# Initial Results of DeepAtom on DUD-E

## Weekly update

Week 2 April, 2021

# Overview

Recap of network architecture
- i. Atom information integration block
- ii. Stacked feature extraction block
- iii. Classification block

Results on DUD-E
- i. Data
- ii. Performance

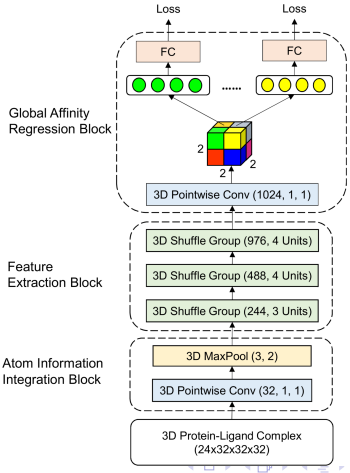
# Updates Since Last Week

1. Implemented the data pipeline.
2. Implemented DeepAtom's Shufflenet-based CNN architecture.
3. Did some optimization for DeepAtom on DUD-E to understand areas of improvement.

# The three block overview

**DeepAtom has three blocks:**

1. Atom information integration block.
2. Stacked feature extraction block.
3. Classification block.

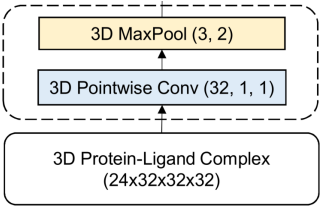


# i. Atom information integration block

**Atomic information across the ~~24~~ 8 channels is aggregate in a pointwise fashion, and the output is then pooled.**

1. $\text{PWConv}(W, h)_{(i,j,k)} = \Sigma_m^{24} W_m \cdot h_{(i,j,k,m)}$ is used to map the $8 \times 32^3$ input to $32^3$.
2. The max-pooling downsamples the tensor to $16^3$.

Semantically, this block processes the input into non-linear function of the linear combination of the various channels (interaction types), parsimoniously.

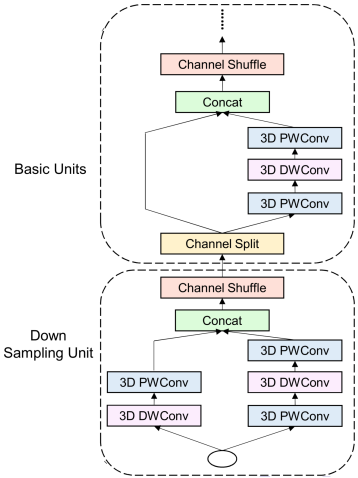


# ii. Stacked feature extraction block

**Three 3D shuffle groups are stacked, each containing a parsimonious combination of pointwise and depthwise convolutions.**
Shuffling, splitting, and then processing only certain channels encourages parsimonious while maintaining performance.

$$\mathrm{DWConv}(W, h)_{(i,j,k)} =$$

$$\Sigma_{s,t,r}^{S,T,R} W_{s,t,r} \odot h_{(i+s,j+t,k+r)}$$

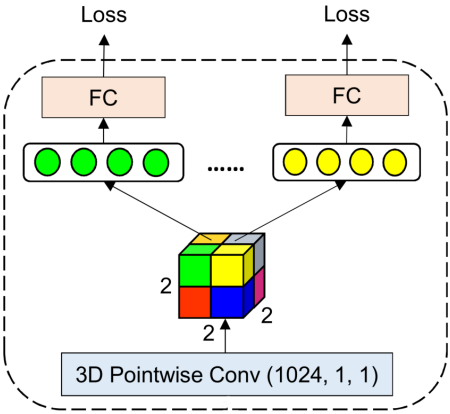


# iii. Classification Block

**Previously, an ensemble of eight fully-connected networks**, derived from the eight channels of the output of the second block, are used to create predictions.

Due to overfitting issues, I tried a few simpler approaches, settling on this:

1. A 3D pointwise convolution.
2. Average pooling.
3. Flattening into a 976-vector to be passed through a single 976-to-1 fully-connected layer.




*Previous approach*

# Data: a simplified attempt

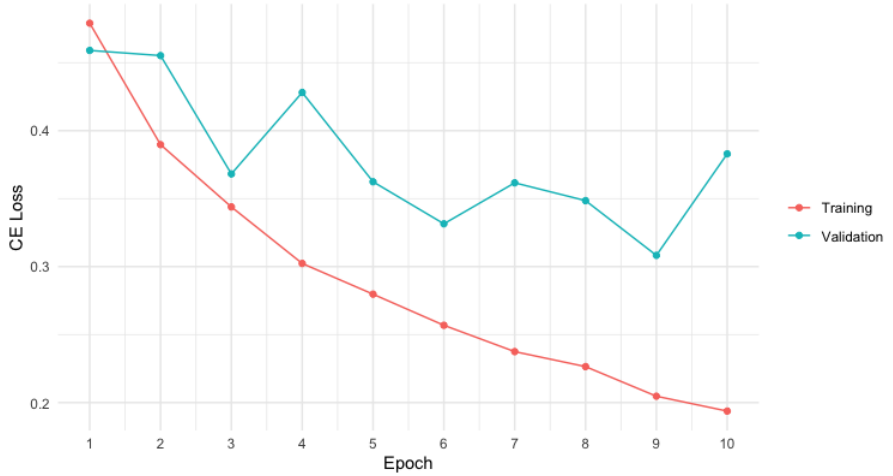
**For this initial test, the model had it easy.**

- *A limited number of proteins:* the model was trained and validated on examples from the *same* 10 proteins.
- *A high actives ratio:* the ratio of active ligands to decoys was 1:3, far greater than what occurs naturally.
- No lack of data: the training set had $\sim 13,000$ cocomplexes in total: about double than when this architecture was used for BAP originally.

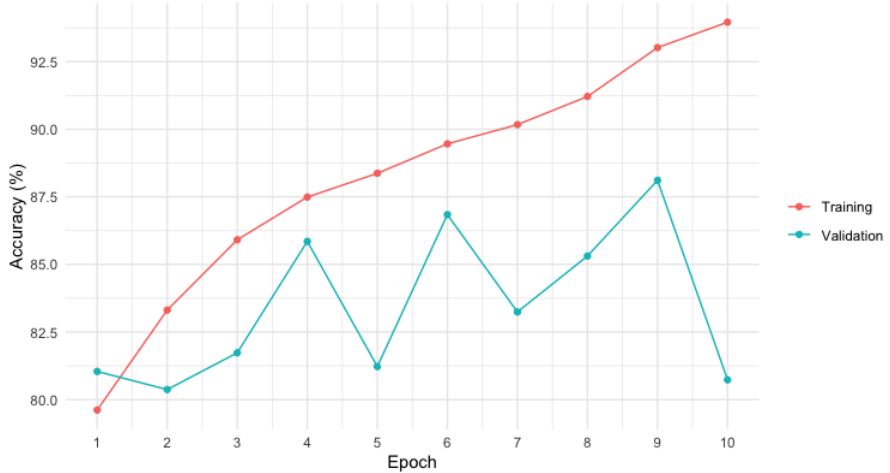
**However,**

- The rasterized 3D cocomplex grid was "channelized" into 8 rather than 24 channels, based on the `MoleculeKit` rather than the `Arpeggio` package.

# Loss curves



# Accuracy curves



# Validation confusion matrix and associated statistics

| Prediction \ Truth | (+) | (−) |
|---|---|---|
| (+) | TPR = 439 | FPR = 247 |
| (−) | FNR = 184 | TNR = 1847 |

# Evaluation

**The model has middling performance, and overfits very quickly.**

- The shuffle groups, while designed to save parameters without increasing the model's bias, are likely too many in number.
- The reduced dimensionality of the input due to the fewer channels likely did not help.
- The model does not cope well with class imbalance.

**Therefore, some next steps:**

- Use `Arpeggio`'s channelization.
- Use more data, and weight positive data more highly.
- Tune discrete aspects of the network architecture (e.g.: number of layers).
- Enable better data processing and optimization using NSCC.

# For More Details...

DeepAtom:

Li, Y., Rezaei, M. A., Li, C., & Li, X. (2019)

DeepAtom: A framework FOR Protein-Ligand binding affinity prediction

*IEEE/ACM Transactions on Computational Biology and Bioinformatics.*