Journal Report 16

1/29/20-2/7/20

Sohom Paul
Computer Systems Research Lab
Period 1, White

Daily Log

Wednesday, January 29

Started Tensorboard tutorial to visualize losses.
Encountered some errors with localhost not connected.

Thursday, January 30

Took AMC 12A.

Saturday, February 1

Started comparing test losses between multitask and single task learned networks.
Redesigned task clustering structure.

Monday, February 3

Discovered the DeepChem library and started reading documentation.
Installed requisite packages.

Saturday, February 8

Started building first message-passing neural network (MPNN) from tutorial.
Encountered parsing errors when loading dataset.

Timeline

Date	Goal	Met
Jan 27	Further investigate deep relationship networks.	Built toy model; abandoned integration.
Feb 3	Analyze benefits provided by multitask learning and begin hyperparameter tuning.	Compared evalutation losses; did not start hyperparameter search.
Feb 10	Identify and start implementing alternative neural methods.	Yes; installed DeepChem to investigate MPNNs.
Feb 17	Understand code from MPNN tutorial. Start building neural fingerprinting.	
Feb 24	Continue building neural fingerprinting.	

Reflection

I had set my initial goals for the week of 1/29-1/31 without realizing that it would be a short week with the AMC 12A during the only full-length class. Because of this, I focused on testing whether my task clustering did in fact improve upon the results of single task learning (which I really ought to have done before the winter demonstration). This testing confirmed what I had suspected and hoped to be true, that even with equivalent hyperparameters (same number of epochs, learning rate, and size of training set), the losses for the multitask networks were substantially loewr than that of the single task network in almost all cases. The results are shown in the table below. However, we do have some notable examples: the second and third rotational constants (B and C) are really horribly predicted by the multitask network, and mu and homo are predicted slightly worse than for the single task network. For this reason, when my partner and I are ready to train our final networks before tjSTAR, I intend to adjust my cluster structure to separate out B and C. Whether I remove mu and homo as well will require further testing, considering that removing them from the clusters might adversely affect the learning of the other tasks.

Task	Single Task Loss	Multitask Loss	Percent Reduction
A	2.84445	2.45822	-13.58%
B	0.21984	2.04991	832.45%
C	0.05537	0.80964	1362.31%
mu	0.80060	0.91109	13.80%
alpha	3.96902	0.28879	-92.72%
homo	0.51020	0.65306	28.00%
lumo	0.36548	0.13411	-63.31%
r2	2.76059	0.53923	-80.47%
zpve	1.37364	0.11786	-91.42%
u0	3.51353	0.26782	-92.38%
u298	3.61184	0.13014	-96.40%
h298	4.15636	0.00193	-99.95%
g298	4.50168	0.04913	-98.91%
cv	2.37409	0.00456	-99.81%

I have reported in my most recent journal that I intended to start engaging in hyperparameter search, arguing that because I would need several rounds of training, it would be better to start it early. However, after further discussion with my partner, I'm holding off doing this hyperparameter search until we have finalized our architecture, considering that further adjustments she will be making to the structure of the convolutional layers of our network will possibly change what hyperparameters we ought to be using. That being said, I did end up spending some time trying to learn how to use the Tensorboard visualization tools. I encountered some problems when I tried running Tensorboard using the IPython line magic. Tensorboard was reportedly running on a process that I couldn't locate with `htop`, and the IPython window was telling me that they couldn't connect to localhost. This is a problem that I'm kicking to down the road when loss visualization becomes more important to us.

For the week of 2/3-2/7, I rediscovered the DeepChem library. My partner and I had originally considered building our project based on this library instead of Spektral, but chose Spektral intead because of some of the more advanced networks that came pre-implemented. However, I have just found out that some users have contributed code for message-passing neural networks to DeepChem. Understanding the code, however, will require me to gain a better understanding of PyTorch. Also, the dataset that the code makes reference to is stupidly not included in the GitHub repository, so I had to pull it from a different source, generating some parse errors that I haven't fully resolved yet.

Thinking to the future: our pared-down vision for our project, building *a* multitask graph convolutional network, has already been completed: the rest of the year is for refinement and improvements. However, one thing that has I was really hoping to achieve was to build a network capable of property prediction for more general sets of molecules. Specifically, I want to be able to let the user query the properties of a general molecule, not just one of the 133k molecules in our dataset. (Granted, once we leave those 133k molecules, then we have no good way of seeing how accurate our predictions are, which is why I've delayed starting this goal.) To this end, I've started looking at the work of Duvenaud et al., 2015[1] which uses neural fingerprinting to reduce the arbitrary input molecule to a fixed size feature vector, which then in turn can be passed through a neural network for property prediction. Also, some time in the future, I will build a simple user interface to let the user draw their own molecules.

[1] https://arxiv.org/pdf/1509.09292.pdf