FLATIRON INSTITUTE
Center for Computational Biology

PREDICTING PROTEIN FUNCTION USING GEOMETRIC DEEP LEARNING ON GRAPHS

VLADIMIR GLIGORIJEVIĆ
14 March 2019

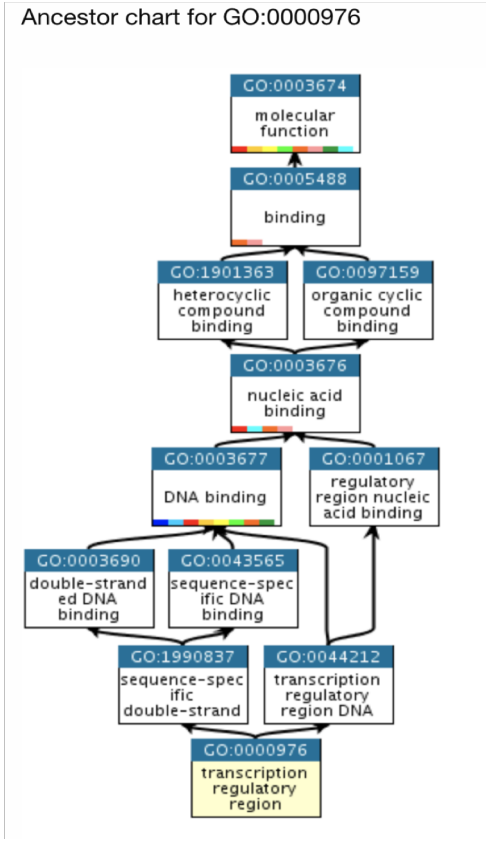
Outline

- Introducing the problem
- ML methods for function prediction:
- **Sequence-based methods**
  - Alignment-free methods: Convolutional Neural Networks (CNNs)
- **Structure-based methods**
  - Graph-CNNs for mapping structure-to-function
  - Graph-CNN variants
- **Network-based methods**
  - Deep network fusion (deepNF)
- Conclusions and future work

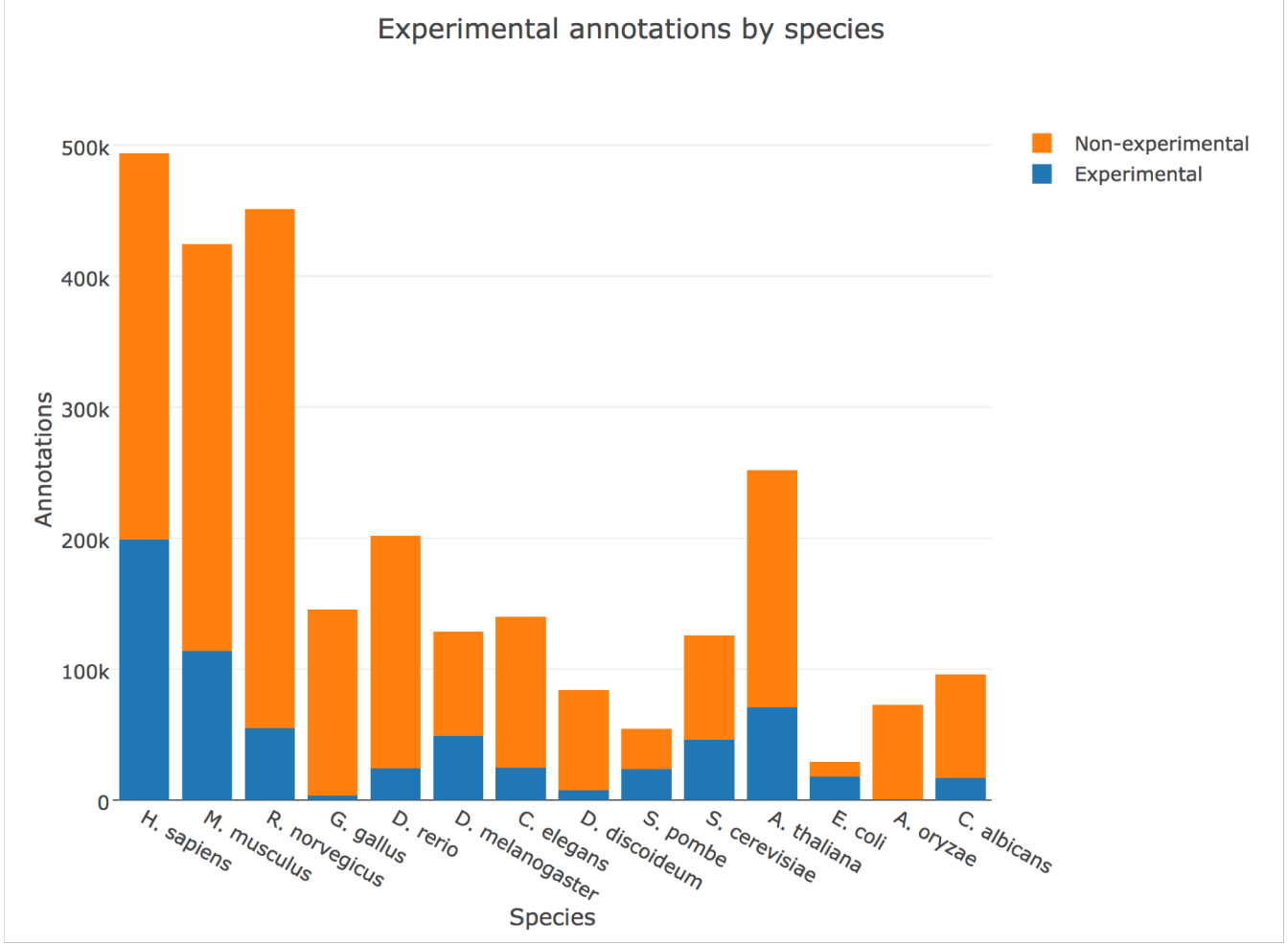
FLATIRON INSTITUTE
Center for Computational Biology

What do we mean by Function?

- Gene Ontology (GO):
  - hierarchical classification scheme for organizing and describing protein functions
- Consist of 3 independent DAGs:
  - **Molecular Function (MF)**
  - Biological Process (BP)
  - Cellular Component (CC)
- **Goal**: Narrow down the search space (of ~40,000 possible GO terms) for experimentalists



QuickGO: https://www.ebi.ac.uk/QuickGO



FLATIRON INSTITUTE
Center for Computational Biology

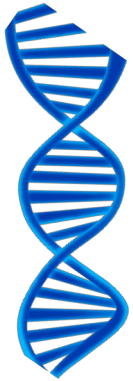
*Data from Gene Ontology (GO)

GO prediction: challenges

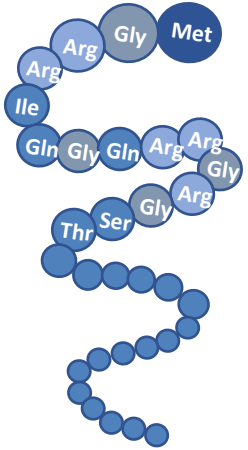
- Large numbers of yet to be discovered labels
- No real negative examples
- Class imbalance (implied by hierarchy)
- Cell context and need for integration of very diverse data types
- Evolutionary relationships make evaluation challenging
- Protein-level annotations

FLATIRON INSTITUTE
Center for Computational Biology

We can use different data for function prediction



DNA sequence

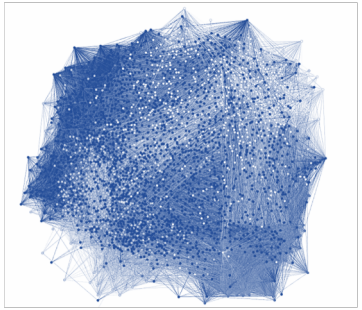


Protein sequence

Cheap and easy thanks to sequencing technology.

Protein structure

X-ray crystallography; NMR; Rosetta predictions.



Protein interaction networks

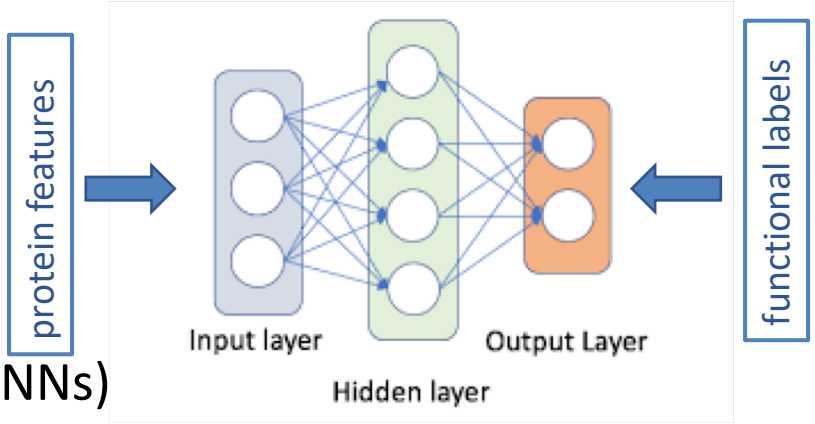
Y2H system, AP-MS experimental methods.

Deep learning is well-suited for function prediction

- Traditional methods
  - Sequence alignment methods (e.g., BLAST) → **fail on non-homologous sequences**
  - Machine learning methods → **predict one function at a time**

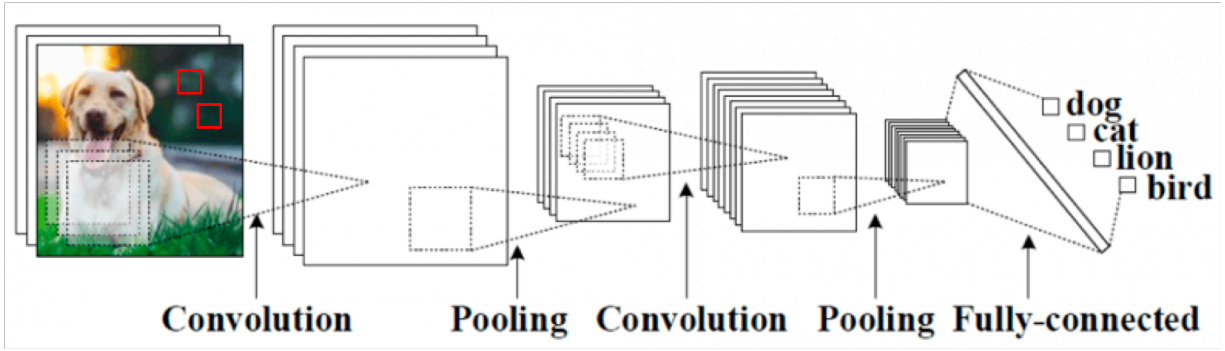
    (e.g., SVM, Logistic regression, etc.) → **require feature engineering**

- Benefits of using Neural Networks (NN)
  - multiple functions
  - high-level features
  - do not require feature engineering (e.g., CNNs)



CNN for sequence-to-function mapping

Convolution Neural Networks

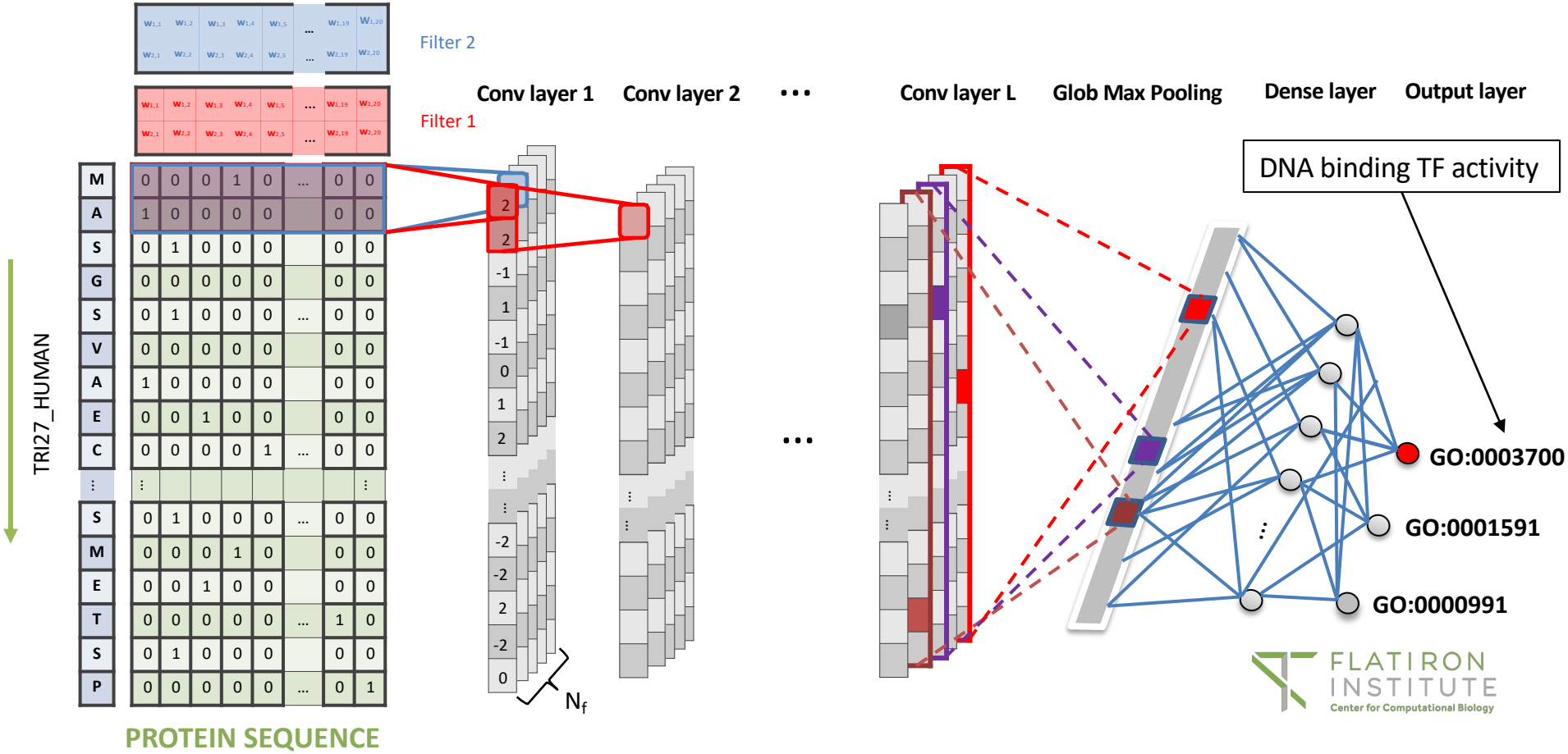


$$h_{ij}^k = \sigma((\mathbf{W}^k * x)_{ij} + \mathbf{b}_k)$$

where $\mathbf{W}^k$ are filters **learnable** through gradient descent

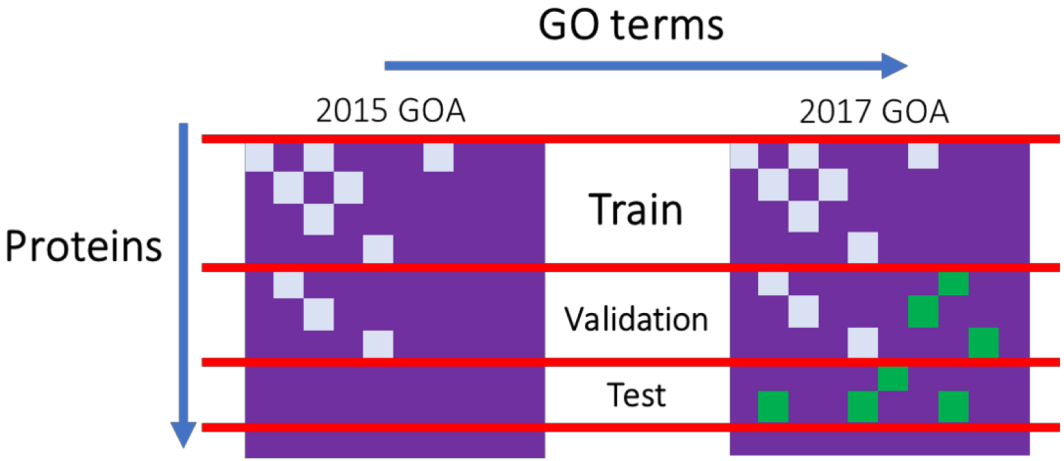
- Reduce number of parameters to be learned
- Localize and share weights (conv filters) → self-similarity
- Translation invariance
- Computationally efficient

FLATIRON INSTITUTE
Center for Computational Biology


Filter 2
Filter 1
Conv layer 1
Conv layer 2
Conv layer L
Glob Max Pooling
Dense layer
Output layer
DNA binding TF activity
GO:0003700
GO:0001591
GO:0000991
TRI27_HUMAN
PROTEIN SEQUENCE
N_f
FLATIRON INSTITUTE
Center for Computational Biology

We evaluate performance on new functional annotations over time

Temporal holdout (based on CAFA)

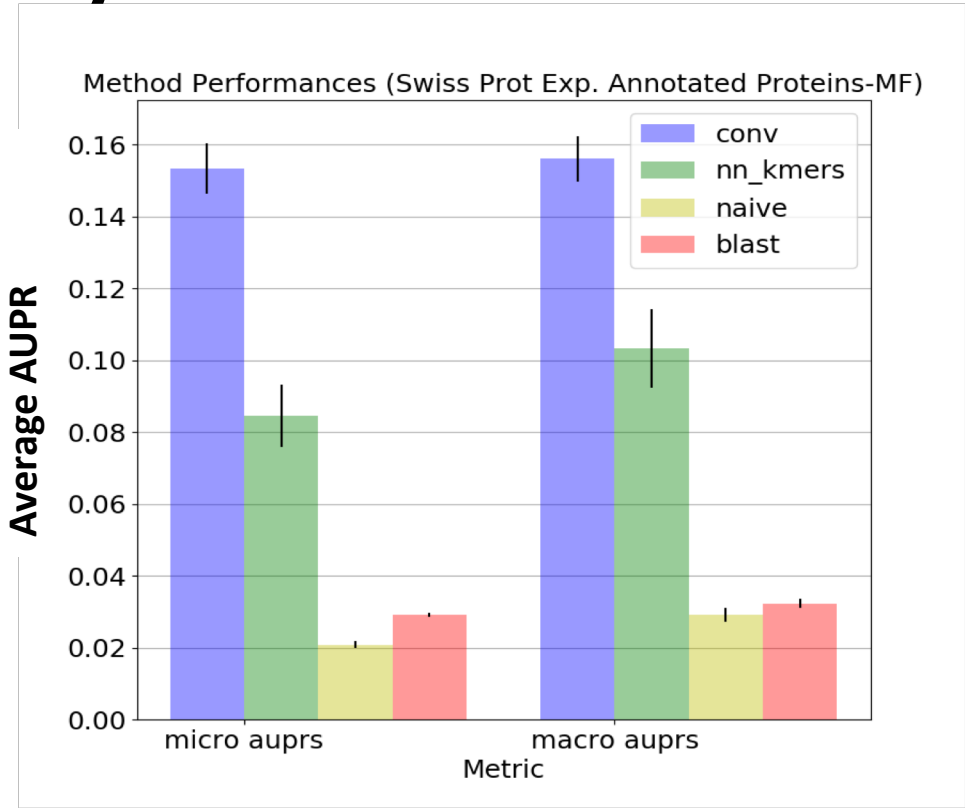


- Data: UniProt GOA

We only use experimentally validated annotation

FLATIRON INSTITUTE
Center for Computational Biology

CNN outperforms baselines on Swiss-Prot manually annotated and reviewed sequences



Meet Barot

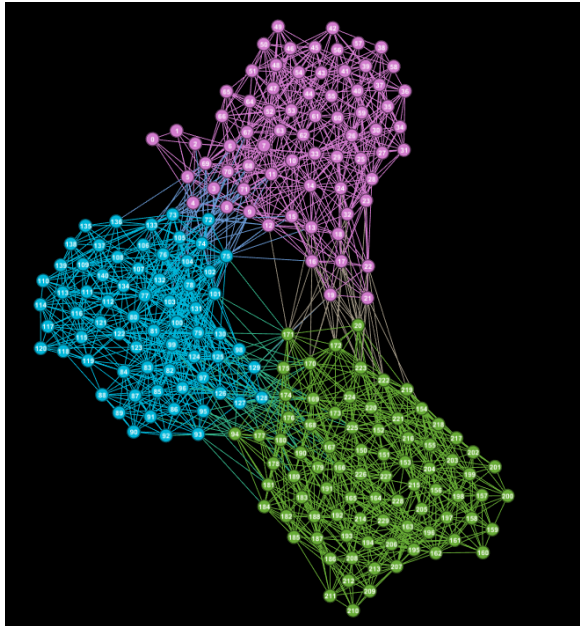
FLATIRON INSTITUTE
Center for Computational Biology

GCN for structure-to-function mapping

FLATIRON INSTITUTE
Center for Computational Biology

Example: 1U8R-B

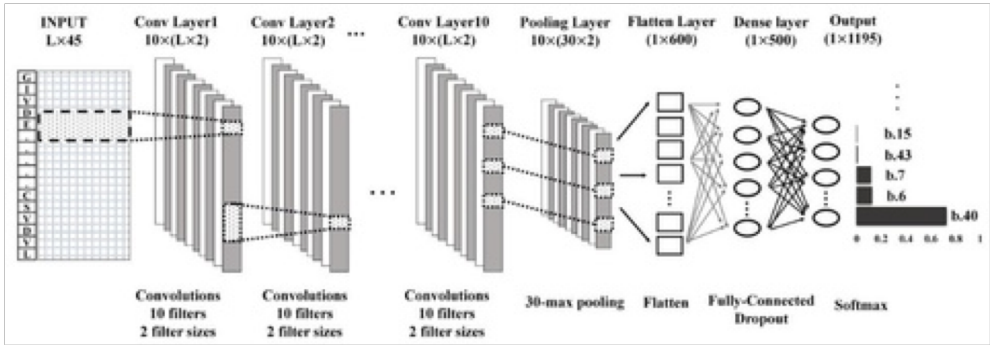
$C\alpha$-$C\alpha$
$< 10Å$



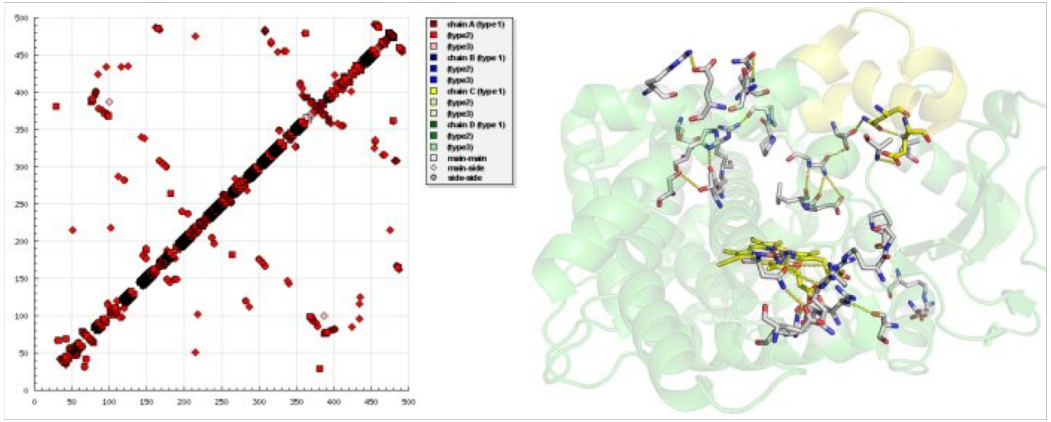
FLATIRON INSTITUTE
Center for Computational Biology

A new encoding scheme is needed for structural features

Regular CNN:



Protein contact map

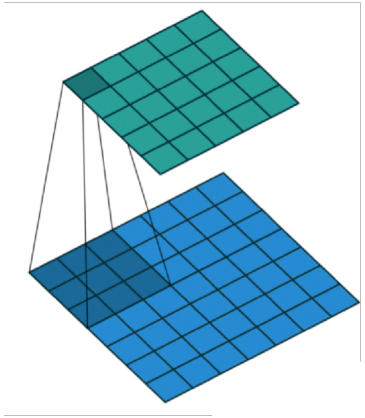


FLATIRON INSTITUTE
Center for Computational Biology

Hou, J. et al. "*DeepSF: deep convolutional neural network for mapping protein sequences to folds*." *Bioinformatics* (2017)

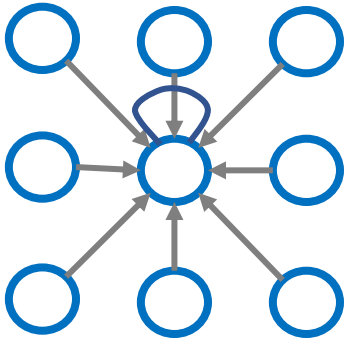
We use Graph Convolutions for processing contact maps

CNN



Image

Graph CNN



Graph

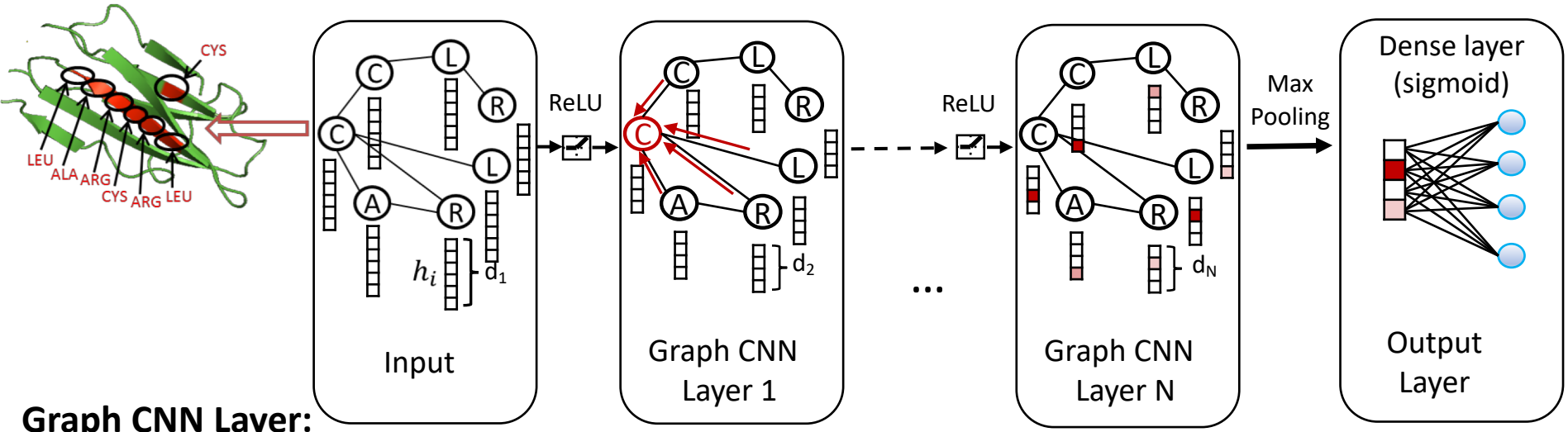
FLATIRON INSTITUTE
Center for Computational Biology

Graph Convolutional Networks

- <u>Types of GCNs:</u>
- Spectral formulation
- Spatial formulation

- <u>Applications:</u>
- Node classification
- Graph classification

An example of a Graph CNN...



Graph CNN Layer:

$$H^{(l+1)} = \sigma[\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}]$$

- Output to next layer / result
- 3. Nonlinearity
- 1. Normalize graph structure
- 2. Multiply node properties and weights

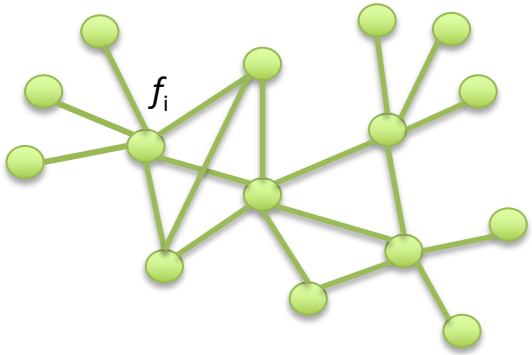
A – adjacency matrix representing contact map
D – diagonal degree matrix
$\mathbf{H}^{(0)}$ – input feature matrix ($\mathbf{H}^{(0)}=\mathbf{I}$, 1-hot encoding if features not available)
$\mathbf{W}^{(l)}$- weight matrix for layer l

FLATIRON INSTITUTE
Center for Computational Biology

Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." *arXiv preprint arXiv:1609.02907* (2016).

Brief intro to Graph Theory

- Undirected graph: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with **vertices** $\mathcal{V} = \{1, \ldots, n\}$ and **edges** $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$
- Represented by **adjacency matrix**, $\mathbf{A} \in \mathbb{R}^{n \times n}$, with $a_{ij} = 1$ if $(i, j) \in \mathcal{E}$ and $a_{ij} = 0$, otherwise
- Functions over graph vertices: $L^2(\mathcal{V}) = \{f : \mathcal{V} \to \mathbb{R}\}$ represented as vectors: $\mathbf{f} = (f_1, \ldots, f_n) \in \mathbb{R}^n$
- Inner product operation: $< \mathbf{f}, \mathbf{g} >_{L^2(V)} = \sum_{i \in \mathcal{V}} f_i g_i = \mathbf{f}^T \mathbf{g}$

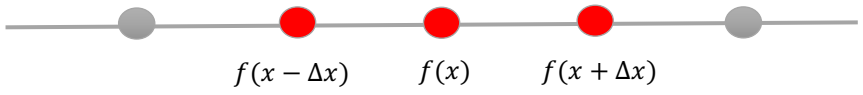


FLATIRON INSTITUTE
Center for Computational Biology

Laplace operator

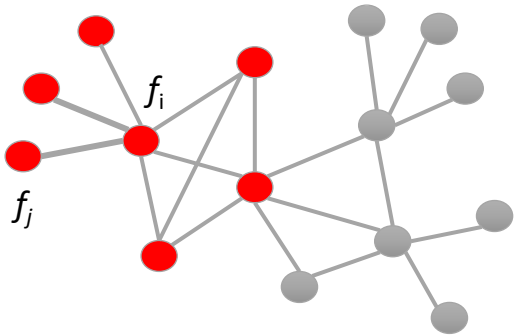
Euclidean

- $\Delta : L^2 \to L^2$
- In $\mathbb{R}$: $\Delta f = \lim_{\delta x \to 0} \frac{f(x+\delta x) - 2f(x) + f(x-\delta x)}{(\delta x)^2}$
- Discrete Laplace operator:
  $\Delta f \propto [f(x) - f(x+\Delta x)] + [f(x) - f(x-\Delta x)]$



Non-Euclidean

- $\Delta : L^2(\mathcal{V}) \to L^2(\mathcal{V})$
- On graphs: $(\Delta f)_i = \sum_{j:(i,j)\in\mathcal{E}} a_{ij}(f_i - f_j)$
- Matrix representation:
  $\mathbf{\Delta} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D} = diag(\sum_{j \neq i} a_{ij})$



Spectrum of the Laplacian

Euclidean

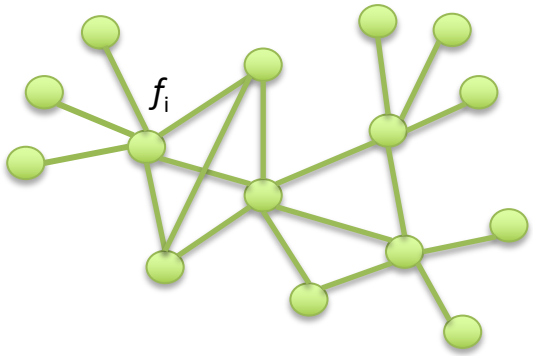
- Eigendecomposition: $-\Delta e^{2\pi i\xi x} = (2\pi\xi)^2 e^{-2\pi i\xi x}$
- Fourier transform: $\hat{f}(\xi) = \langle f, e^{2\pi i\xi x}\rangle = \int_{\mathbb{R}} f(x)e^{-2\pi i\xi x}dx$
- Inverse transform: $f(x) = \langle \hat{f}, e^{-2\pi i\xi x}\rangle = \int_{\mathbb{R}} \hat{f}(\xi)e^{2\pi i\xi x}d\xi$
- Fourier basis = Laplacian eigenfunctions

Non-Euclidean

- Normalized Laplacian: $\hat{\mathbf{\Delta}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{\Delta}\mathbf{D}^{-\frac{1}{2}} = \mathbf{I}_n - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$
- Eigendecomposition: $\hat{\mathbf{\Delta}}\boldsymbol{\phi}_i = \lambda_i\boldsymbol{\phi}_i$;
  - Non-negative **eigenvalues**: $0 \leq \lambda_1 \leq \lambda_2 \cdots \leq \lambda_n \leq 2$; Matrix form: $\mathbf{\Lambda} = diag(\lambda_1, \ldots, \lambda_n)$
  - Corresponding orthogonal **eigenvectors**: $\langle \boldsymbol{\phi}_i, \boldsymbol{\phi}_j\rangle = \delta_{ij}$; Matrix form: $\mathbf{\Phi} = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_n)$
- Eigendecomposition in matrix form: $\hat{\mathbf{\Delta}} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^T$
- Fourier transform: $\hat{f}(\lambda_l) = \langle f, \phi_l\rangle = \sum_{i=1}^n f_i\phi_{li}^*$; Matrix-vector form: $\hat{\mathbf{f}} = \mathbf{\Phi}^T\mathbf{f}$
- Inverse transform: $f_i = \langle f, \phi_l^*\rangle = \sum_{l=1}^n \hat{f}(\lambda_l)\phi_{li}$; Matrix-vector form: $\mathbf{f} = \mathbf{\Phi}\hat{\mathbf{f}}$

Convolution operation

- Classically, for two signals $f, g \in L^2(\mathbb{R})$ the convolution is defined as: $f * g(x) = \int_{\mathbb{R}} f(u)g(x-u)du$
- No clear analogue for graphs (in spatial domain) $\implies$ compute convolution in the Fourier domain
- Convolution theorem: $\widehat{(f * g)}(\xi) = \hat{f}(\xi) \circ \hat{g}(\xi) \implies$ take inverse
- Convolution of signals on a graph (in matrix-vector notation): $\mathbf{f} * \mathbf{g} = \mathbf{\Phi}(\mathbf{\Phi}^T\mathbf{g}) \circ (\mathbf{\Phi}^T\mathbf{f}) = \mathbf{\Phi}\,\mathrm{diag}(\hat{g}_1, \ldots, \hat{g}_n)\mathbf{\Phi}^T\mathbf{f}$



Filter coefficients depend on Laplacian eigenvectors!

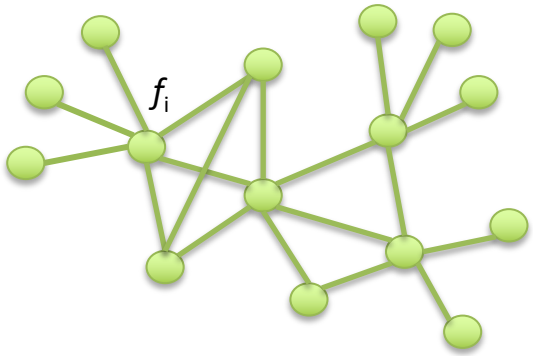
FLATIRON INSTITUTE
Center for Computational Biology

Spectral graph CNN

Convolution in spectral domain:

$$\mathbf{h} = \mathbf{\Phi}\mathbf{W}\mathbf{\Phi}^T\mathbf{f}$$

where $\mathbf{W}$ is n x n diagonal (learnable) matrix of spectral filter coefficients

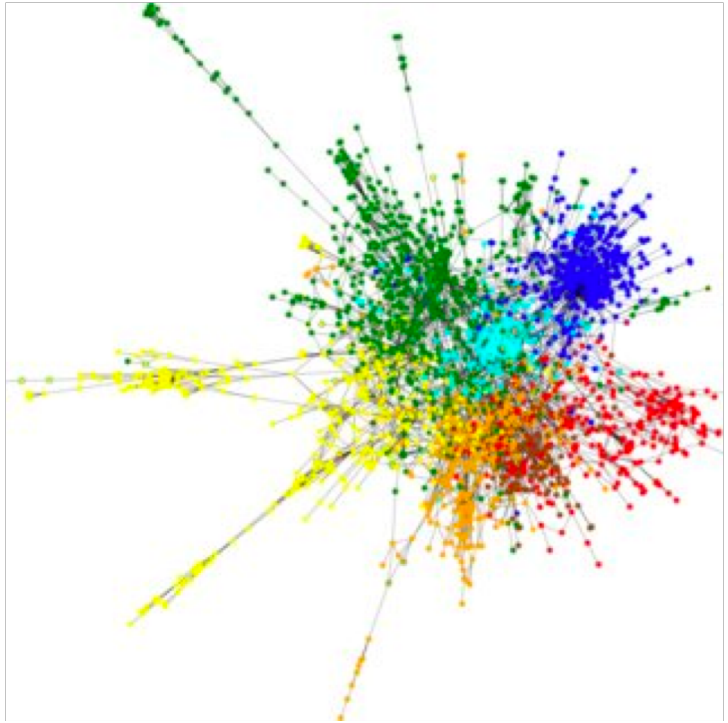


- **Filters are basis-dependent → does not generalize across graphs** 🙁
- **Learning complexity O(n)** 🙁
- **Computational complexity O(n^2)** 🙁
- **Filters are not localized in space** 🙁

FLATIRON INSTITUTE
Center for Computational Biology

Henaff, M., Bruna, J., & LeCun, Y. (2015). Deep convolutional networks on graph-structured data. *arXiv:1506.05163*.

Example: CoRA citation network



FLATIRON INSTITUTE
Center for Computational Biology

Bronstein MM, Bruna J, LeCun Y, Szlam A, Vandergheynst P. (2017) Geometric deep learning: going beyond euclidean data. IEEE Signal Processing Magazine.

Chebyshev CNN

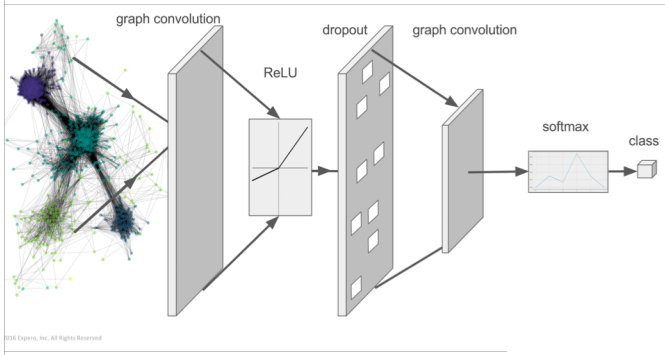
- Parametrize the filter using a smooth spectral function $g(\lambda)$
- Application of the filter: $\mathbf{h} = g(\hat{\mathbf{\Delta}})f = \mathbf{\Phi} g(\mathbf{\Lambda})\mathbf{\Phi}^T\mathbf{f}$
- Spectral filters represented by K^{th} order polynomials, $g_w(\mathbf{\Lambda}) = \sum_{k=0}^{K} w_k \mathbf{\Lambda}^k$, are **exactly K-localized**, where $\mathbf{w} = (w_1, \ldots, w_K)$ is the vector of filter parameters
- Number of parameters K (fixed) is independent of graph size n (same learning complexity as regular CNNs)
- No explicit computation of $\mathbf{\Phi}$, $\mathbf{\Phi}^T$ because $\mathbf{h} = g_w(\hat{\mathbf{\Delta}})\mathbf{f}$ $\implies O(K|\mathcal{E}|)$ computational complexity

FLATIRON INSTITUTE
Center for Computational Biology

Defferrad M, Bresson, X, Vandergheynst P. (2016) Convolutional neural networks on graphs with fast localized spectral filtering, NIPS

Graph Convolutional Networks

- First-order approximation ($K = 1$) of spectral graph convolution
- $\mathbf{h} = g_w(\hat{\mathbf{\Delta}})\mathbf{x} = \sum_{k=0}^{1} w_k \hat{\mathbf{\Delta}}^k \mathbf{x} = w(\mathbf{I}_n + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{x}$, for signal $\mathbf{x} \in \mathbb{R}^n$
- Renormalization trick (numerical stability): $\mathbf{I}_n + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} \rightarrow \hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}$, with $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ and $\hat{\mathbf{D}}_{ii} = \sum_j \hat{\mathbf{A}}_{ij}$
- Generalization to signals with C input channels, $\mathbf{X} \in \mathbb{R}^{N \times C}$, and F filters: $\mathbf{H} = \hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}\mathbf{W}$ where $\mathbf{W} \in \mathbb{R}^{C \times F}$ is a matrix of filter parameters, and $\mathbf{H} \in \mathbb{R}^{N \times F}$ is the output activation matrix



FLATIRON INSTITUTE
Center for Computational Biology

Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." *arXiv preprint arXiv:1609.02907* (2016).

GCN variants

- $a_v^{(k)} = AGGREGATE^{(k)}\left(\left\{h_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right)$
- $h_v^{(k)} = COMBINE^{(k)}\left(h_v^{(k-1)}, a_v^{(k)}\right)$
- $h_G = READOUT\left(\left\{h_v^{(K)} | v \in G\right\}\right)$
- In GCN (Kipf & Welling, 2017):
  $a_v^{(k)} = MEAN\left(\left\{ReLU(\mathbf{W}h_u^{(k-1)}), \forall u \in \mathcal{N}(v)\right\}\right)$

FLATIRON INSTITUTE
Center for Computational Biology

The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

An End-to-End Deep Learning Architecture for Graph Classification

Muhan Zhang, Zhicheng Cui, Marion Neumann, Yixin Chen
Department of Computer Science and Engineering, Washington University in St. Louis
{muhan, z.cui, m.neumann}@wustl.edu, chen@cse.wustl.edu

Abstract

Neural networks are typically designed to deal with data in tensor forms. In this paper, we propose a novel neural network architecture accepting graphs of arbitrary structure. Given a dataset containing graphs in the form of (G, y) where G is a graph and y is its class, we aim to develop neural networks that read the graphs directly and learn a classification function. There are two main challenges: 1) how to extract useful features characterizing the rich information encoded in a graph for classification purpose, and 2) how to sequentially read a graph in a meaningful and consistent order. To address the first challenge, we design a localized graph convolution model and show its connection with two graph kernels. To address the second challenge, we design a novel SortPooling layer which sorts graph vertices in a consistent order so that traditional neural networks can be trained on the graphs. Experiments on benchmark graph classification datasets demonstrate that the proposed architecture achieves highly competitive performance with state-of-the-art graph kernels and other graph neural network methods. Moreover, the architecture allows end-to-end gradient-based training with original graphs, without the need to first transform graphs into vectors.

1 Introduction

The past few years have seen the growing prevalence of neural networks on application domains such as image classification (Alex, Sutskever, and Hinton 2012), natural language processing (Mikolov et al. 2013), reinforcement learning (Mnih et al. 2013), and time series analysis (Cui, Chen, and Chen 2016). The connection structure between layers makes neural networks suitable for processing signals in tensor forms where the tensor elements are arranged in a meaningful order. This fixed input order is a cornerstone for neural networks to extract higher-level features. For example, if we randomly shuffle the pixels of an image shown in Figure 1, then state-of-the-art convolutional neural networks (CNN) fail to recognize it as an eagle.

Although images and many other types of data are naturally presented with order, there is another major category of structured data, namely graphs, which usually lack a tensor representation with fixed ordering. Examples include molecular structures, knowledge graphs, biological networks, social

Copyright © 2018, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

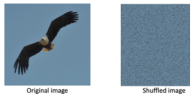


Figure 1: A consistent input ordering is crucial

networks, and text documents with dependenci
of ordered tensor representations limits the app
neural networks on graphs. In this paper, we aim
novel neural network structures that can accept
learn predictive functions.

Recently, there is a growing interest in general
networks to graphs. (Bruna et al. 2013) genera
lutional networks to graphs in the spectral dor
filters are applied on a graph's frequency mode
by graph Fourier transform. This transformati
expensive multiplications with the eigenvector r
graph Laplacian. To reduce the computation b
ferrard, Bresson, and Vandergheynst 2016) pa
the spectral filters as Chebyshev polynomials of
and achieved efficient and localized filters. One
the above spectral formulations is that they rely
spectrum of the graph Laplacian, and thus are s
for graphs with a single structure (and varying si
tices). Spatial formulations, on the contrary, are r
to a fixed graph structure. To extract local featu
works independently proposed to propagate featu
neighboring vertices. (Duvenaud et al. 2015) pro
entiable Neural Graph Fingerprints, which propa
between 1-hop neighbors to simulate the traditi
fingerprint. (Atwood and Towsley 2016) propose
CNN, which propagates neighbors with differen
center using different weights. Later, (Kipf and W
developed a first-order approximation of the spec
tion in (Defferrard, Bresson, and Vandergheynst
also resulted in propagation between neighbor
(Niepert, Ahmed, and Kutzkov 2016) proposed
of spatial graph convolution by extracting fixed
patches from nodes' neighborhoods and linea

Published as a conference paper at ICLR 2019

How Powerful are Graph Neural Networks?

Keyulu Xu *†
MIT
keyulu@mit.edu

Weihua Hu *‡
Stanford University
weihuahu@stanford.edu

Jure Leskovec
Stanford University
jure@cs.stanford.edu

Stefanie Jegelka
MIT
stefje@mit.edu

Abstract

Graph Neural Networks (GNNs) are an effective framework for representation learning of graphs. GNNs follow a neighborhood aggregation scheme, where the representation vector of a node is computed by recursively aggregating and transforming representation vectors of its neighboring nodes. Many GNN variants have been proposed and have achieved state-of-the-art results on both node and graph classification tasks. However, despite GNNs revolutionizing graph representation learning, there is limited understanding of their representational properties and limitations. Here, we present a theoretical framework for analyzing the expressive power of GNNs to capture different graph structures. Our results characterize the discriminative power of popular GNN variants, such as Graph Convolutional Networks and GraphSAGE, and show that they cannot learn to distinguish certain simple graph structures. We then develop a simple architecture that is provably the most expressive among the class of GNNs and is as powerful as the Weisfeiler-Lehman graph isomorphism test. We empirically validate our theoretical findings on a number of graph classification benchmarks, and demonstrate that our model achieves state-of-the-art performance.

1 Introduction

Learning with graph structured data, such as molecules, social, biological, and financial networks, requires effective representation of their graph structure (Hamilton et al., 2017b). Recently, there has been a surge of interest in Graph Neural Network (GNN) approaches for representation learning of graphs (Li et al., 2016; Hamilton et al., 2017a; Kipf & Welling, 2017; Velickovic et al., 2018; Xu et al., 2018). GNNs broadly follow a recursive neighborhood aggregation (or message passing) scheme, where each node aggregates feature vectors of its neighbors to compute its new feature vector (Xu et al., 2018; Gilmer et al., 2017). After k iterations of aggregation, a node is represented by its transformed feature vector, which captures the structural information within the node's k-hop neighborhood. The representation of an entire graph can then be obtained through pooling (Ying et al., 2018), for example, by summing the representation vectors of all nodes in the graph.

Many GNN variants with different neighborhood aggregation and graph-level pooling schemes have been proposed (Scarselli et al., 2009b; Battaglia et al., 2016; Defferrard et al., 2016; Duvenaud et al., 2015; Hamilton et al., 2017a; Kearnes et al., 2016; Kipf & Welling, 2017; Li et al., 2016; Velickovic et al., 2018; Santoro et al., 2017; Xu et al., 2018; Santoro et al., 2018; Verma & Zhang, 2018; Ying et al., 2018; Zhang et al., 2018). Empirically, these GNNs have achieved state-of-the-art performance in many tasks such as node classification, link prediction, and graph classification. However, the design of new GNNs is mostly based on empirical intuition, heuristics, and experimental trial-and-error. There is little theoretical understanding of the properties and limitations of GNNs, and formal analysis of GNNs' representational capacity is limited.

Neural Message Passing for Quantum Chemistry

Justin Gilmer [1] **Samuel S. Schoenholz** [1] **Patrick F. Riley** [2] **Oriol Vinyals** [3] **George E. Dahl** [1]

Abstract

Supervised learning on molecules has incredible potential to be useful in chemistry, drug discovery, and materials science. Luckily, several promising and closely related neural network models invariant to molecular symmetries have already been described in the literature. These models learn a message passing algorithm and aggregation procedure to compute a function of their entire input graph. At this point, the next step is to find a particularly effective variant of this general approach and apply it to chemical prediction benchmarks until we either solve them or reach the limits of the approach. In this paper, we reformulate existing models into a single common framework we call Message Passing Neural Networks (MPNNs) and explore additional novel variations within this framework. Using MPNNs we demonstrate state of the art results on an important molecular property prediction benchmark; these results are strong enough that we believe future work should focus on datasets with larger molecules or more accurate ground truth labels.

1. Introduction

The past decade has seen remarkable success in the use of deep neural networks to understand and translate natural language (Wu et al., 2016), generate and decode complex audio signals (Hinton et al., 2012), and infer features from real-world images and videos (Krizhevsky et al., 2012). Although chemists have applied machine learning to many problems over the years, predicting the properties of molecules and materials using machine learning (and especially deep learning) is still in its infancy. To date, most research applying machine learning to chemistry tasks (Hansen et al., 2015; Huang & von Lilienfeld, 2016;

[1]Google Brain [2]Google [3]Google DeepMind. Correspondence to: Justin Gilmer <gilmer@google.com>, George E. Dahl <gdahl@google.com>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

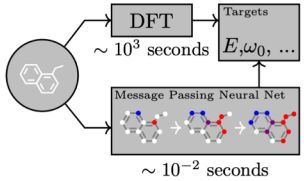


Figure 1. A Message Passing Neural Network predicts quantum properties of an organic molecule by modeling a computationally expensive DFT calculation.

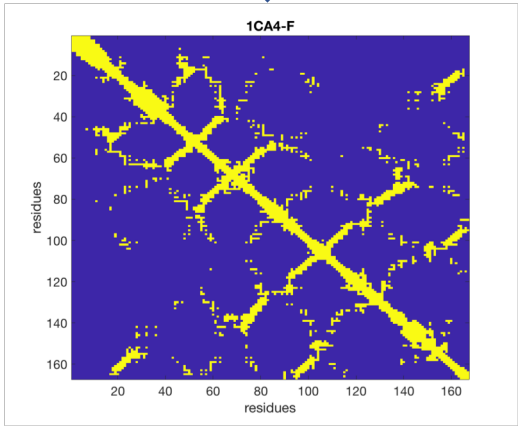
Rupp et al., 2012; Rogers & Hahn, 2010; Montavon et al., 2012; Behler & Parrinello, 2007; Schoenholz et al., 2016) has revolved around feature engineering. While neural networks have been applied in a variety of situations (Merkwirth & Lengauer, 2005; Micheli, 2009; Lusci et al., 2013; Duvenaud et al., 2015), they have yet to become widely adopted. This situation is reminiscent of the state of image models before the broad adoption of convolutional neural networks and is due, in part, to a dearth of empirical evidence that neural architectures with the appropriate inductive bias can be successful in this domain.

Recently, large scale quantum chemistry calculation and molecular dynamics simulations coupled with advances in high throughput experiments have begun to generate data at an unprecedented rate. Most classical techniques do not make effective use of the larger amounts of data that are now available. The time is ripe to apply more powerful and flexible machine learning methods to these problems, assuming we can find models with suitable inductive biases. The symmetries of atomic systems suggest neural networks that operate on graph structured data and are invariant to graph isomorphism might also be appropriate for molecules. Sufficiently successful models could someday help automate challenging chemical search problems in drug discovery or materials science.

In this paper, our goal is to demonstrate effective machine learning models for chemical prediction problems

For benchmarking we use the PDB

< 6.5Å



Experimental PDB structures:

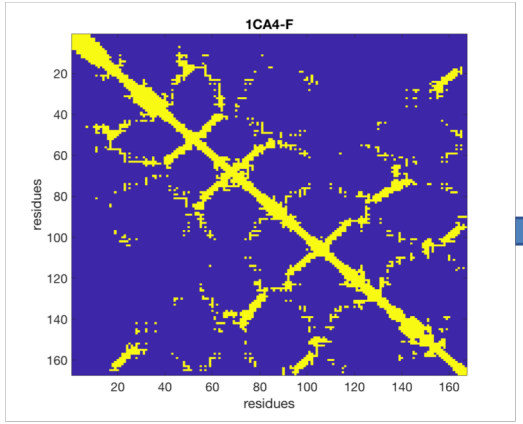
annotations	Chains	PDB IDs
Experimental annotations	107451	40478
Inferred annotations	179852	76675

-PDB chains with L >= 40 residues (40 <= L <= 500 are padded with zeros)
-PDB chains with GO terms (InterPro2GO) with evidence codes: EXP, IDA, IPI, IMP, IGI, IEP, TAS and IC
-Distance between all atoms (d < 6.5Å)
-Experimental SEQRES sequences for training CNN

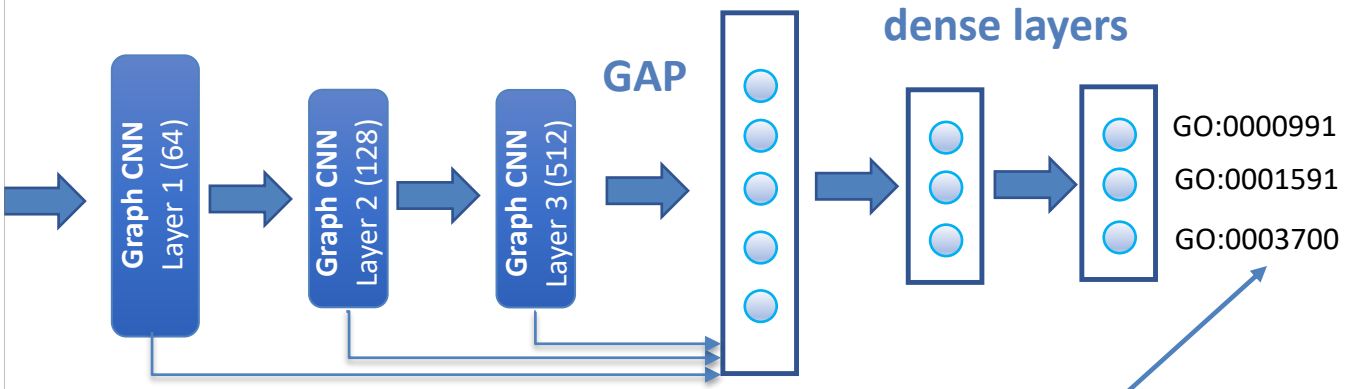
Train/Test (BLASTClust):

Graph-CNN: implementation details...

Our model (schematic representation):



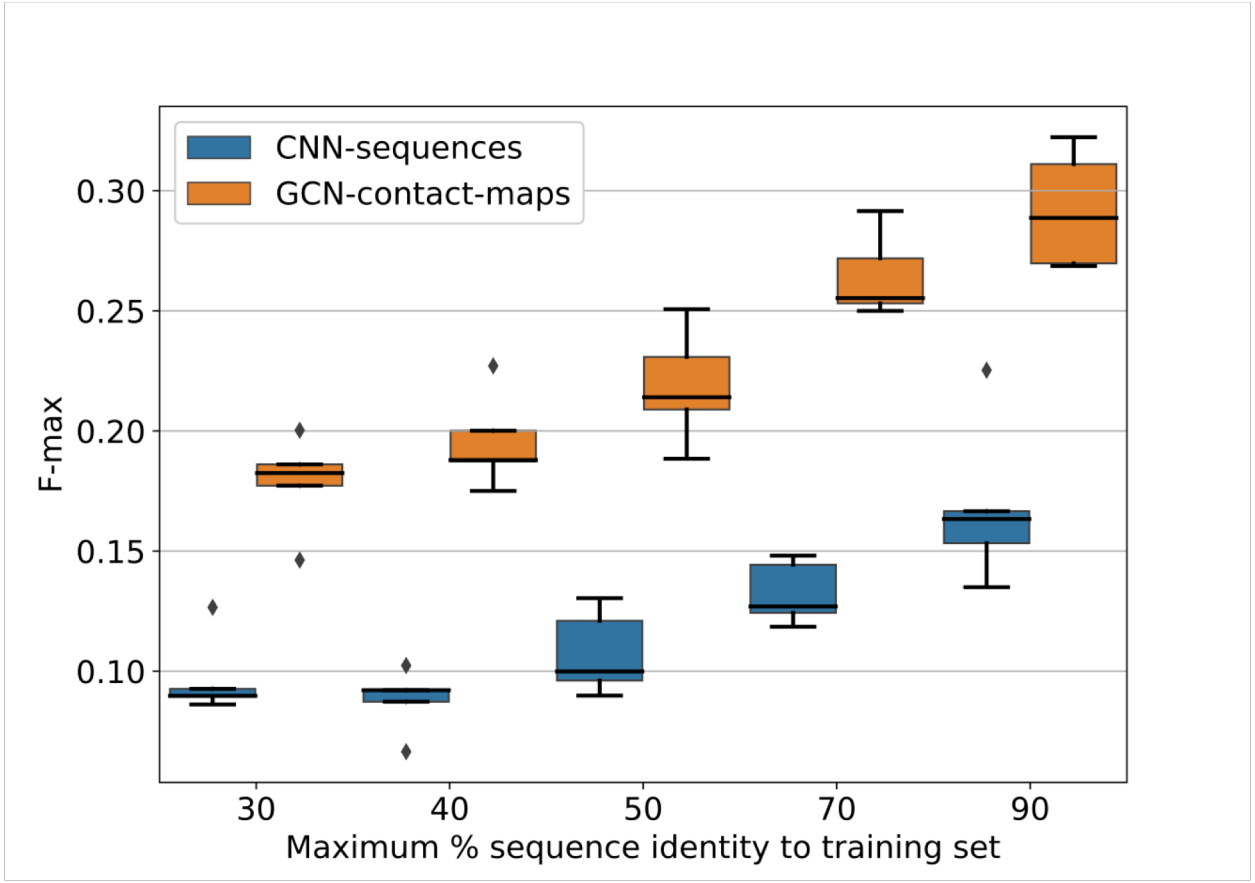
A [None, 600 , 600]
S [None , 600, 20]



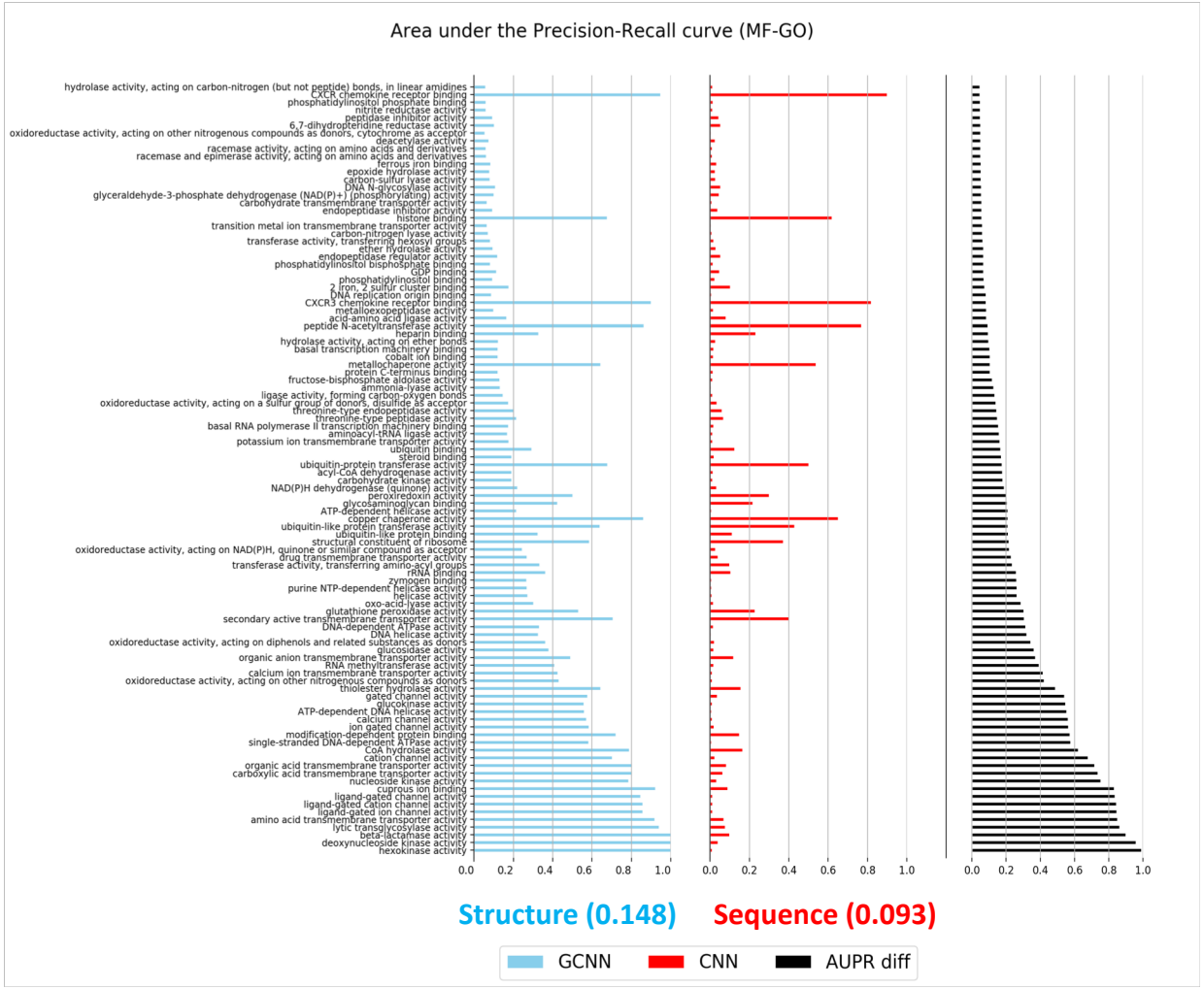
Optimizer: Adam, lr = 0.001
Loss: binary cross entropy
Batch size: 64

DNA binding TF activity

FLATIRON INSTITUTE
Center for Computational Biology



FLATIRON INSTITUTE
Center for Computational Biology



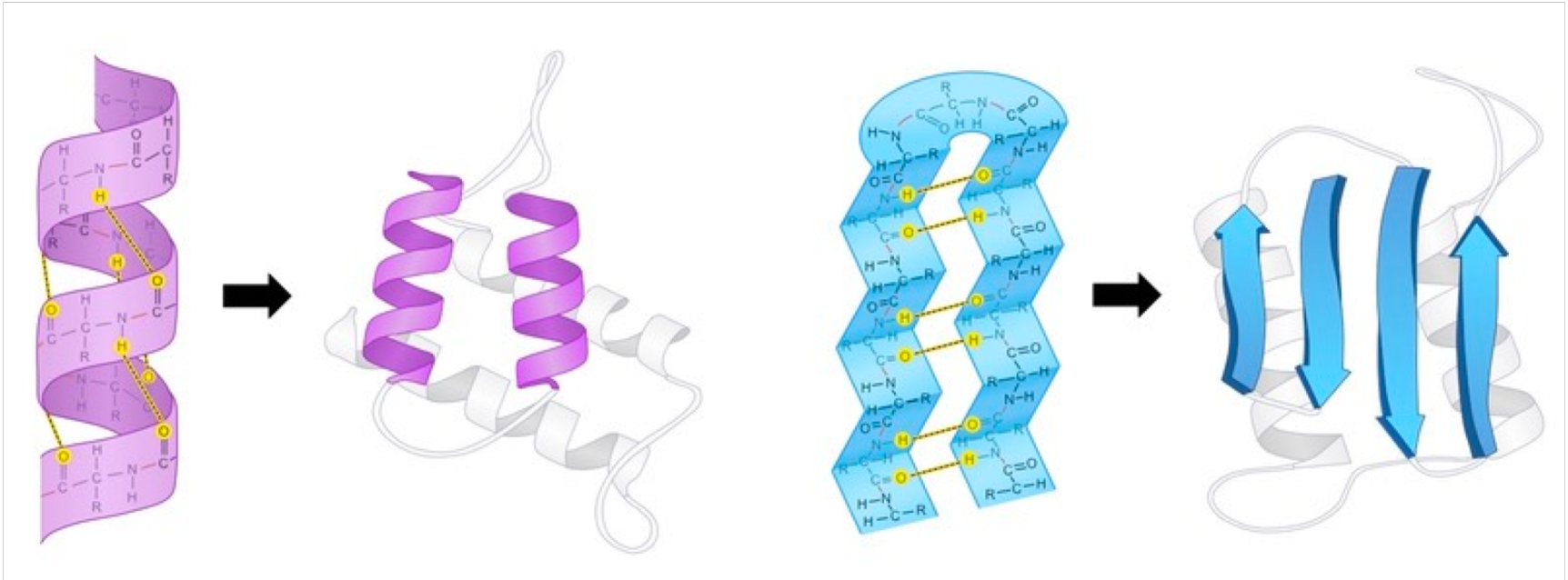
MF-GO: total 543
Top 100 GO terms

FLATIRON INSTITUTE
Center for Computational Biology

GCN autoencoder for protein fold discovery

FLATIRON INSTITUTE
Center for Computational Biology

Protein secondary structure



Alpha-helices **Beta-pleated sheets**

FLATIRON INSTITUTE
Center for Computational Biology

CATH classification of structures

- **Class** refers to the secondary structure content (e.g. mainly-alpha, mainly-beta, mixed alpha/beta or 'few secondary structures').
- **Architecture** refers to the general arrangement of the secondary structures irrespective of connectivity between them (e.g. alpha/beta sandwich).
- **Topology**, also known as the 'fold' level, takes into account the connectivity of secondary structures in the chain.
- **Homologous Superfamily** refers to domains that are believed to be related by a common ancestor.

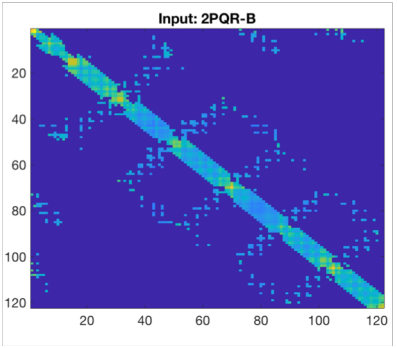
Each level has a **CATH code** associated with it. Have a look at the following:

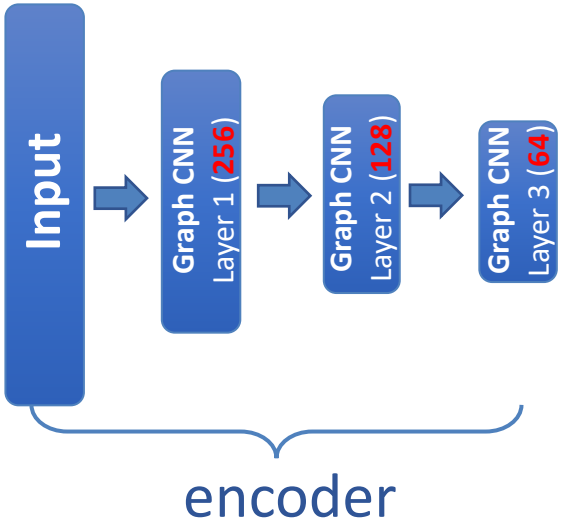
CATH Classification

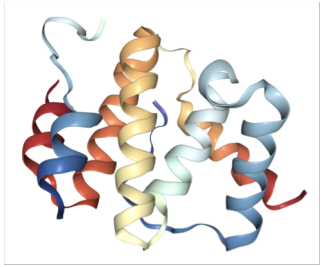
Level	CATH Code	Description
C	3	Alpha Beta
A	3.40	3-Layer(aba) Sandwich
T	3.40.50	Rossmann fold
H	3.40.50.620	HUPs

FLATIRON INSTITUTE
Center for Computational Biology

Autoencoders for reducing feature space for representing different protein fold classes

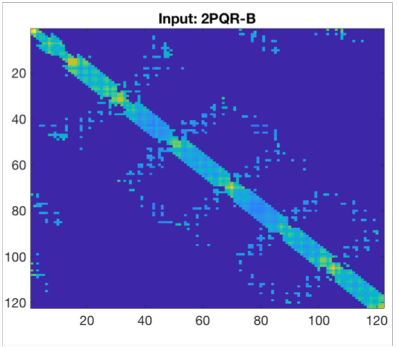


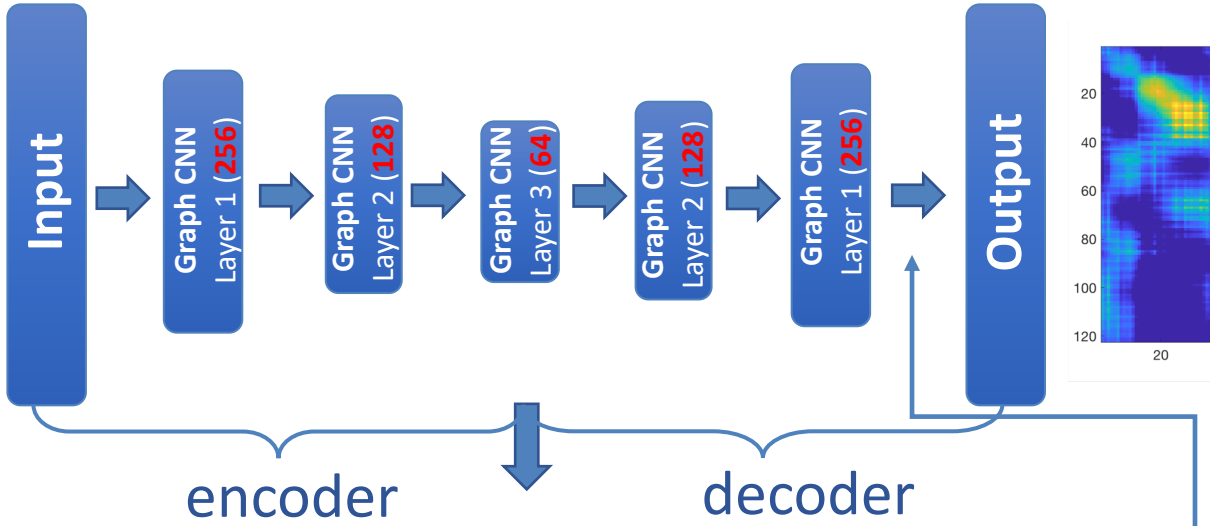


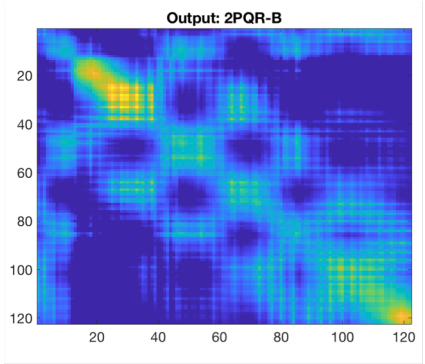


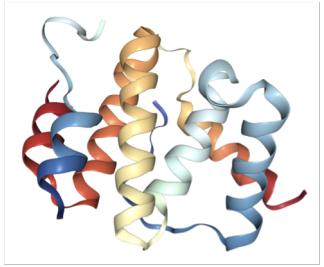
FLATIRON INSTITUTE
Center for Computational Biology

Autoencoders for reducing feature space for representing different protein fold classes





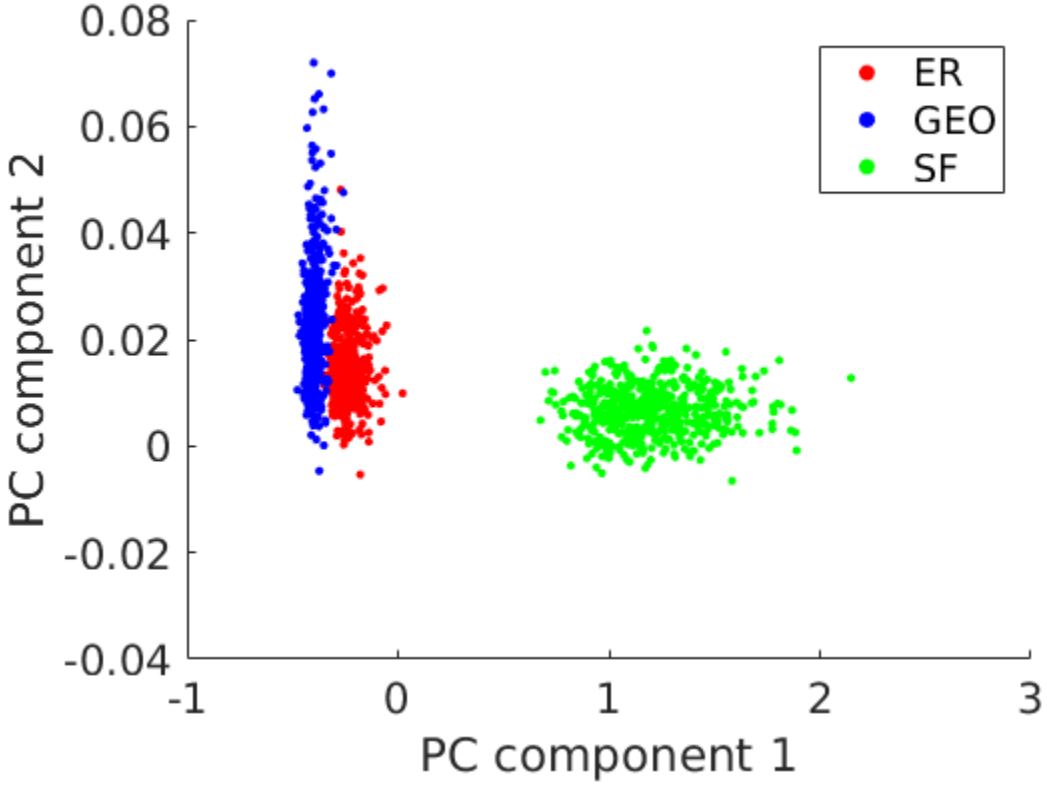




$$\bar{A} = \sigma(HH^T)$$

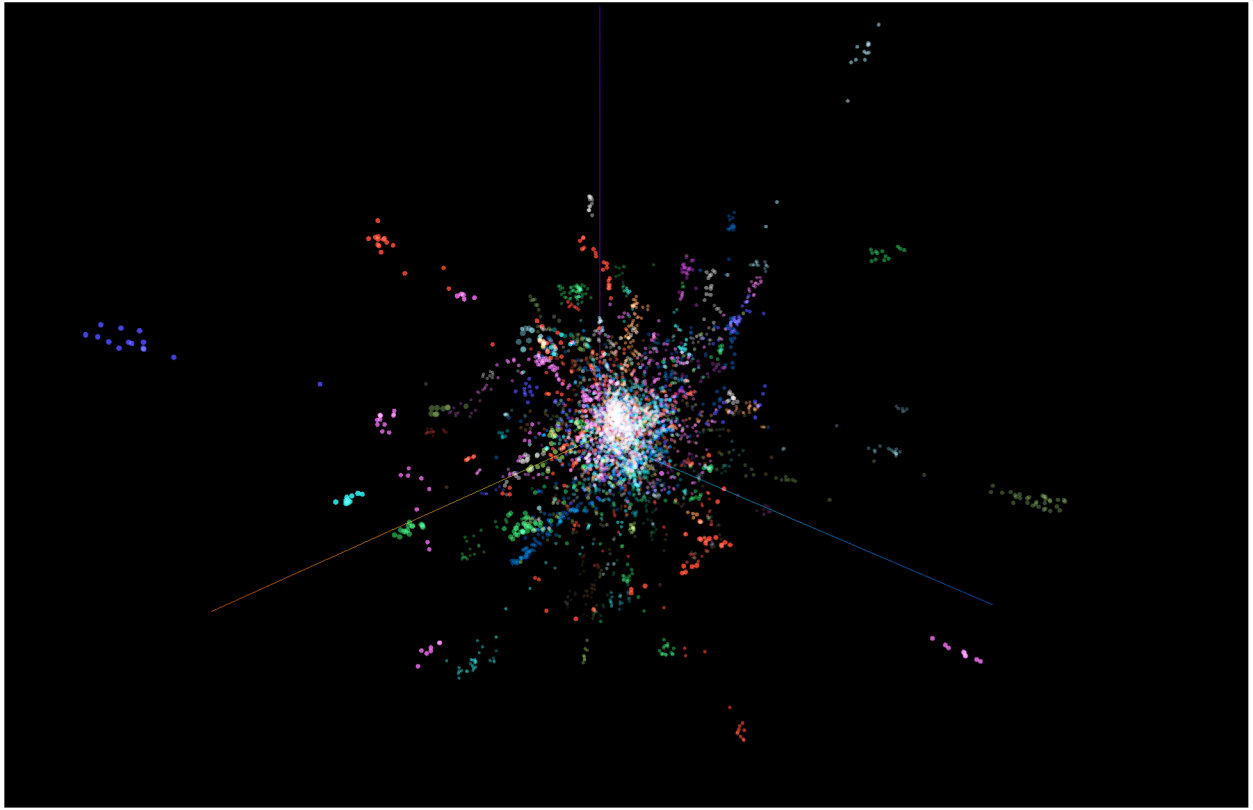
$$Loss(A, \bar{A}) = \|A - \bar{A}\|$$

FLATIRON INSTITUTE
Center for Computational Biology



FLATIRON INSTITUTE
Center for Computational Biology

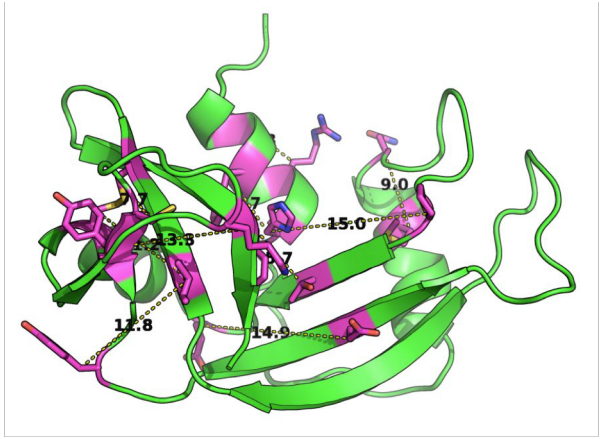
Galaxy of folds



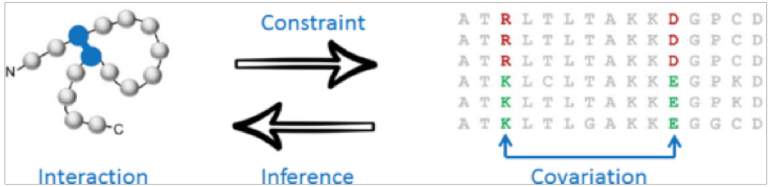
FLATIRON INSTITUTE
Center for Computational Biology

GCNN on Rosetta-predicted structures

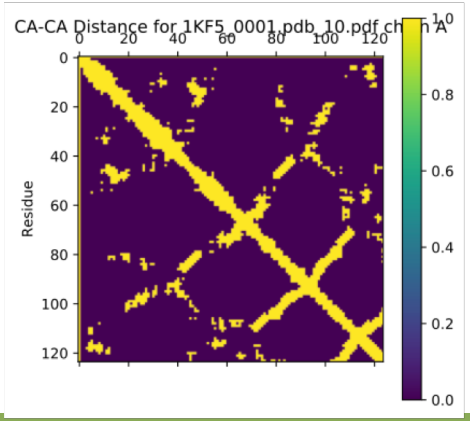
Predicting Contacts

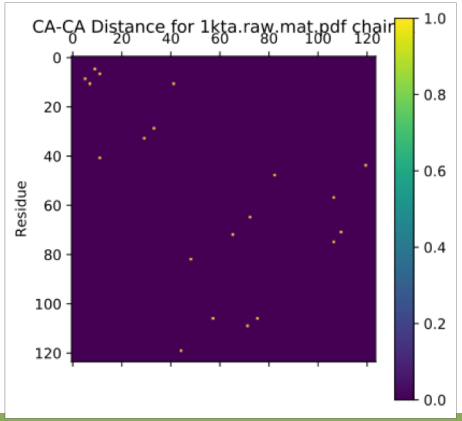


- Methods to predict contacts from sequence are commonly used in protein folding
  - Gremlin, MetaPSICOV



- Problems
  - Sparse
  - Not necessarily give you the interactions that discriminate functions
- Use protein structure prediction to generate

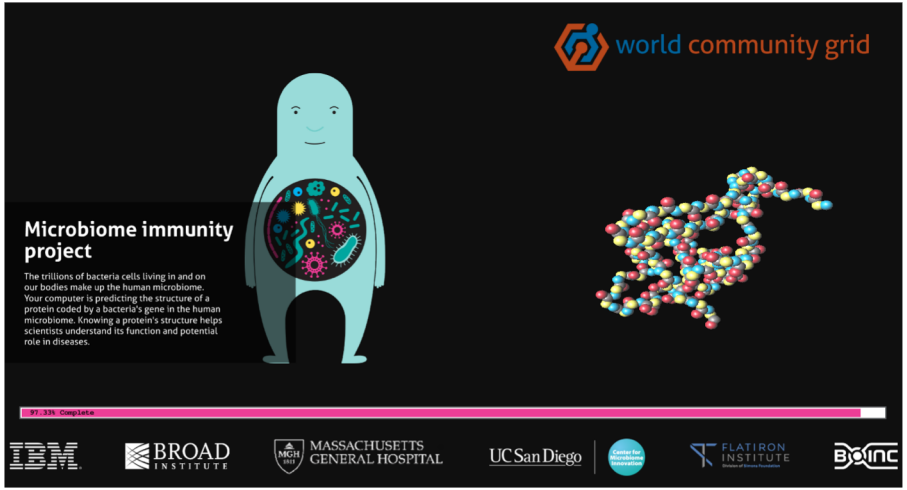




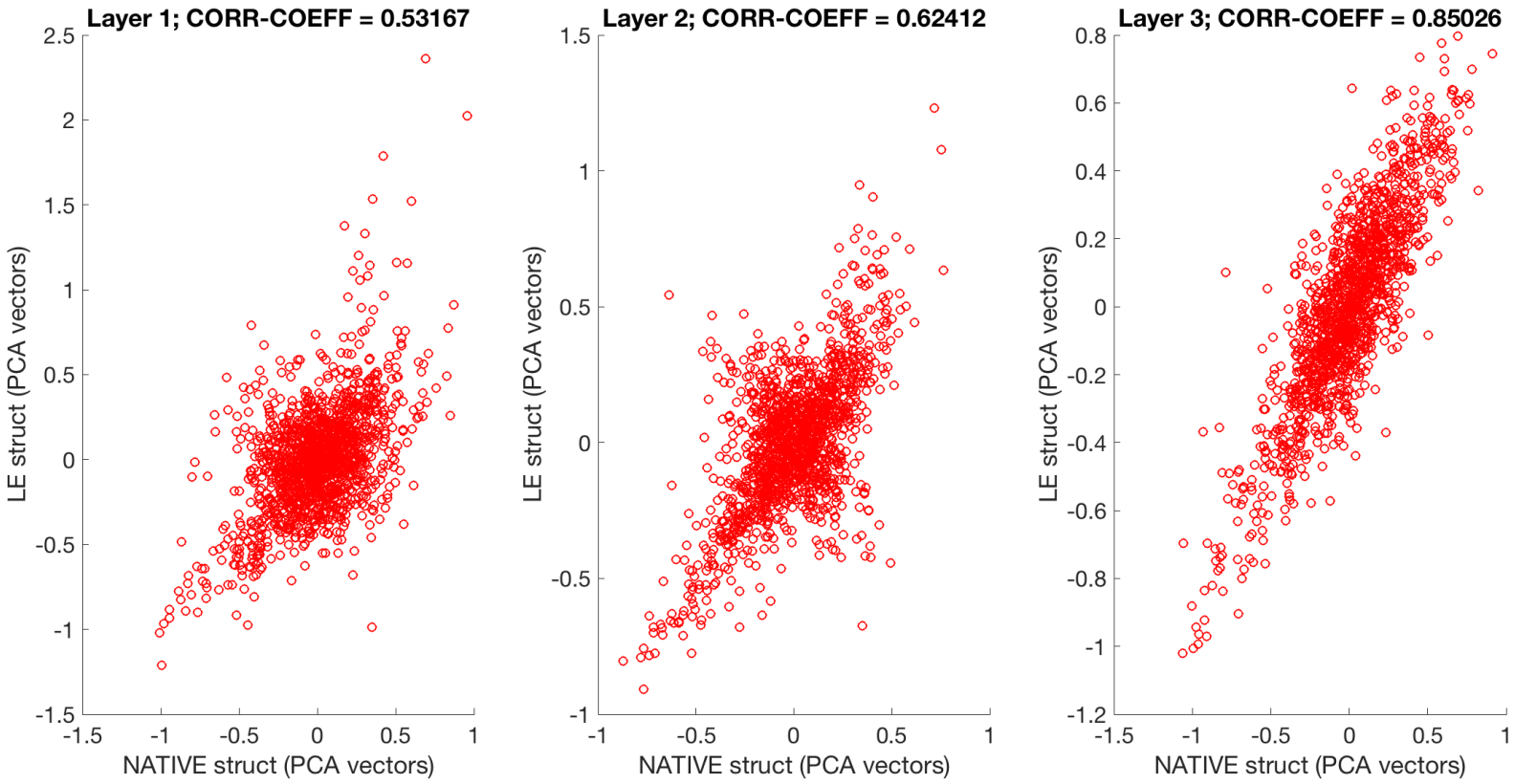
Deep Learning Methods for Function Prediction of Microbial Proteins

<u>Microbiome Immunity Project</u>

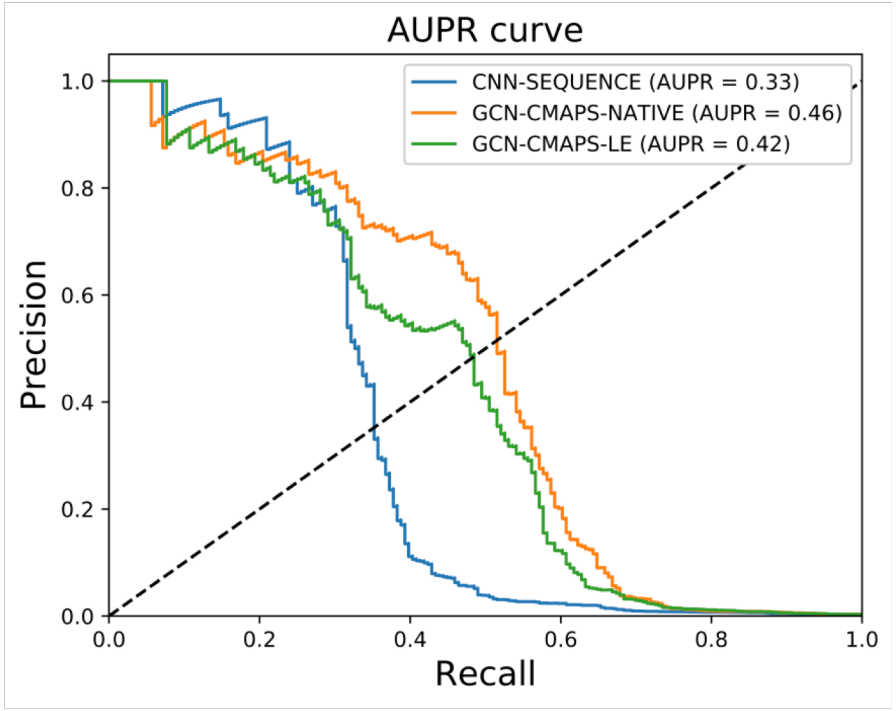
- 90,000 genomes
- 187 million sequences (12.8M gene families)
- predicting functions of new microbial protein families from sequences and contact maps

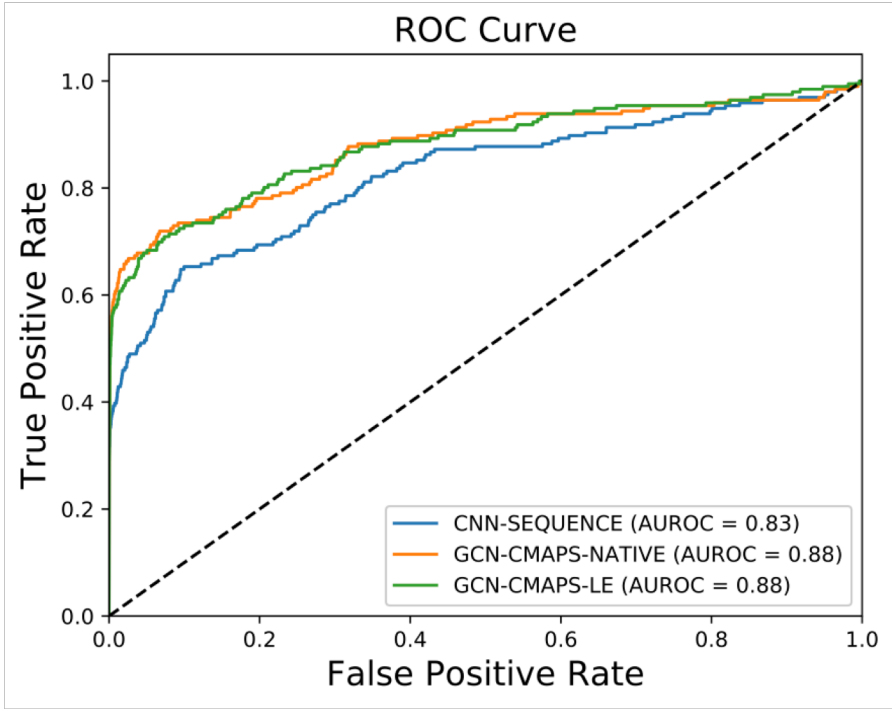


FLATIRON INSTITUTE
Center for Computational Biology


Layer 1; CORR-COEFF = 0.53167
Layer 2; CORR-COEFF = 0.62412
Layer 3; CORR-COEFF = 0.85026
LE struct (PCA vectors)
NATIVE struct (PCA vectors)

Function prediction: NATIVE vs LE structure





FLATIRON INSTITUTE
Center for Computational Biology

Acknowledgements

- Meet Barot, NYU PhD student, SF
- Emily Koo, NYU PhD student, NYU
- Doug Renfrew, Research Scientist, SF
- Bonneau Lab, SF