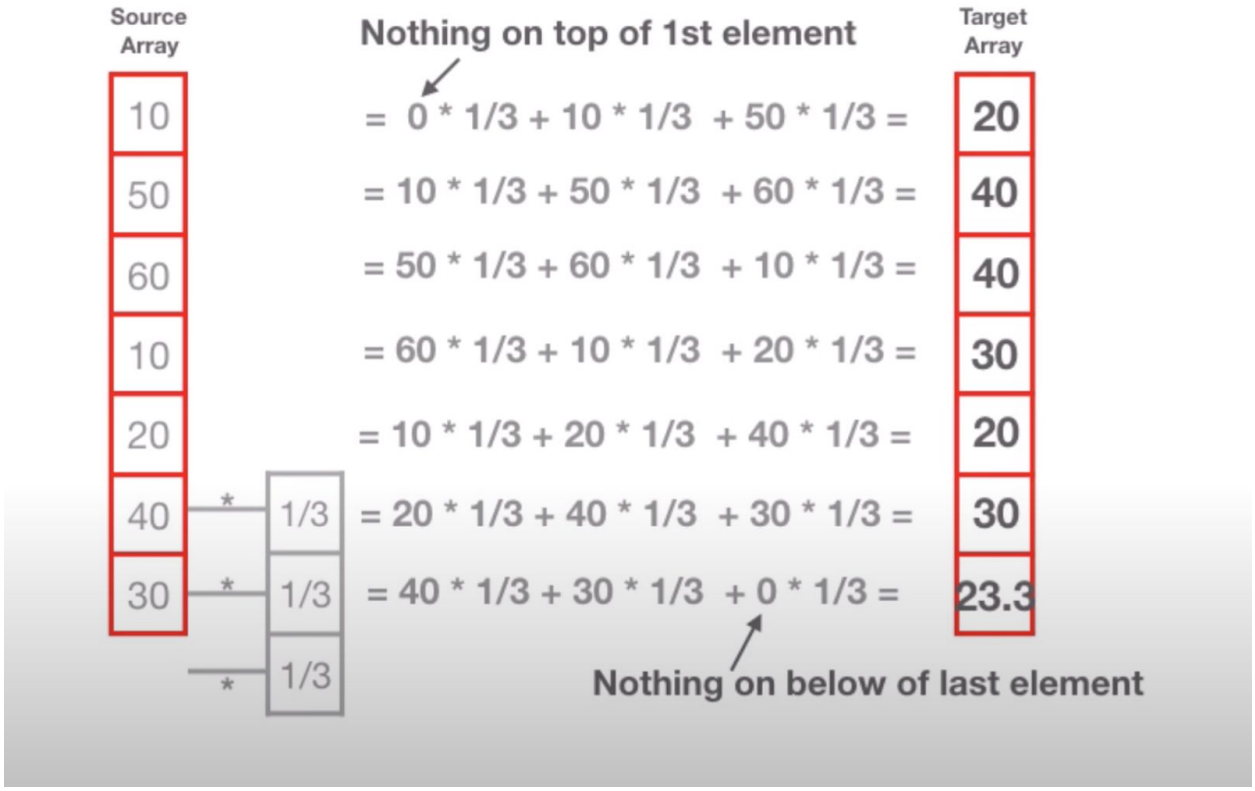
Model #2 Architecture (>2 million parameters):

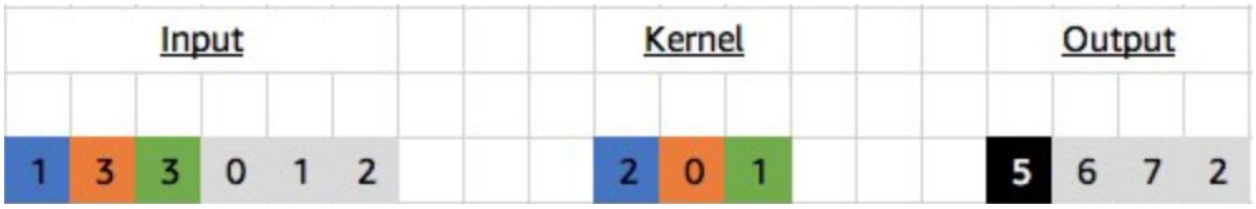
Layer (type)	Output Shape	Param #	Connected to
smiles_input (InputLayer)	(None, 100)	0	
fasta_input (InputLayer)	(None, 5000)	0	
embedding_9 (Embedding)	(None, 100, 128)	6144	smiles_input[0][0]
embedding_10 (Embedding)	(None, 5000, 256)	5888	fasta_input[0][0]
conv1d_25 (Conv1D)	(None, 98, 32)	12320	embedding_9[0][0]
conv1d_28 (Conv1D)	(None, 4998, 32)	24608	embedding_10[0][0]
p_re_lu_25 (PReLU)	(None, 98, 32)	3136	conv1d_25[0][0]
p_re_lu_28 (PReLU)	(None, 4998, 32)	159936	conv1d_28[0][0]
conv1d_26 (Conv1D)	(None, 96, 64)	6208	p_re_lu_25[0][0]
conv1d_29 (Conv1D)	(None, 4996, 64)	6208	p_re_lu_28[0][0]
p_re_lu_26 (PReLU)	(None, 96, 64)	6144	conv1d_26[0][0]
p_re_lu_29 (PReLU)	(None, 4996, 64)	319744	conv1d_29[0][0]
conv1d_27 (Conv1D)	(None, 94, 128)	24704	p_re_lu_26[0][0]
conv1d_30 (Conv1D)	(None, 4994, 128)	24704	p_re_lu_29[0][0]
p_re_lu_27 (PReLU)	(None, 94, 128)	12032	conv1d_27[0][0]
p_re_lu_30 (PReLU)	(None, 4994, 128)	639232	conv1d_30[0][0]
global_max_pooling1d_9 (GlobalM	(None, 128)	0	p_re_lu_27[0][0]
global_max_pooling1d_10 (Global	(None, 128)	0	p_re_lu_30[0][0]
concatenate_5 (Concatenate)	(None, 256)	0	global_max_pooling1d_9[0][0] global_max_pooling1d_10[0][0]
dense_9 (Dense)	(None, 1024)	263168	concatenate_5[0][0]
dropout_5 (Dropout)	(None, 1024)	0	dense_9[0][0]
dense_10 (Dense)	(None, 512)	524800	dropout_5[0][0]
output (Dense)	(None, 1)	513	dense_10[0][0]

Total params: 2,039,489
Trainable params: 2,039,489
Non-trainable params: 0


smiles_input: InputLayer
fasta_input: InputLayer
embedding_3: Embedding
embedding_4: Embedding
conv1d_7: Conv1D
conv1d_10: Conv1D
p_re_lu_7: PReLU
p_re_lu_10: PReLU
conv1d_8: Conv1D
conv1d_11: Conv1D
p_re_lu_8: PReLU
p_re_lu_11: PReLU
conv1d_9: Conv1D
conv1d_12: Conv1D
p_re_lu_9: PReLU
p_re_lu_12: PReLU
global_max_pooling1d_3: GlobalMaxPooling1D
global_max_pooling1d_4: GlobalMaxPooling1D
concatenate_2: Concatenate
dense_3: Dense
dropout_2: Dropout
dense_4: Dense
output: Dense

How do 1-D convolutions work?





The advantage of using a CNN architecture is that it is able to extract the spatial features from the data using kernels, which other networks are unable to do. This works great for the molecular and protein data because it is able to better understand the spatial properties of both structures and their interactions. This is not needed in the generative model because it does not need to understand spatial properties, but rather what should come next in a sequence; a problem LSTM cells are historically known to thrive on. This is because LSTMs have forms of memory given in the name (Long-Short Term Memory), allowing it to look back at previous information when making calculations. This allows it to better predict the next character in a sequence by considering what precedes it. LSTM cells struggle with larger input sequences however, like a 5000 character protein sequence, hence the use of a CNN architecture.

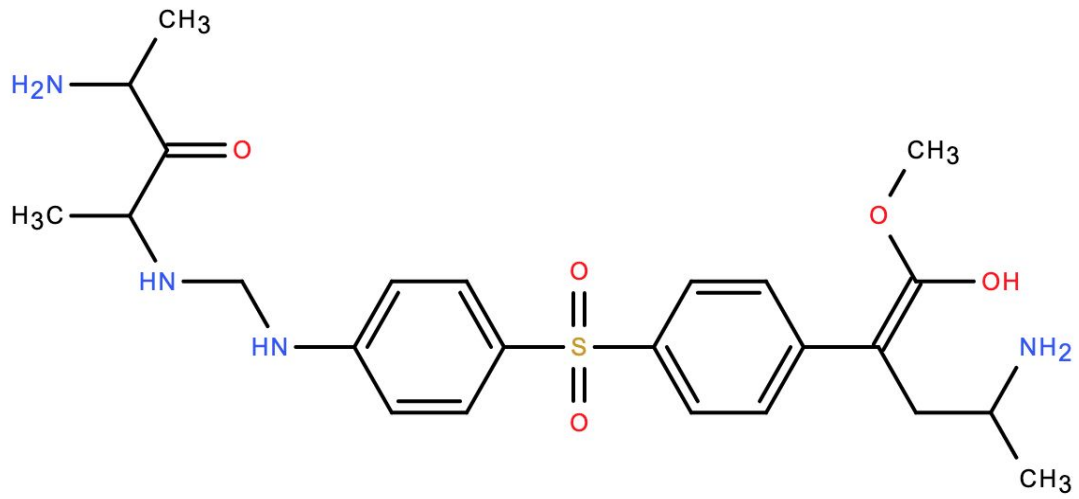
Model #2 was trained on 500,000 interactions between 3,811 proteins and 323,728 molecules, and validated on 1,577 interactions between 680 proteins and 1,575 molecules. All interactions come from the KIBA (Kinase Inhibitor Bio-Activity) dataset.

Iterative Process:
Both models were used to create an iterative process in which the best drug candidates can be created. Model #1 uses whatever existing inhibitors there are for a target protein to create new compounds by generating randomly, generating methodically, and augmenting, with Model #2 then scoring these compounds by predicting the IC50 value of the compound to the target protein. The next iteration would then use the best compounds (the ones with the lowest IC50) to create even more compounds to be scored, and so on until the best drugs are generated.

Testing:
"Of the 29 SARS-CoV-2 proteins, four make up the virus's actual structure, including the S protein. One group of the other 25 coronavirus proteins regulates how the virus assembles copies of itself and how it sneaks past the host immune system. These so-called nonstructural proteins are expressed as two huge polyproteins that are then cleaved into 16 smaller proteins. An enzyme called the main protease, which performs 11 of those cleavages, is also a highly promising drug target." (What do we know about the novel coronavirus's 29 proteins?)

The iterative process to create the best drug candidate was tested on this protein (SARS coronavirus 3C-like proteinase). After a few iterations, the models came up with a drug with a better predicted affinity than existing inhibitors of the protein.

Drug Candidate: CC(C(=O)C(C)N)NCNc1ccc(S(=O)(=O)c2ccc(C(CC(C)N)=C(O)OC)cc2)cc1



Predicted IC50: 306.12613 nM