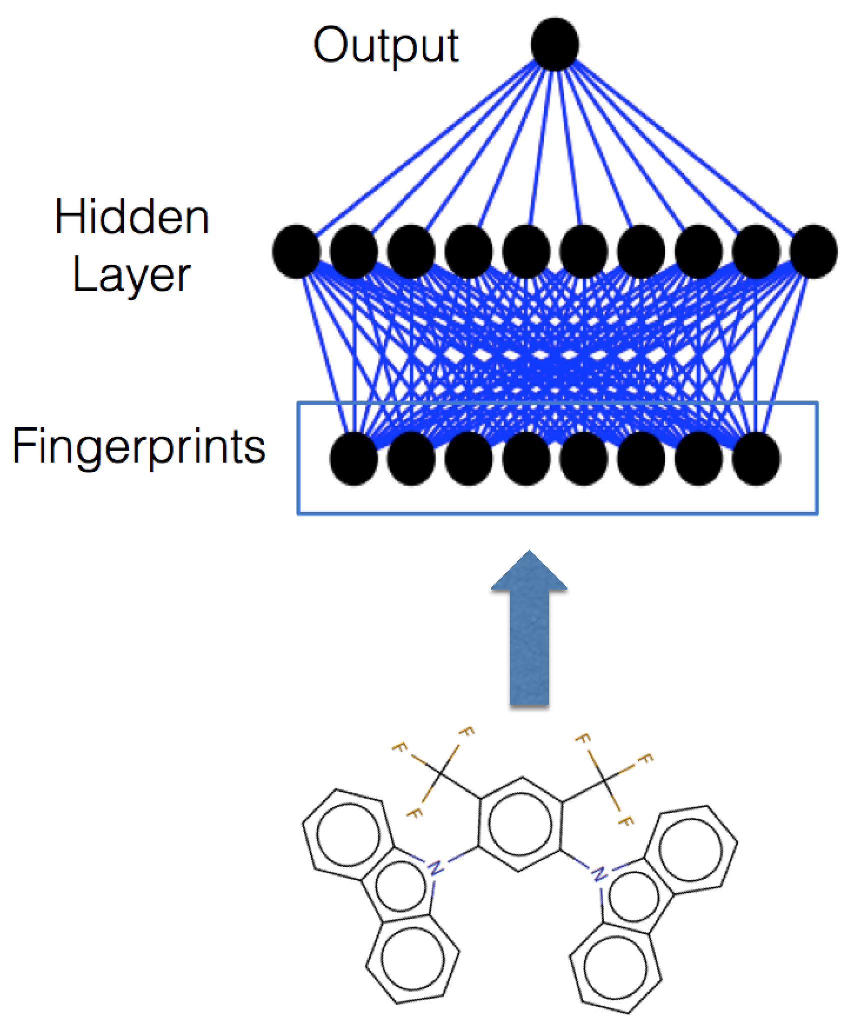
Convolutional Networks on Graphs for Learning Molecular Fingerprints

David Duvenaud*, Dougal Maclaurin*, Jorge Aguilera-Iparraguirre
Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, Ryan P. Adams

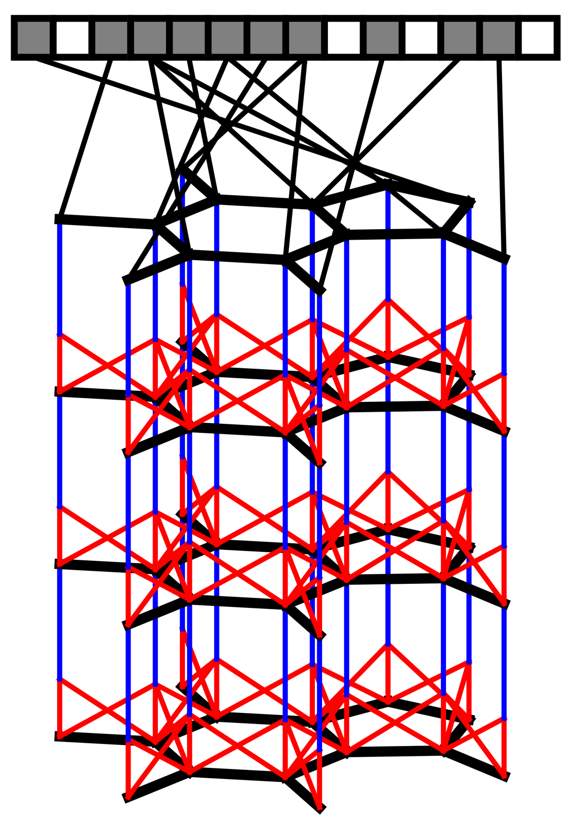
Problem

- How to regression on graphs?
- Input can be any size or shape
- Hard to turn into fixed-length vector
- In our case, graphs represent molecules
- Applications to photovoltaics, organic LEDS, flow batteries and pharmaceuticals



Circular fingerprints

- Maps variable-sized molecular graph to fixed-length binary vector
- Binary features indicate presence of substructures



Can be efficiently computed using local operations:

- At each layer, hash the features of each atom and its neighbors/bonds
- More layers correspond to increasing radius of substructures
- Interpret each hash as integer and set that entry to one

Currently state-of-the-art for large-scale regression and classification.

Convolutional neural nets on graphs

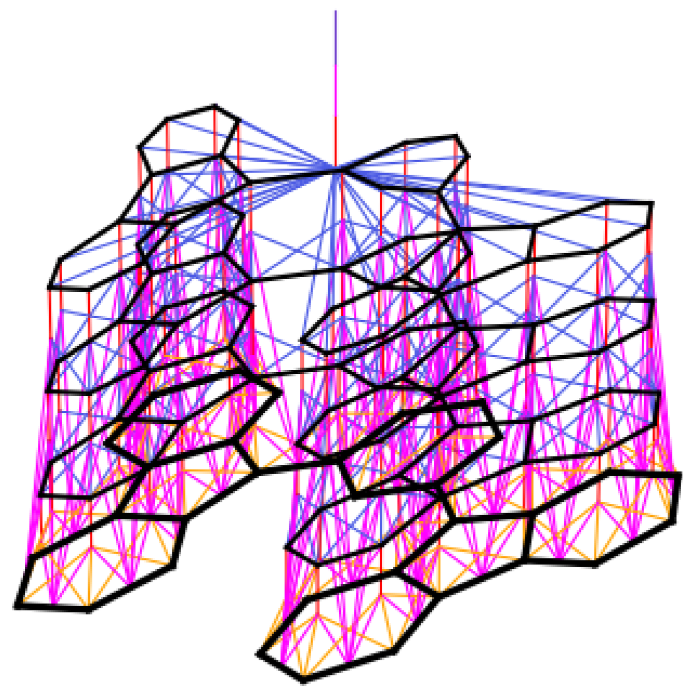
How to make graph fingerprints differentiable? Replacing ops:

Hash → Neural net
Index → Softmax
Write → Add

Gives end-to-end differentiable convolutional network.

Can be trained to adapt to particular tasks.

Information flow graph:

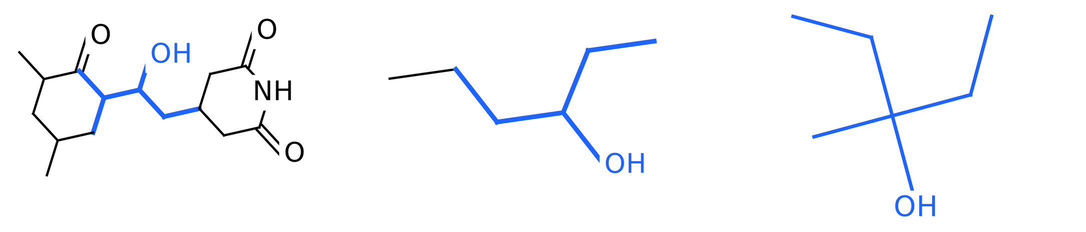


Message passing between neighbors, then final pooling step

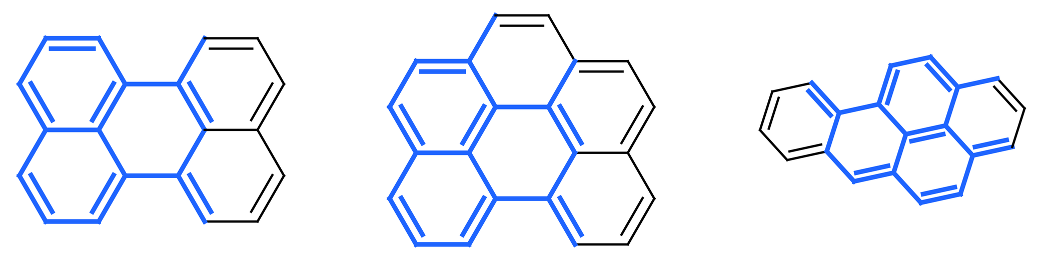
Neural fingerprints are interpretable

When fed into linear layer, can see how fragments affect prediction:

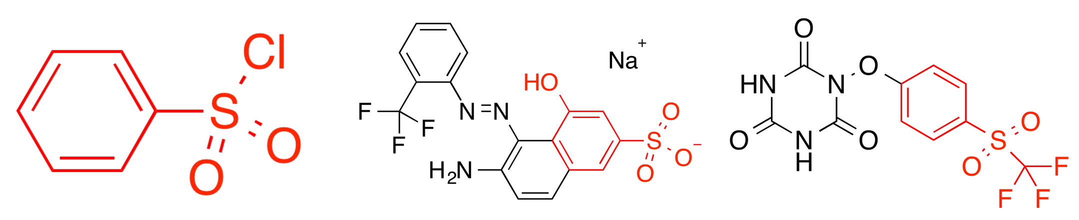
Fragments predictive of solubility



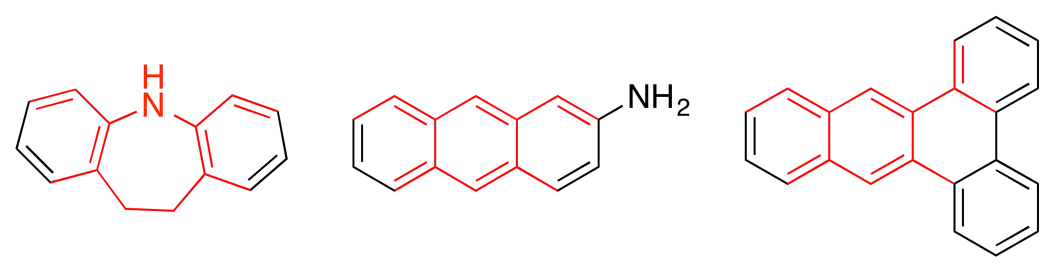
Fragments predictive of insolubility



Fragments predictive of toxicity on SR-MMP dataset

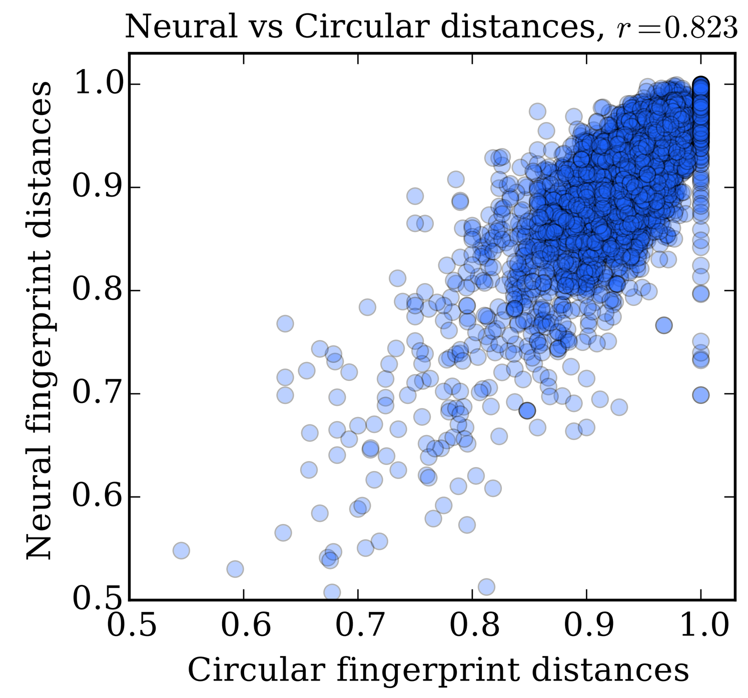


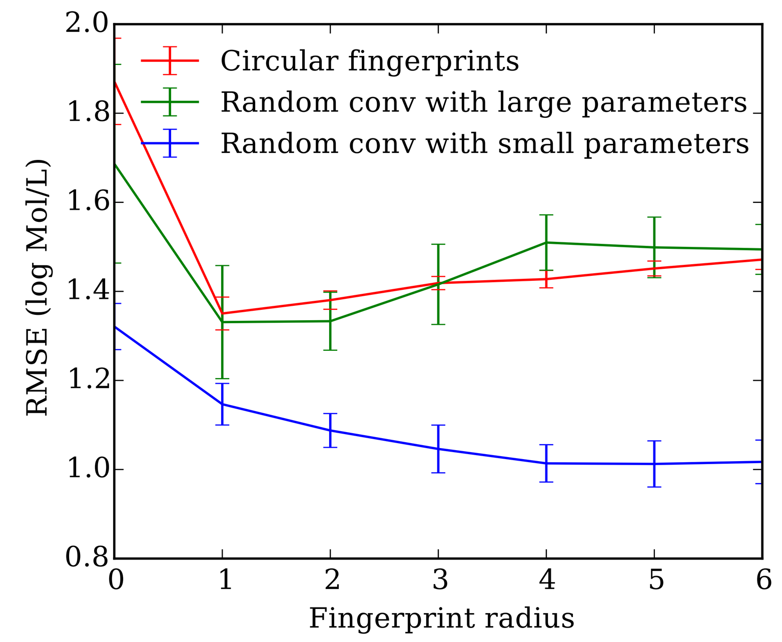
Fragments predictive of toxicity on NR-AHR dataset



Neural graph fingerprints generalize circular fingerprints

Large random weights give similar behavior to circular fingeprints:





Small random weights already much better than circular fingerprints!
Can do even better by optimizing for given task.

Predictive accuracy

Neural graph fingerprints fed to neural net generalizes state of the art:

Dataset	Solubility	Drug efficacy	Photovoltaic efficiency
Units	log Mol/L	EC_{50} in nM	percent
Predict mean	2.07 ± 0.10	1.21 ± 0.03	2.53 ± 0.02
Circular FPs + linear layer	1.31 ± 0.05	**1.06 ± 0.01**	1.62 ± 0.03
Circular FPs + neural net	1.18 ± 0.05	1.16 ± 0.04	1.41 ± 0.03
Neural FPs + linear layer	0.87 ± 0.06	**1.07 ± 0.01**	1.61 ± 0.06
Neural FPs + neural net	**0.72 ± 0.05**	**1.08 ± 0.01**	**1.20 ± 0.04**

Conclusion

- Can learn graph features end-to-end!
- Works on other types of graphs too
- Code at `github.com/HIPS/neural-fingerprint`
- Autodiff package that works on standard Numpy code: `github.com/HIPS/autograd`