

# GraphNNs and GenAI in Drug Discovery & Design

Vladimir Gligorijević

PSI 2025 ML-SCI

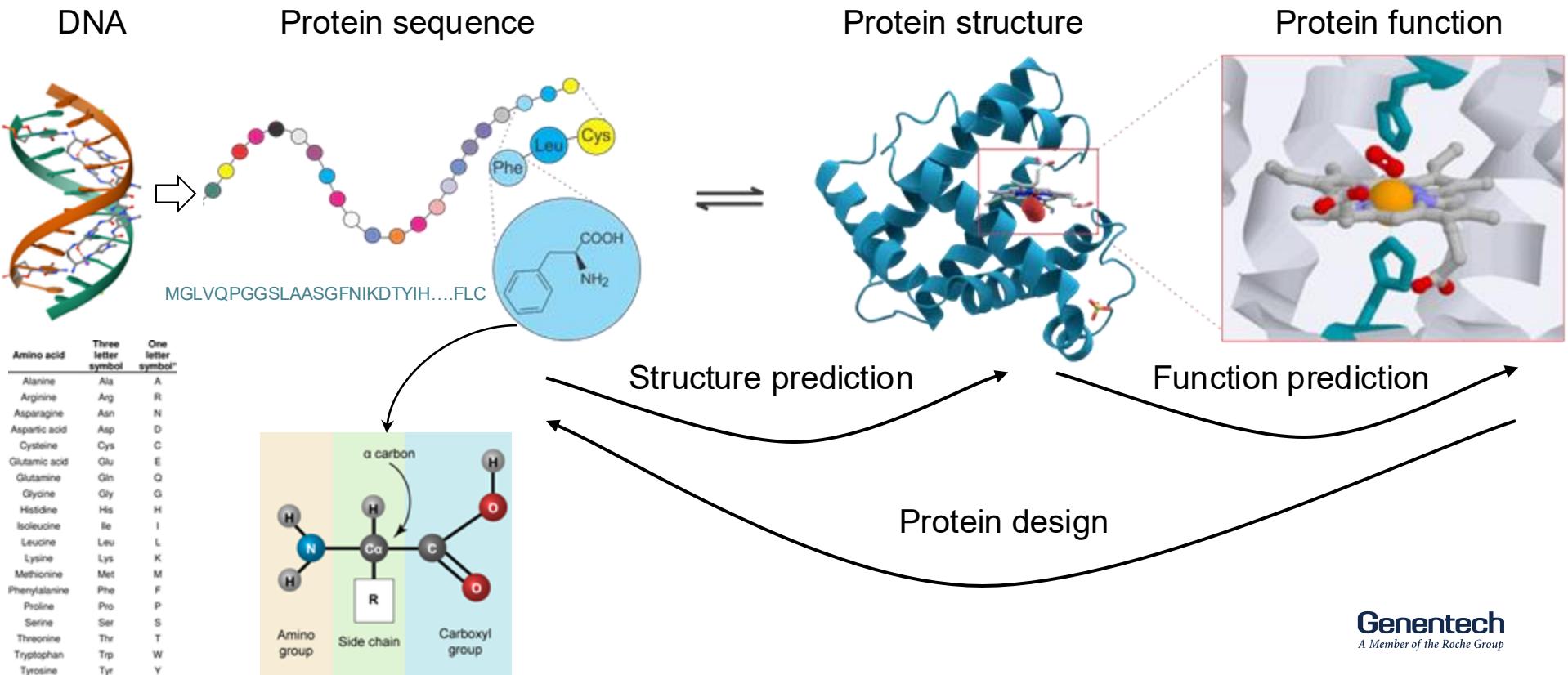
Petnica, Aug 8, 2025



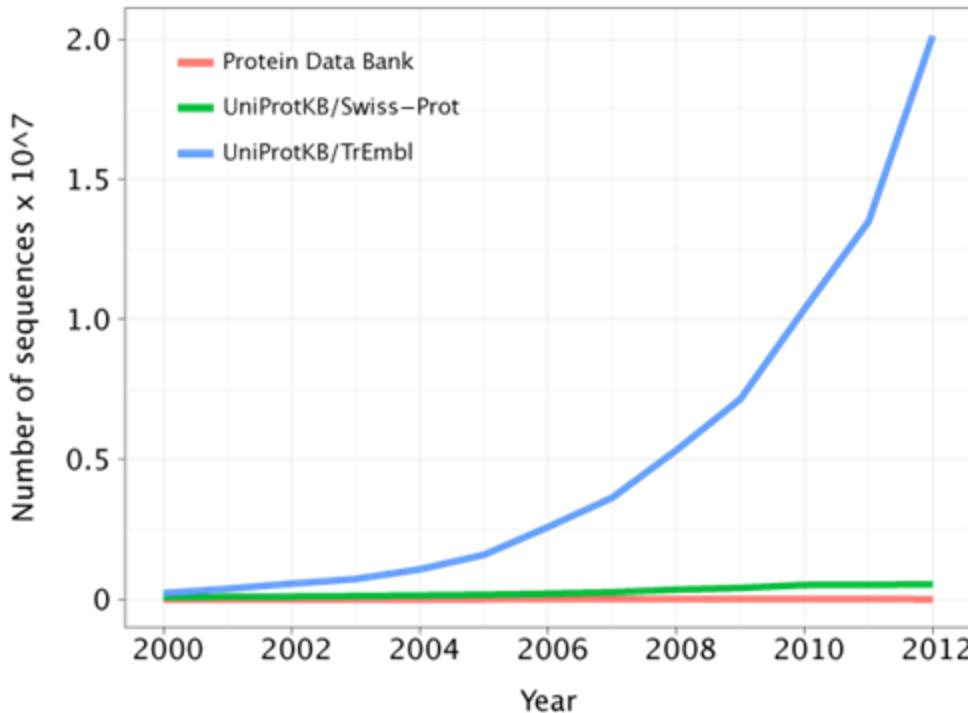
## Outline

- Basic concepts in protein biology & biotech industry
- Molecule representation (sequences & graphs)
- Model architectures for modeling sequence and structure data: ConvNN, RNN, Transformers, GraphNN, etc.
- Practical work:  
[https://github.com/VGligorijevic/PSI-ML\\_2025-protein-folds](https://github.com/VGligorijevic/PSI-ML_2025-protein-folds)
- Generative AI: diffusion models, multi-objective optimization, active learning, etc.

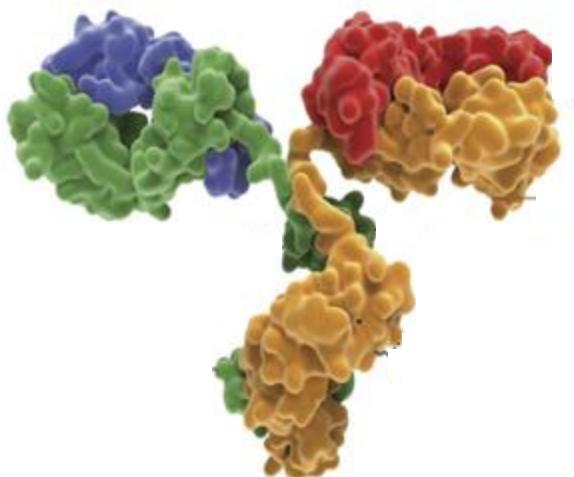
# Central dogma in molecular biology



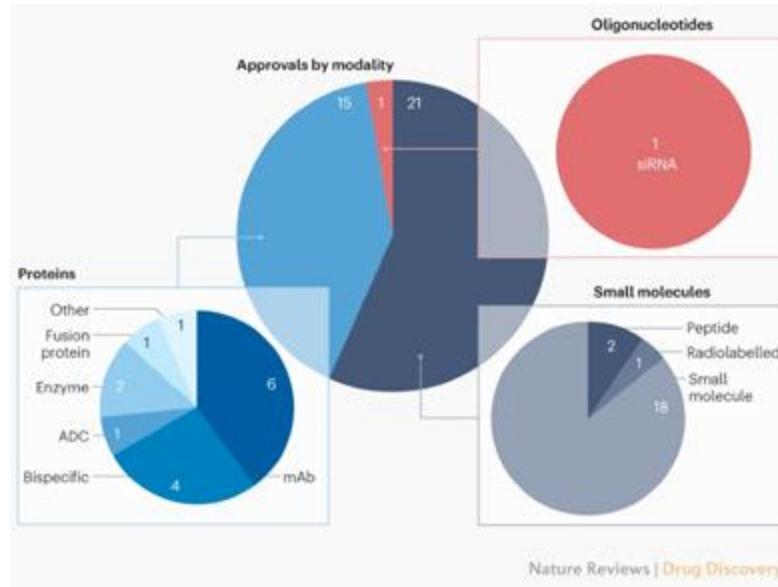
## Publicly available datasets



# Antibodies: our molecules of obsession



Antibodies: Y-shaped proteins belonging to the immunoglobulin superfamily



Abs were 20% of FDA approved drugs in 2022 (Mullard 2023)

Image credit: Antibody drug conjugate: the “biological missile” for targeted cancer therapy

# Key diseases of application

Autoimmune diseases

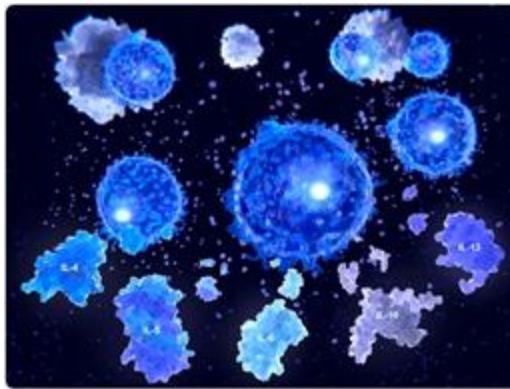
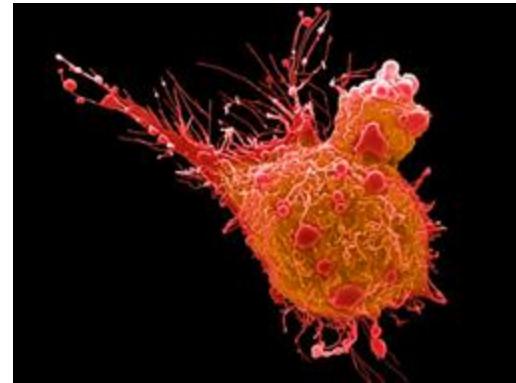


Image Credit: Juan Gaertner/Shutterstock.com

rheumatoid arthritis, Crohn's,  
ulcerative colitis, kidney  
transplants, asthma

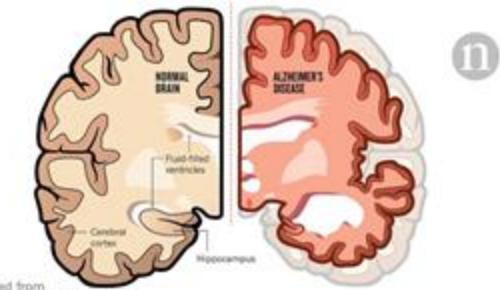
Cancer



Credit: Steve Gschmeissner/Science Photo Library

breast cancer, leukemia,  
rectal/prostate cancer,  
neuroblastoma lymphoma,  
melanoma, ..

Alzheimer's Disease

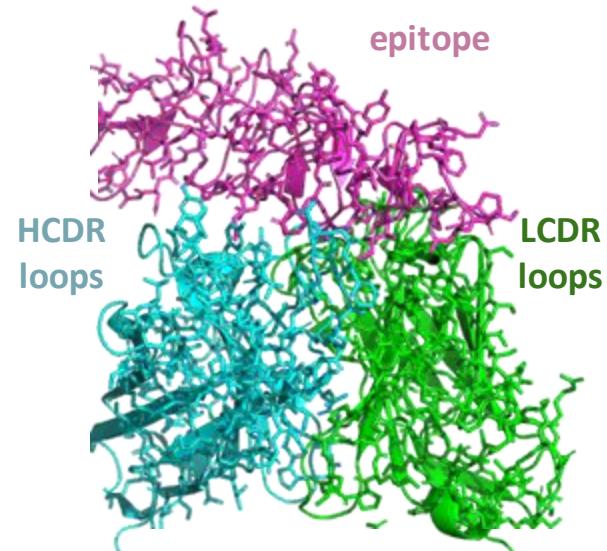
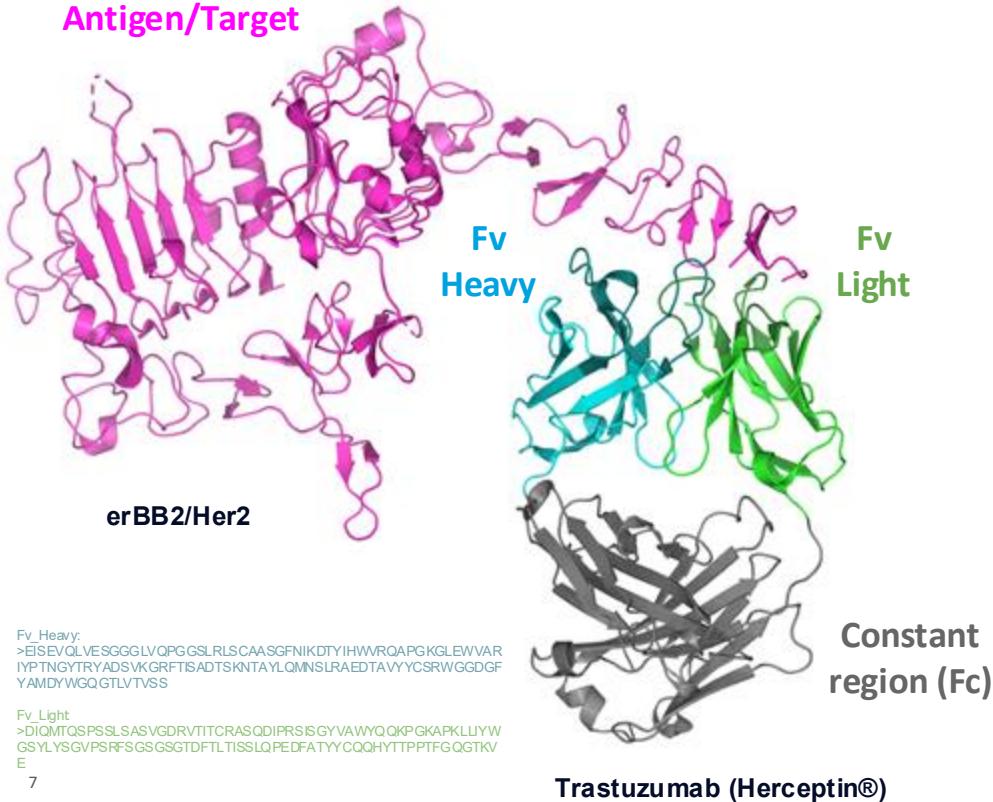


Adapted from  
illustration by Stacy Jannis/  
Alzheimer's Association

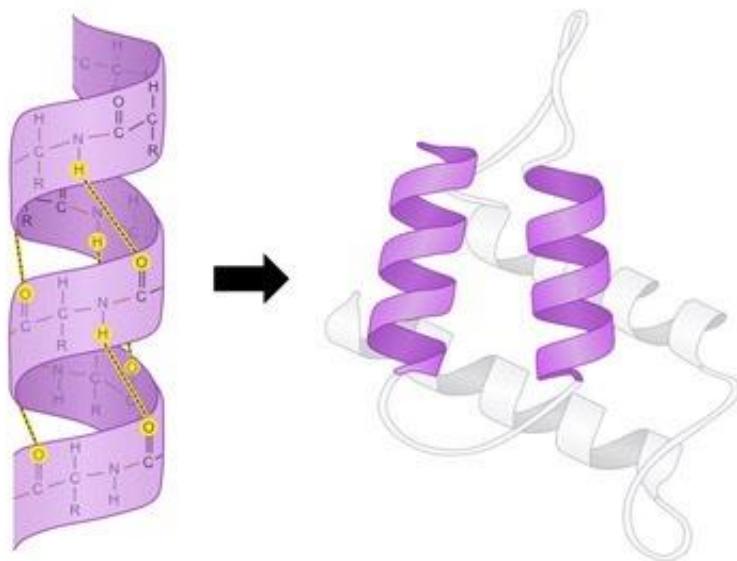
Lecanemab  
27% improvement in  
cognitive decline

# Anatomy of an antibody

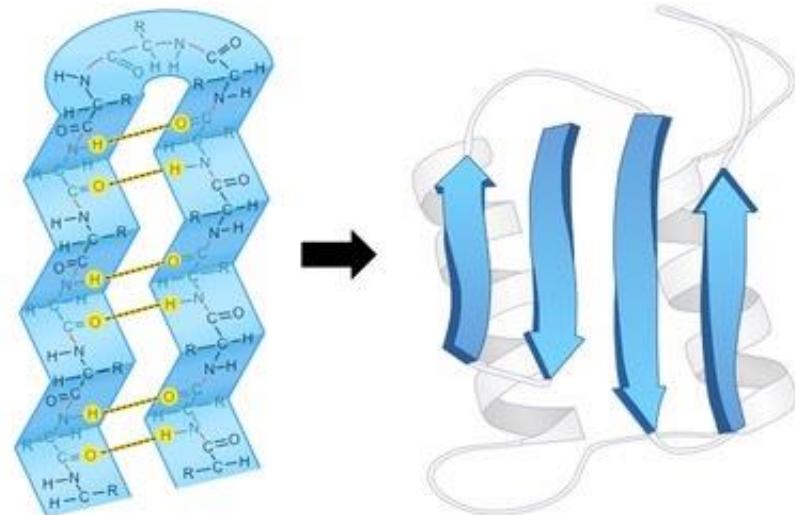
Antigen/Target



# Protein secondary structure



Alpha-helices



Beta-pleated sheets

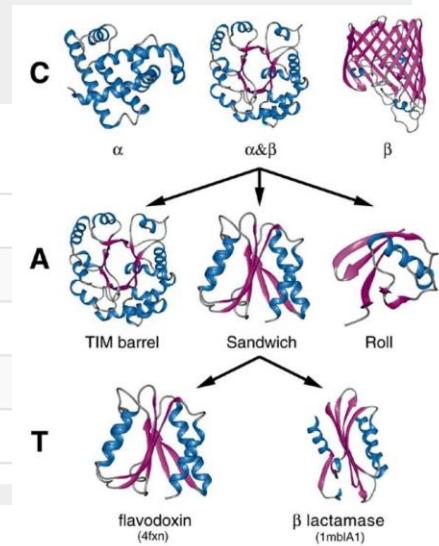
# CATH classification of structures

- **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



# How do we represent proteins/antibodies in our ML algorithms ?

## Sequence Features

- Amino Acid Sequence (e.g., 1-hot encoding)
- Chemical descriptors
- Evolutionary information (MSAs, numbering schemes; e.g., AHo)

## Structural Features

- Voxelization (atom point cloud)
- Protein Graph (inter-residue distances/contact map + torsion angles)
- Secondary structure
- Protein surface



## Learned Sequence Features

- ProtVec
- Autoencoders (AE)
- Language Models (ESM, UniRep, AntiBerty, etc.)

## Learned Structural Features

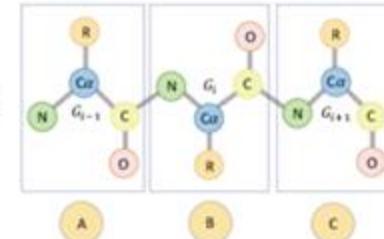
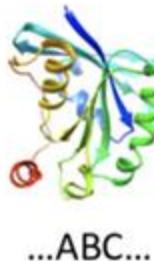
- Surface fingerprints (dMASIF)
- Tokenizing the structure + Language Models (multi-modal LMs)

# Protein featurization for ML models

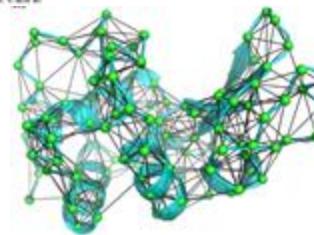
```

AREALVPFG-- -KDAPELLIAV TVLTSMEASD LVD-LGMTLS
SREILEPYG-- -KERPLLIGV TVLTSMESAD LQG-IGILSA
VVNALQSITL- -KEKPLLIGV TILTSLDGSD LKT-LGIQEK
AKDGLIAGTPA GHHSVPKLLAV TELTSISDDV LRNEQNCRLP
  
```

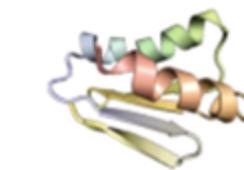
Original AA sequence or MSA (1-hot encoding)



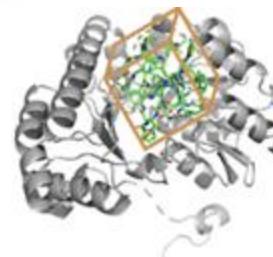
Backbone “frames” (used in AlphaFold, RosettaFold, etc.)



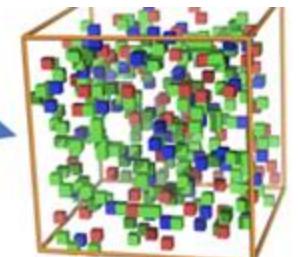
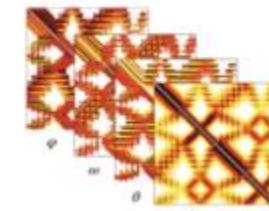
Protein graph (GNNs,  
EGNNs, etc. )



Distance and orientations (DeepAb, trRosetta, etc. )



Voxelized atome point cloud (3D CNNs, MAE, etc.)



## 1D Representations

Sequence

Physicochemical descriptors

## 2D Representations

Contact maps

Graphs

## 3D Representations

Surfaces

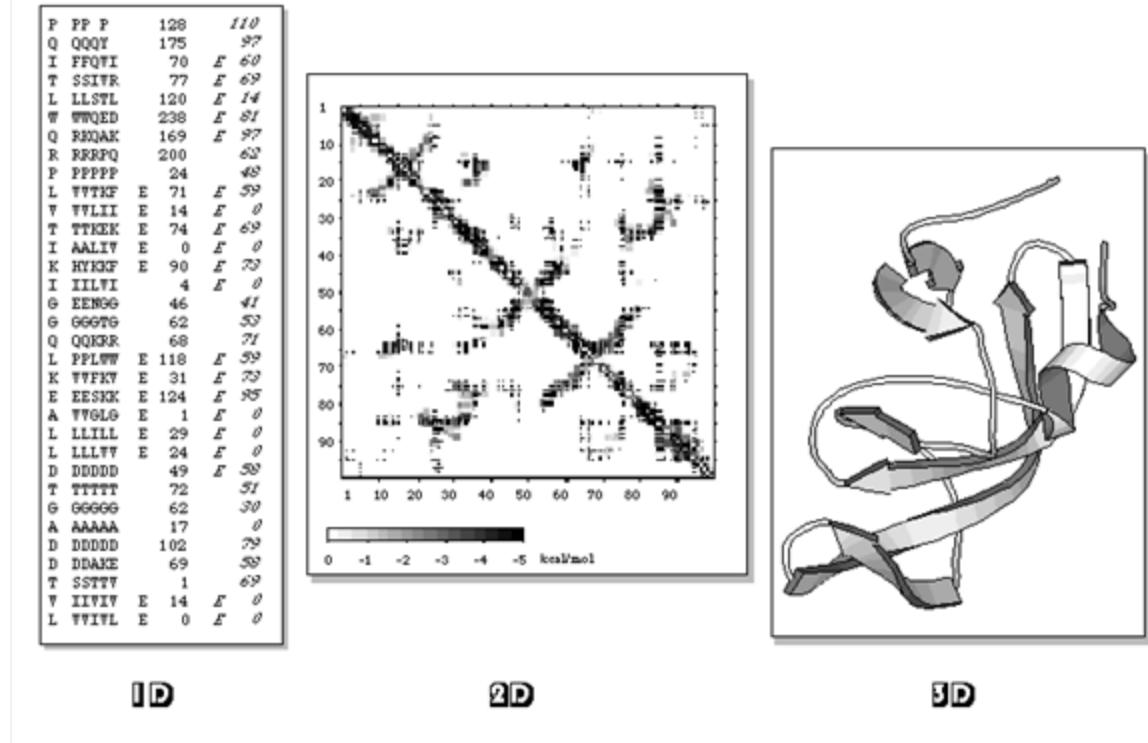
Point clouds

Geometric graphs

## 4D Representations?

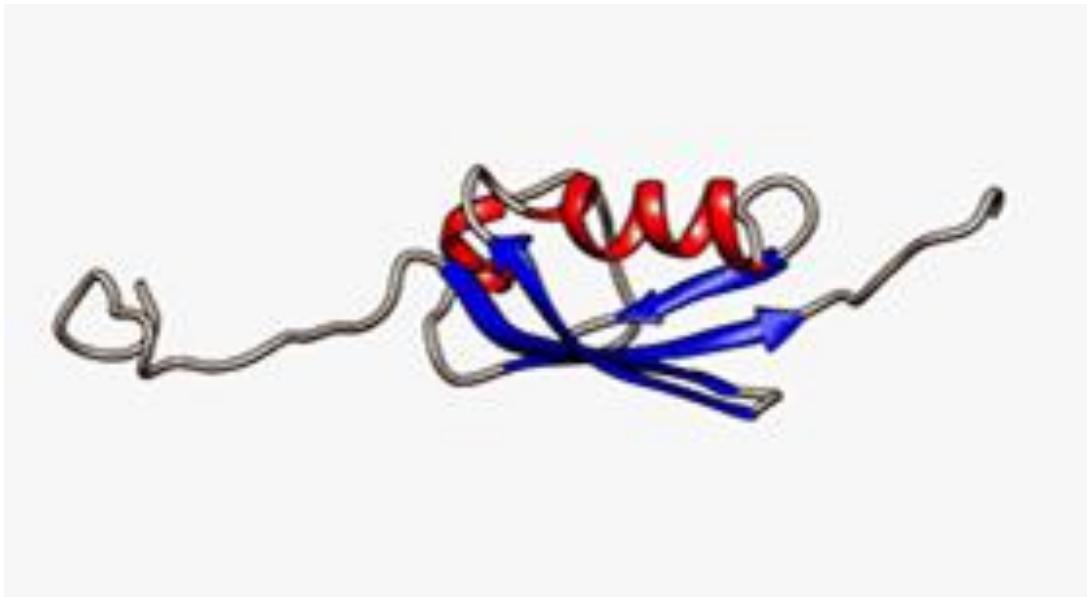
Structure + Dynamics!

P	PPP	128	E	110
Q	QQY	175	E	97
I	FFQFI	70	E	60
T	SSITR	77	E	69
L	LLSTL	120	E	14
W	WWQED	238	E	81
Q	REQAK	169	E	97
R	RRRPQ	200	E	62
P	PPPPP	24	E	48
L	TTTTF	71	E	59
V	TVLII	14	E	0
T	TTKEK	74	E	69
I	AALIV	0	E	0
K	HYKGF	90	E	73
I	IILVI	4	E	0
G	EENGG	46	E	41
G	GGGTG	62	E	53
Q	QQKRR	68	E	71
L	PPLWW	118	E	59
K	TVFKT	31	E	73
E	EESKK	124	E	75
A	TVGLG	1	E	0
L	LLILL	29	E	0
L	LLLTV	24	E	0
D	DDDDD	49	E	58
T	TTTTT	72	E	51
G	GGGGG	62	E	50
A	AAAAA	17	E	0
D	DDDDD	102	E	79
D	DDAKE	69	E	59
T	SSTTV	1	E	69
V	XIVIV	14	E	0
L	TVVTL	0	E	0



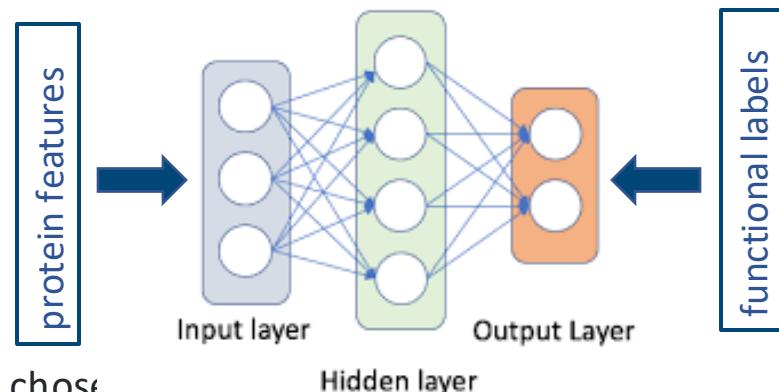
# Protein dynamics

Proteins move(!) How can we incorporate this into ML models in a principled manner?



# Deep Learning vs. Traditional ML

- Traditional ML methods (e.g., SVM, Logistic Regression, Random Forest, KNNs, etc.)
  - require feature engineering
  - offer more interpretable solutions
  - usually used in low-data regime
- Deep Neural Networks
  - high-level features
  - don't require feature engineering (depends on the choice)
  - require lots of data points



# Protein representation

- Sequence representation (example: TRI27\_HUMAN):

k-mers (k=3):

MASGSVAEC...SMETSP  
MASGSVAEC...SMETSP

MAS GTP MET ... ETS TSP

counts = [1, 0, 6, ..., 4, 2]

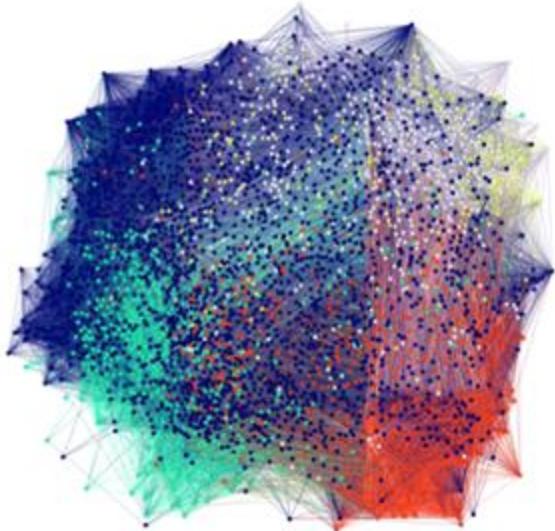
1-hot encoding:

	M	A	S	G	V	E	C	:	S	M	E	T	S	P
0	0	0	0	1	0	...	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

MSA PSSM:

	M	A	S	G	S	V	A	E	C	:	S	M	E	T	S	P
-1	-2	-1	4	-1	...	2	-1									
-1	3	-4	-2	2		7	-1									
-2	8	-2	-1	0		0	-2									
-7	-3	-1	0	-12		2	-7									
-3	-8	-2	9	-3	...	-1	-1									
-4	-2	-7	4	-1		1	-1									
-1	-2	-1	0	-9		8	-1									
-2	-2	-1	-11	-1		-9	-1									
-3	-2	-10	4	-1	...	7	-1									
⋮	⋮	⋮	⋮	⋮		⋮	⋮									
-7	-2	-5	4	5	...	2	-1									
-3	-3	-7	9	-13		-3	-1									
-1	-2	-1	4	-7		-5	-1									
-7	-22	-1	7	-1	...	3	-1									
-12	-2	-2	2	-14		-6	-1									
-16	-25	0	-17	-1	...	7	-1									

# Features: broad view of biology requires diverse features



Networks

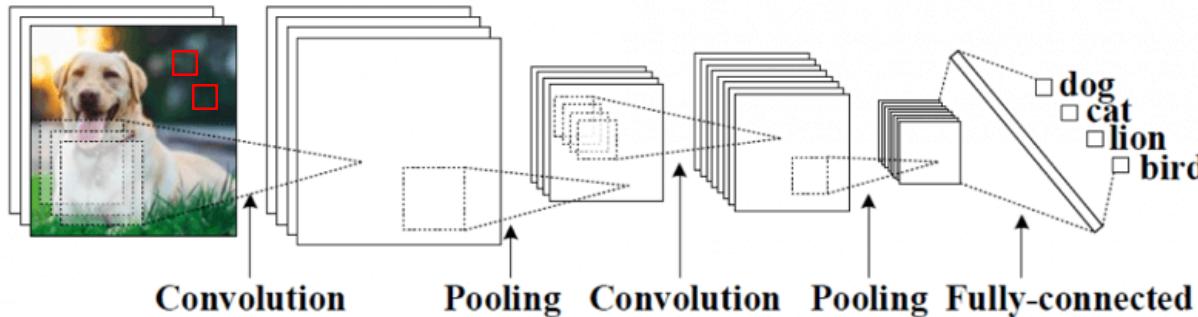
FRHEVILVDALLKRH.SVSAAR.ELDI...PQPTASRGLARERK.ALCDF  
PRLLRLFDALFTC.SVTKAEE.RLQI...SQPTVSIWHLARLREQ.ELEDDP  
PKLLOLFDVLYQCR.SVTRAEE.QLQI...SQPTISIWLARLIRE.QLNDE  
FMHETVFOALMRTK.SVSLAAE.MLDW...PQPTLSRHLKQLRLRE.HFGNO  
N....VFDALYRIC.SACKAAH.ALSN...PQPTLSRKLWLAQRLRT.HFDDP  
LNLLVSLDTLLAEN.SVERAAE.RLQI...SNSAMSRLSIRE.IFCDF  
LNLLFALDVLIAEG.SVARAAE.RLEI...SPSAMSRKLARLIRE.ATGDP  
LNLLVTLNALLAEQ.SVAGAAE.RLQI...SPSAMSRALAPLRLRA.TMNDP  
MNLLVALDALLQDG.SVVGAAQ.RMNI...SPAAMSRTLGIRRE.ALCDP  
LNLLVALNALIIEER.SVTGATR.RLHI...SSPVMSRPLARLIRE.TLDDP  
LNLLAALDALLIEEG.SVAGAAA.RLHV...TAPAMSRSLGIRR.TTGDD  
ENLLVCLVKVLIELZEL.SVTRAEE.RLCI...SOSAVSKSLAKLRLQ.QFDDP  
LNLLVCLVKVLFEEL.NVTRTAH.RLCE...SOSAVSKALAKLRLQ.QFDDP  
LNLLVILVKVLLEEQ.SVTRAAS.RLHI...SOSALSKSLSNRRLRD.TLDDP  
LNLLKVLVLMLEER.SVARCAE.RLFV...SPGAVSHALAKLRLQ.MFSDP  
LNLLVALVRLVIZES.SVSKAAV.RLNI...SOSAMSRVLGRRLRD.LLGDP  
LNLLRALDALDDE.RNVTRAJV.RLAI...TQPAMSGMLTRRLRD.SFDDP  
LNLLMIFDAILAE.GVTRAAE.RLAM...SOSAVSKGLACLRLQ.AFGDP  
LNLLPILLALHEDAR.SVSMAAQ.QLCM...SQPGVSTALAKLRLT.AFGDP  
LNLLVTLQVLLTEH.NVTRAAS.RLNE...SOPSVSVHLAKLRLA.IFDDP  
LNLLVSLQVLLZEC.NVTRAQ.RLHV...SQPALSAQLGRRLRH.LFDDP  
LNLLVTLAVLRLER.SVTRAEE.CLHI...SOPAVSGALGRRLRE.LFDDP  
ENLLVVLQALLSE.RNVTRAQ.RLAM...SQPAVSHALGRRLRE.LFDDP  
LNLLRVLDALLRER.NVSRRAE.RLAL...SOPAVSNALNRLRE.LLDDP  
LNLLRVFDMLLREQ.NVSRRAE.RLAL...TQPTVSNALRLRD.QLGDP

Sequence



Structure

# Convolution Neural Networks



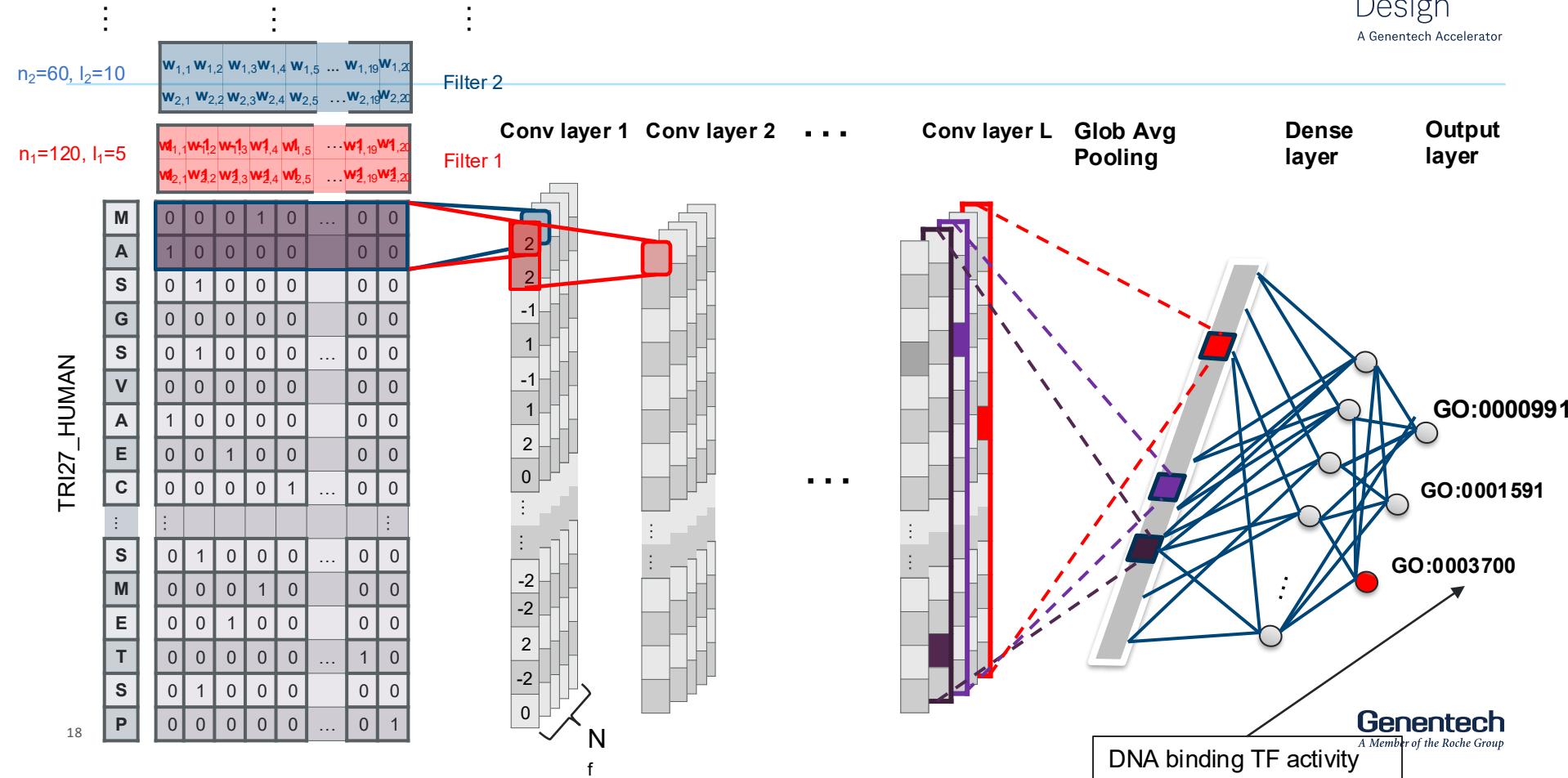
$$h_{ij}^k = \sigma((\mathbf{W}^k * x)_{ij} + \mathbf{b}_k)$$

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

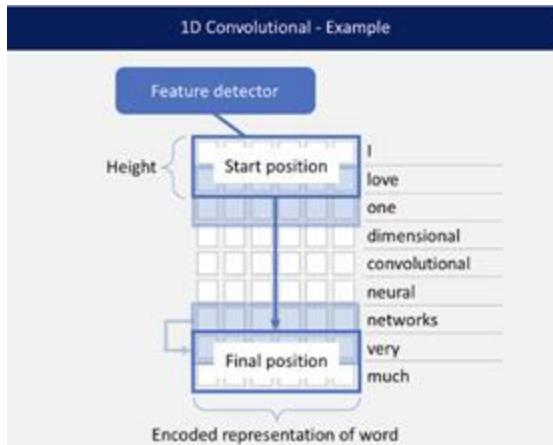
# CNN for sequence-to-function mapping

Prescient  
Design  
A Genentech Accelerator



# Architectural evolution

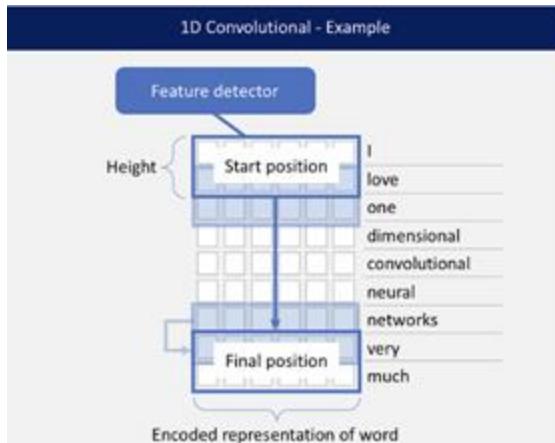
## Convolutional Neural Networks



In this example for natural language processing, a sentence is made up of 9 words. Each word is a vector that represents a word as a low dimensional representation. The feature detector will always cover the whole word. The height determines how many words are considered when training the feature detector. In our example, the height is two. In this example the feature detector will iterate through the data 8 times.

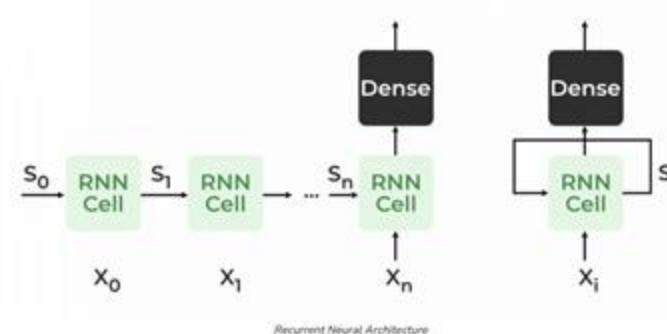
# Architectural evolution

## Convolutional Neural Networks



In this example for natural language processing, a sentence is made up of 9 words. Each word is a vector that represents a word as a low dimensional representation. The feature detector will always cover the whole word. The height determines how many words are considered when training the feature detector. In our example, the height is two. In this example the feature detector will iterate through the data 8 times.

## Recurrent Neural Networks



<https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/>

GRU: On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. Cho et al., 2014

# Architectural evolution

## Attention Is All You Need

Ashish Vaswani\*  
Google Brain  
avaswani@google.com

Noam Shazeer\*  
Google Brain  
noam@google.com

Niki Parmar\*  
Google Research  
nikip@google.com

Jakob Uszkoreit\*  
Google Research  
usz@google.com

Llion Jones\*  
Google Research  
llion@google.com

Aidan N. Gomez\* †  
University of Toronto  
aidan@cs.toronto.edu

Lukasz Kaiser\*  
Google Brain  
lukasz.kaiser@google.com

Illia Polosukhin\* ‡  
illia.polosukhin@gmail.com

2017

120K citations

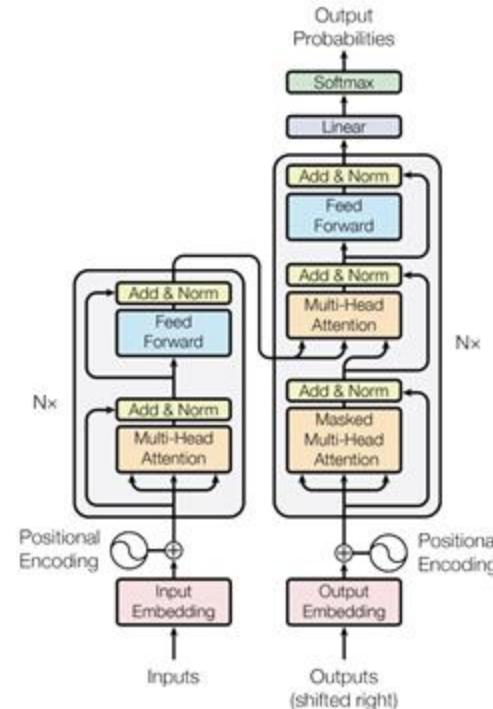
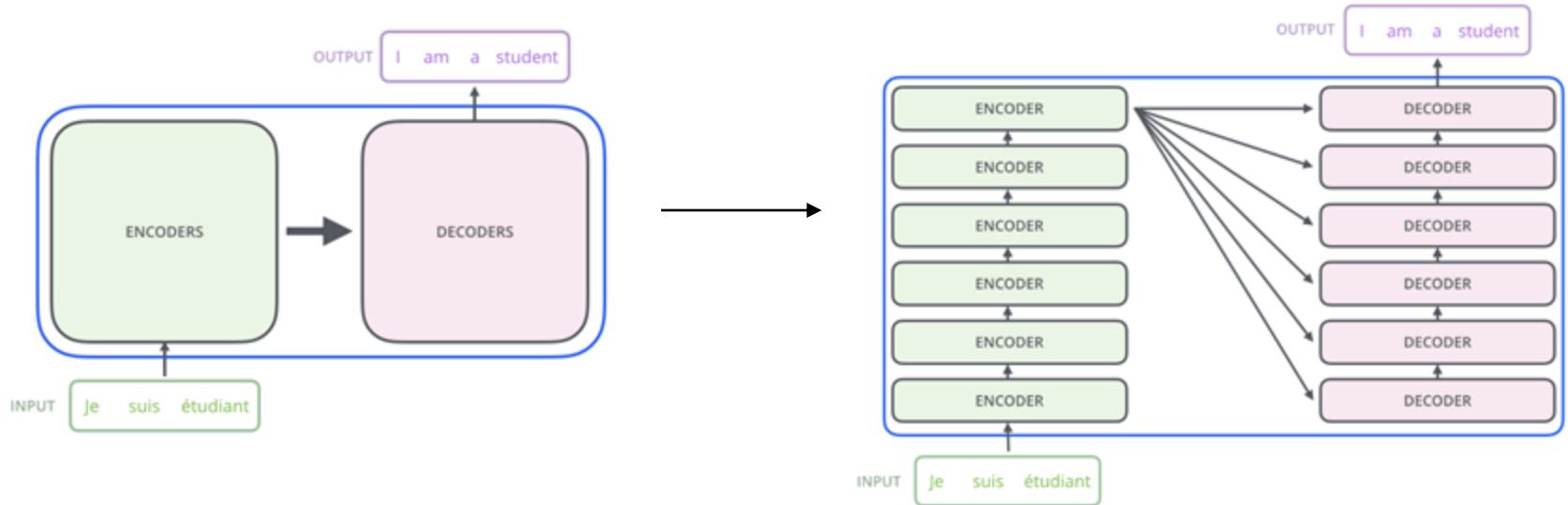


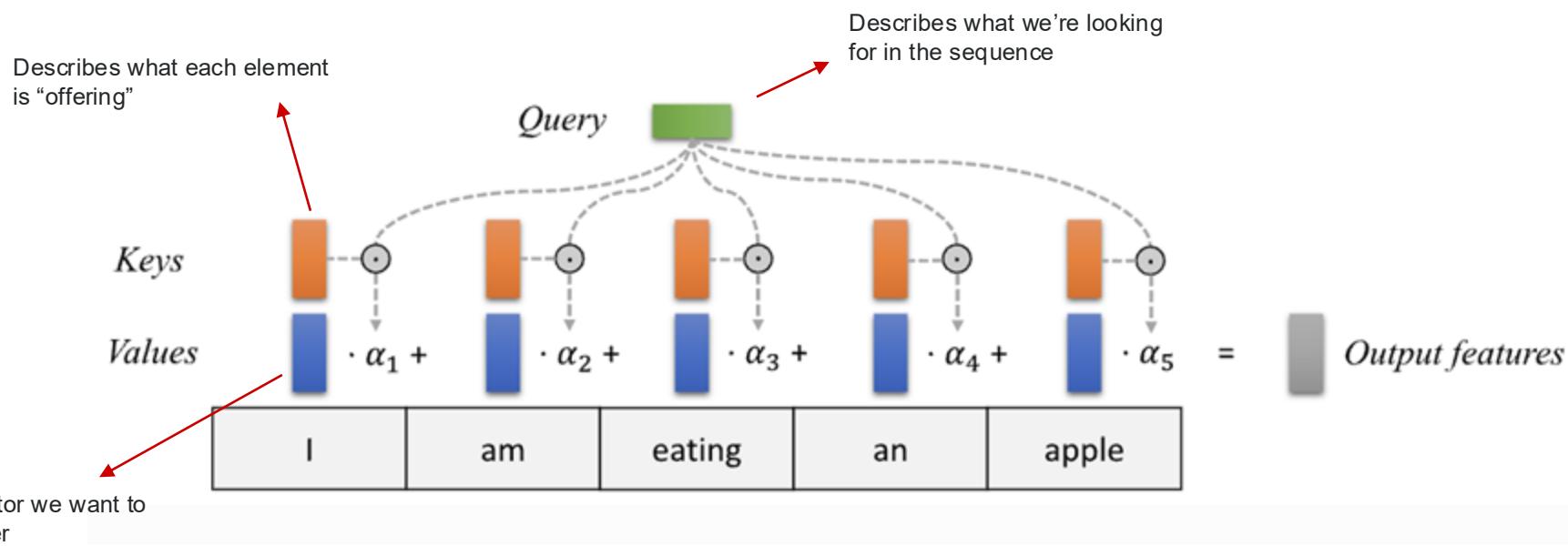
Figure 1: The Transformer - model architecture.

# Transformers



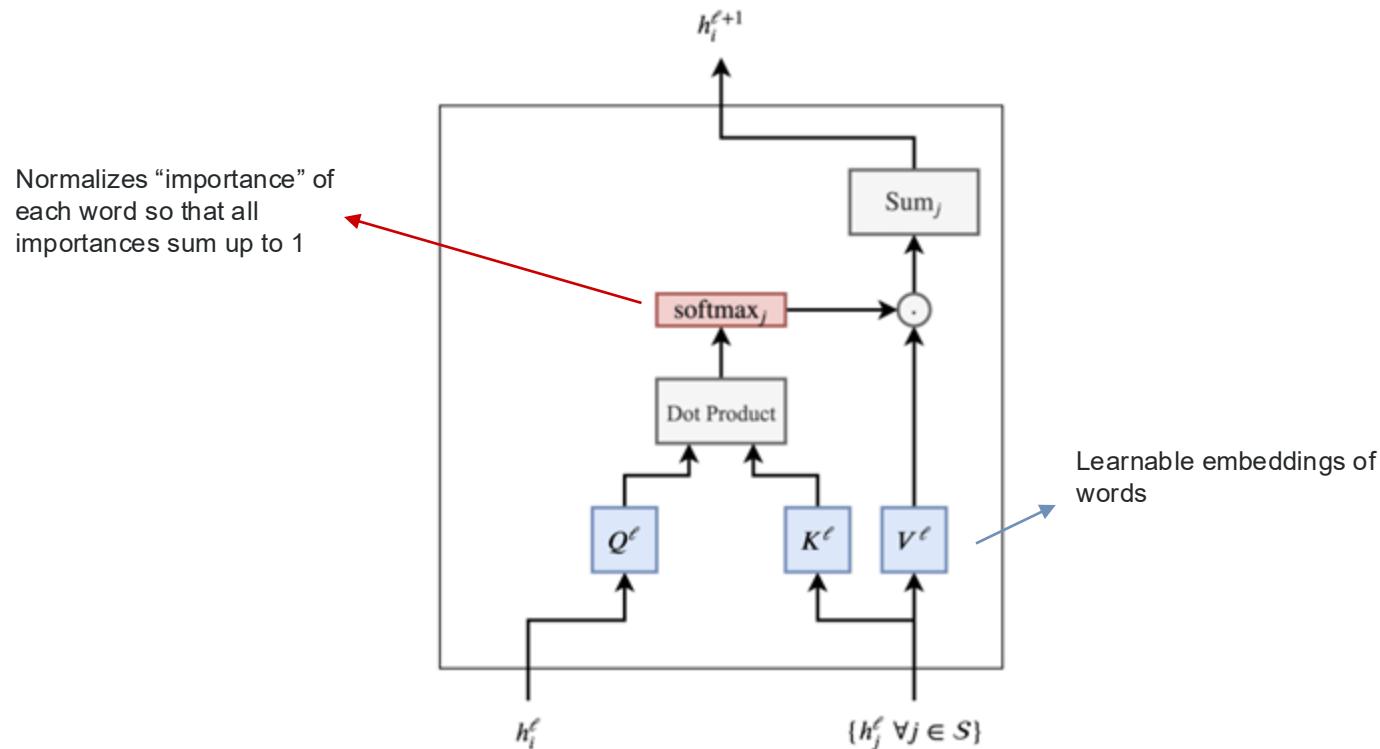
# Transformers: attention mechanism

## Attention



[https://uvalc-notebooks.readthedocs.io/en/latest/tutorial\\_notebooks/tutorial6/Transformers\\_and\\_MHAttention.html](https://uvalc-notebooks.readthedocs.io/en/latest/tutorial_notebooks/tutorial6/Transformers_and_MHAttention.html)

# Transformers



# LLMs

ChatGPT 4o ▾



what are you?

I am an AI designed to assist you with your research and various tasks. How can I help you today?

how were you trained?

I was trained using a vast dataset that includes books, articles, websites, and other text sources. This dataset allows me to understand and generate human-like text. My training involved machine learning techniques, particularly deep learning models, to analyze patterns in the data and learn to predict and generate appropriate responses. This process was overseen by researchers to ensure accuracy and relevance. How can I assist you further?



Is this conversation helpful so far?

Message ChatGPT



ChatGPT can make mistakes. Check important info.

## BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

**Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova**  
Google AI Language

{jacobdevlin, mingweichang, kentonl, kristout}@google.com

Research

## RoBERTa: An optimized method for pretraining self-supervised NLP systems

7/29/2019

## Protein language models (PLMs)

**Evolutionary Scale Modeling (ESM)** protein language models developed by researchers at Facebook AI Research (FAIR) and designed specifically for understanding and predicting the structure and function of proteins based on their amino acid sequences.

**Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences**

Alexander Rives<sup>a,b,1,2</sup>, Joshua Meier<sup>a,1</sup>, Tom Sercu<sup>a,1</sup>, Siddharth Goyal<sup>a,1</sup>, Zeming Lin<sup>b</sup>, Jason Liu<sup>a</sup>, Demi Guo<sup>c,3</sup>, Myle Ott<sup>a</sup>, C. Lawrence Zitnick<sup>a</sup>, Jerry Ma<sup>d,e,3</sup>, and Rob Fergus<sup>b</sup>

<sup>a</sup>Facebook AI Research, New York, NY 10003; <sup>b</sup>Department of Computer Science, New York University, New York, NY 10012; <sup>c</sup>Harvard University, Cambridge, MA 02138; <sup>d</sup>Booth School of Business, University of Chicago, Chicago, IL 60637; and <sup>e</sup>Yale Law School, New Haven, CT 06511



# Protein language models (PLMs)

**Evolutionary Scale Modeling (ESM)** protein language models developed by researchers at Facebook AI Research (FAIR) and designed specifically for understanding and predicting the structure and function of proteins based on their amino acid sequences.

## Meta Lays Off Team Behind Revolutionary Protein-folding Model

Former scientist and other sources claim that Meta is shifting its focus to monetizing AI models



**Ben Wodecki, Jr. Editor**

August 8, 2023

⌚ 2 Min Read



Codebase: <https://github.com/facebookresearch/esm>

## Protein language models (PLMs)

**Evolutionary Scale Modeling (ESM)** protein language models developed by researchers at Facebook AI Research (FAIR) and designed specifically for understanding and predicting the structure and function of proteins based on their amino acid sequences.

The screenshot shows a news article from Forbes. The title is "Ex-Meta Researchers Have Raised \$40 Million From Lux Capital For An AI Biotech Startup". The article is by Ben Wood, published on August 8. It includes author bios for Kenrick Cai and Iain Martin, and a "2 Min Read" indicator. The background of the article area is black, while the rest of the page has a white header and sidebar.

Ex-Meta Researchers Have Raised \$40 Million From Lux Capital For An AI Biotech Startup

Former scientist at Meta's AI lab

Ben Wood

August 8, 2023

Kenrick Cai Forbes Staff

I'm a staff writer covering venture capital, startups and AI.

Iain Martin Forbes Staff

I'm a senior editor at Forbes and cover tech and venture capital.

2 Min Read



Codebase: <https://github.com/facebookresearch/esm>

# Protein language models (PLMs)

## Main models you should use

Shorthand	esm.pretrained.	Dataset	Description
ESM-2	esm2_t36_3B_UR500() esm2_t48_15B_UR500()	UR50 (sample UR90)	SOTA general-purpose protein language model. Can be used to predict structure, function and other protein properties directly from individual sequences. Released with <a href="#">Lin et al. 2022</a> (Aug 2022 update).
ESMFold	esmfold_v1()	PDB + UR50	End-to-end single sequence 3D structure predictor (Nov 2022 update).
ESM-MSA-1b	esm_msa1b_t12_100M_UR505()	UR50 + MSA	MSA Transformer language model. Can be used to extract embeddings from an MSA. Enables SOTA inference of structure. Released with <a href="#">Rao et al. 2021</a> (ICML'21 version, June 2021).
ESM-Tv	esm1v_t33_650M_UR905_1() ... esm1v_t33_650M_UR905_5()	UR90	Language model specialized for prediction of variant effects. Enables SOTA zero-shot prediction of the functional effects of sequence variations. Same architecture as ESM-1b, but trained on UniRef90. Released with <a href="#">Meier et al. 2021</a> .
ESM-IF1	esm_if1_gvp4_t16_142M_UR50()	CATH + UR50	Inverse folding model. Can be used to design sequences for given structures, or to predict functional effects of sequence variation for given structures. Enables SOTA fixed backbone sequence design. Released with <a href="#">Hsu et al. 2022</a> .

For a complete list of available models, with details and release notes, see [Pre-trained Models](#).



Codebase: <https://github.com/facebookresearch/esm>

# Protein language models (PLMs) - ESM2

RESEARCH

STRUCTURE PREDICTION

## Evolutionary-scale prediction of atomic-level protein structure with a language model

Zeming Lin<sup>1,2†</sup>, Halil Akin<sup>1†</sup>, Roshan Rao<sup>1†</sup>, Brian Hie<sup>1,3†</sup>, Zhongkai Zhu<sup>1</sup>, Wenting Lu<sup>1</sup>, Nikita Smetanin<sup>1</sup>, Robert Verkui<sup>1</sup>, Ori Kabeli<sup>1</sup>, Yaniv Shmueli<sup>1</sup>, Allan dos Santos Costa<sup>4</sup>, Maryam Fazel-Zarandi<sup>1</sup>, Tom Sercu<sup>1</sup>, Salvatore Candido<sup>1</sup>, Alexander Rives<sup>1,2\*</sup>

Direct inference of full atomic-level protein structure from primary sequence using a large language model.

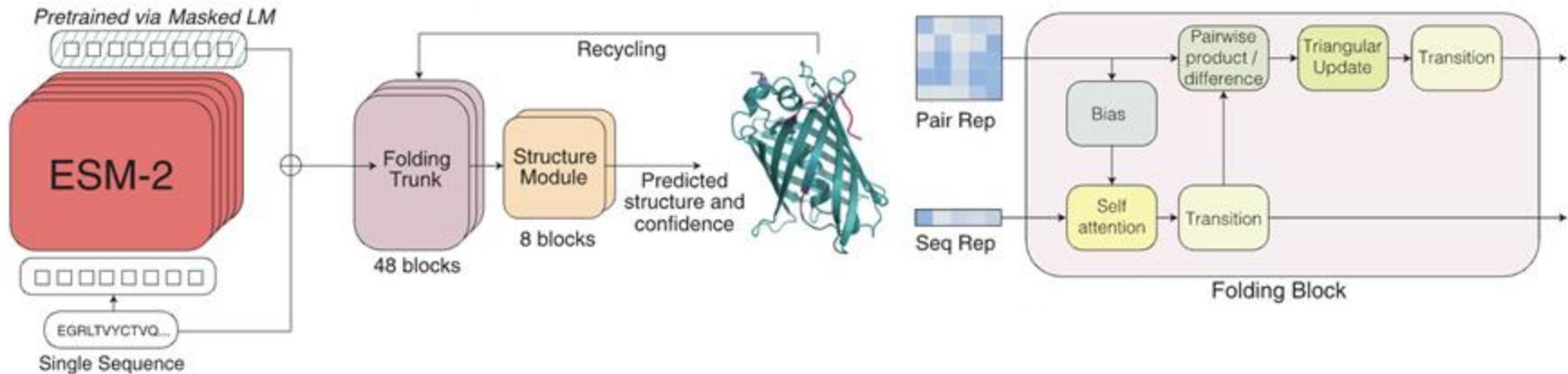
15B parameters



Codebase: <https://github.com/facebookresearch/esm>

# Protein language models (PLMs) - ESM2

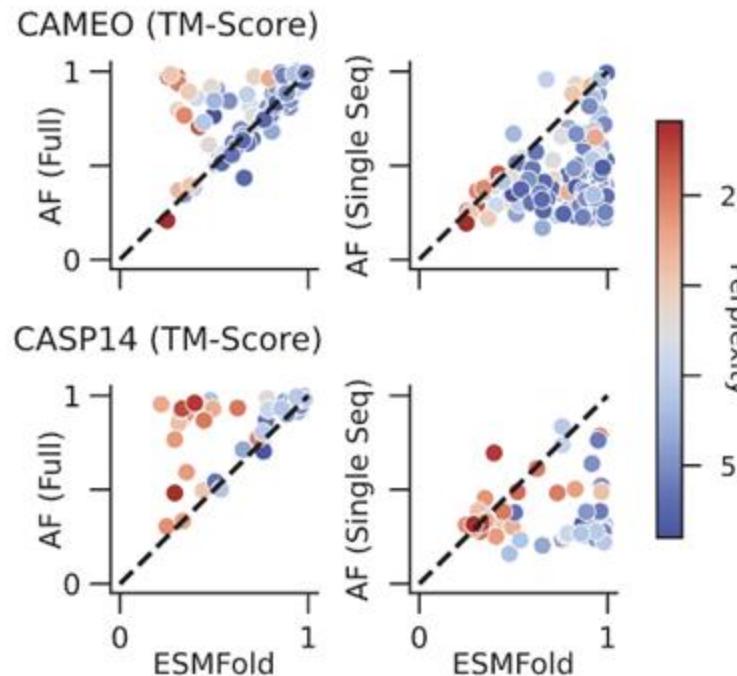
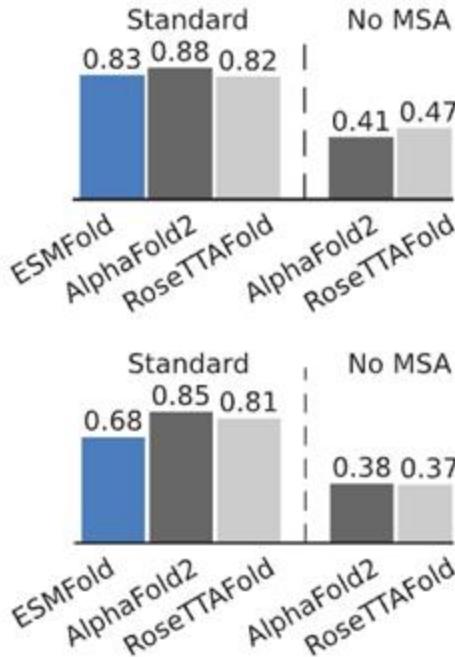
A



ESMFold model architecture. Arrows show the information flow in the network from the language model to the folding trunk to the structure module that outputs 3D coordinates and confidences. LM, language model.

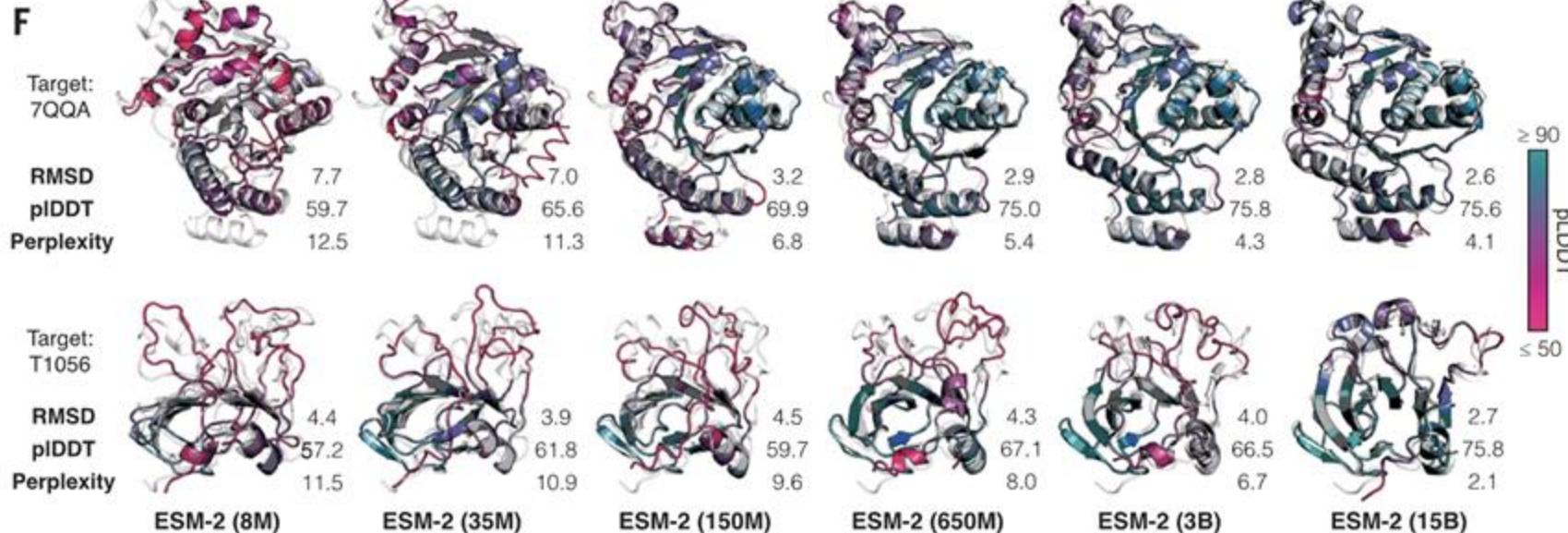
## Protein language models (PLMs) - ESM2

B



ESMFold produces accurate atomic resolution predictions, with similar accuracy to RoseTTAFold on CAMEO. When MSAs are ablated for AlphaFold and RoseTTAFold, performance of the models degrades. Scatter plots compare ESMFold (x axis) predictions with AlphaFold2 (y axis), colored by language model perplexity. Proteins with low perplexity score similarly to AlphaFold2. AF, AlphaFold2.

## Protein language models (PLMs) - ESM2

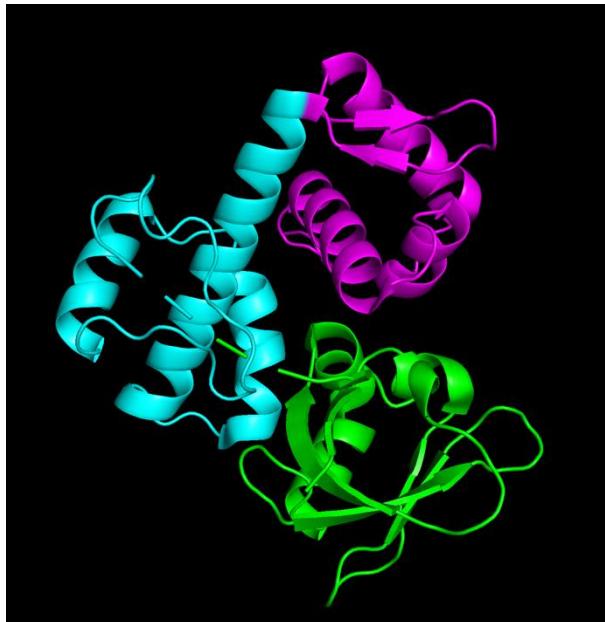


Structure predictions on CAMEO structure 7QQA and CASP target 1056 at all ESM-2 model scales, colored by pLDDT (pink, low; teal, high). For 7QQA, prediction accuracy improves at the 150-million-parameter threshold. For T1056, prediction accuracy improves at the 15-billion-parameter threshold.

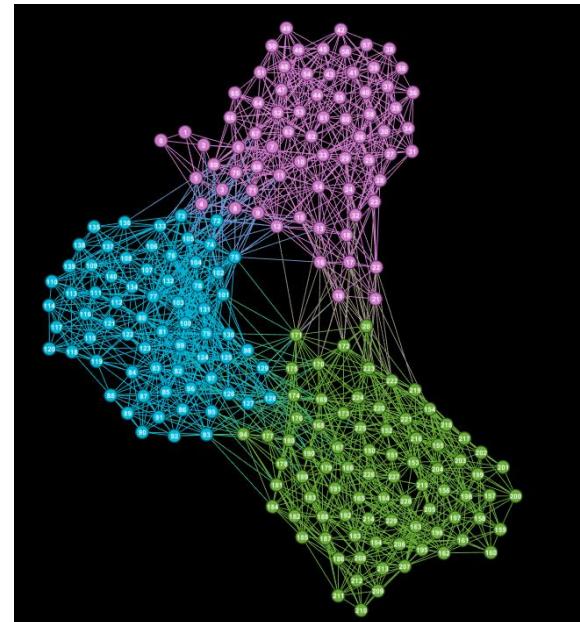
# GraphNN

---

## Example PDB: 1U8R-B



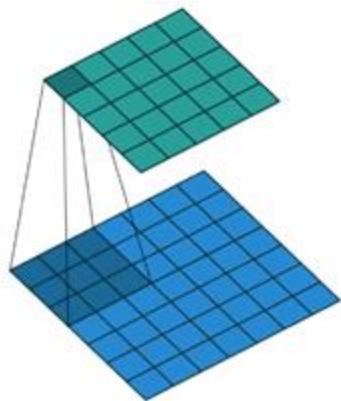
$C\alpha-C\alpha$   
 $< 10\text{\AA}$



```
>>MNELVDTTEMYLRTIYDLEEEGVTPLRARIOERLDQSGPTVSQTVSRMERDGLLRVAGDRHLELTEKGRALAIAMVRKHRLAERLLVDVIGLPWEVVAEACRWEHVMSEDVERRLVKLNNPTTSPFGNP  
IPGLDELGVGPEPGADDANLVRTELPGSPAVVVRLTEHVQGDIDLTTRKDAGVVPNARVTETPPGGGVIVIPGHENVTLPHEMAHAVKVEKV
```

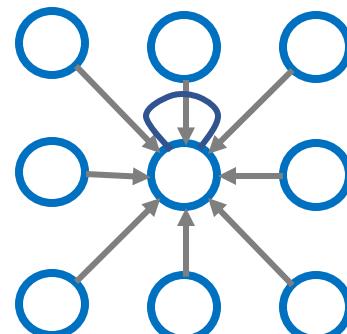
# Geometric deep learning: CNN for graphs

**CNN**

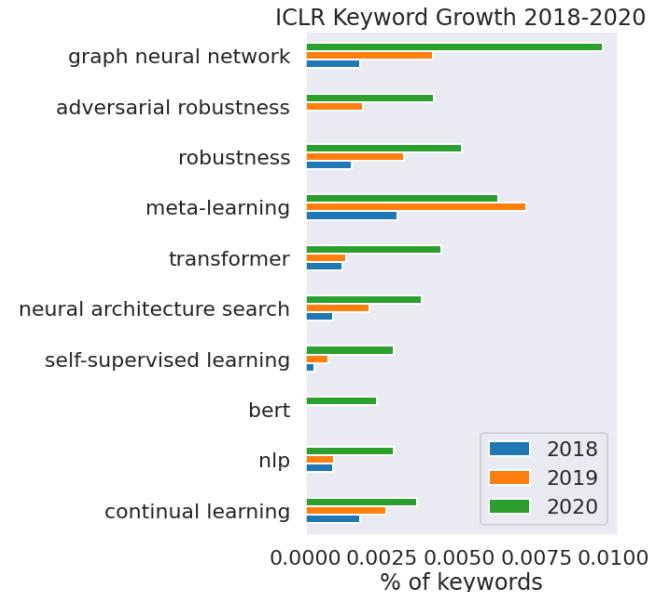


grid-like data structure

**GraphNN**



graph-like data structure



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

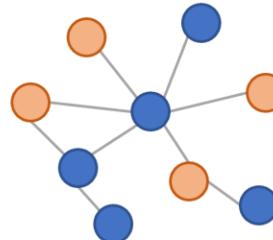
Bronstein MM, Bruna J, LeCun Y, Szlam A, Vandergheynst P. (2017) Geometric deep learning: going beyond euclidean data. IEEE Signal Processing Magazine.

# GraphNNs: types and applications

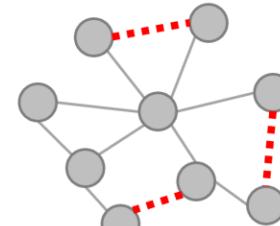
## Types of GNNs:

- Spectral formulation
- Spatial formulation

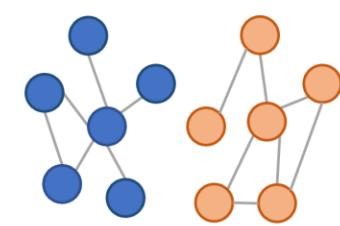
**Node Classification**



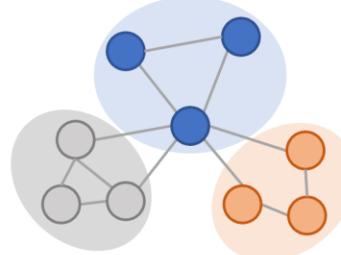
**Link Prediction**



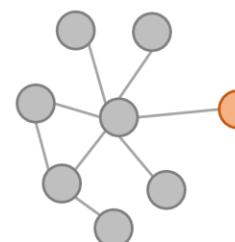
**Graph Classification**



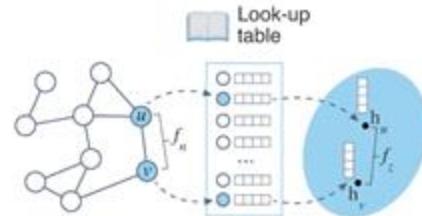
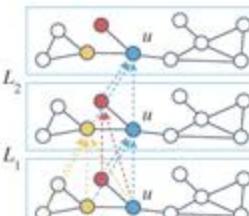
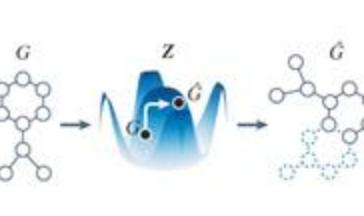
**Community Detection**



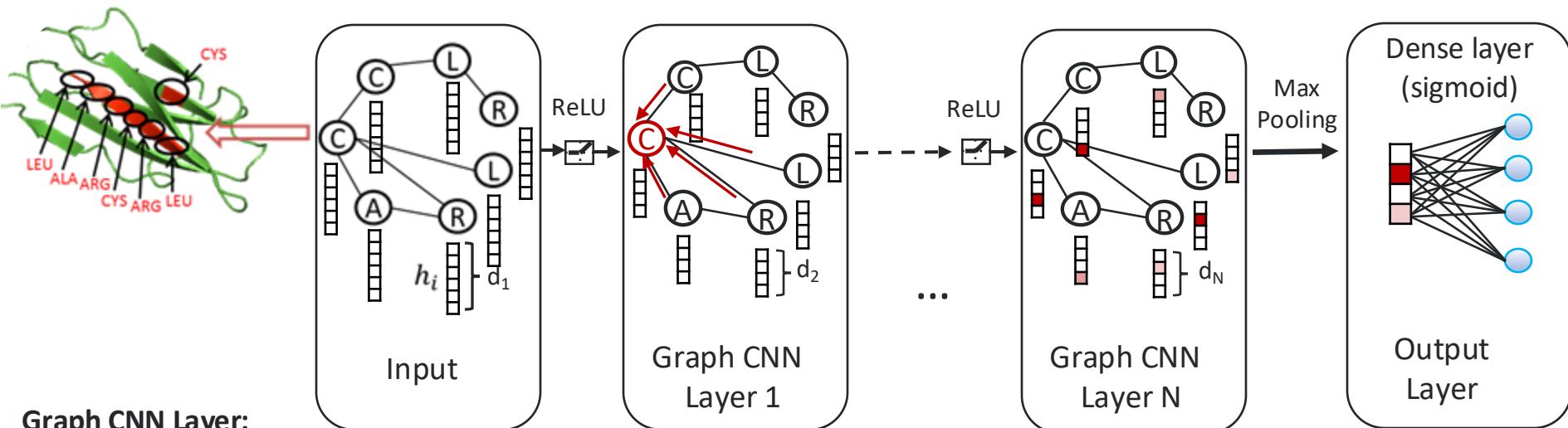
**Anomaly Detection**



# History of the graph-based methods

	a Shallow network embeddings	b Graph neural networks	c Generative graph models
<b>Approach</b>			
<b>Input</b>	Graph structure	Local graph neighbourhoods Node and edge attributes	Graph structure Node and edge attributes Graph attributes
<b>Representation output</b>	Node embeddings Edge embeddings	Node embeddings Edge embeddings Subgraph embeddings Graph embeddings	Graph structure
<b>Learning task</b>	Node-property prediction Link-property prediction	Node-property prediction Link-property prediction Graph-property prediction	Molecular graph generation Molecule optimization
<b>Examples of implementation methods</b>	DeepWalk, Node2vec, LINE, Metapath2vec	GCN, GIN, GAT, JK-Net	GCPN, JT-VAE, GraphRNN

# An example of a GraphNN...



**A** – adjacency matrix representing contact map

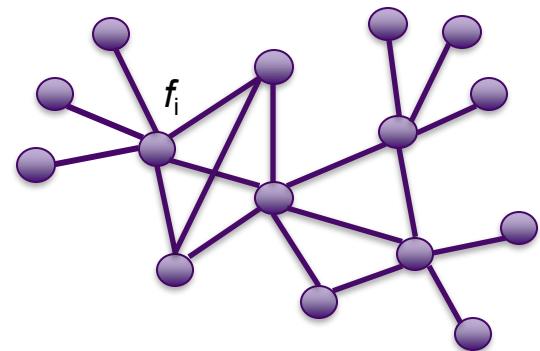
**D** – diagonal degree matrix

**H<sup>(0)</sup>** – input feature matrix (**H<sup>(0)</sup>**=**I**, 1-hot encoding if features not available)

**W<sup>(l)</sup>** – weight matrix for layer l

# Brief intro to Graph Theory

- Undirected graph:  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , with **vertices**  $\mathcal{V} = \{1, \dots, n\}$  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:  $L^2(\mathcal{V}) = \{f : \mathcal{V} \rightarrow \mathbb{R}\}$  represented as vectors:  $\mathbf{f} = (f_1, \dots, f_n) \in \mathbb{R}^n$
- Inner product operation:  $\langle \mathbf{f}, \mathbf{g} \rangle_{L^2(V)} = \sum_{i \in \mathcal{V}} f_i g_i = \mathbf{f}^T \mathbf{g}$

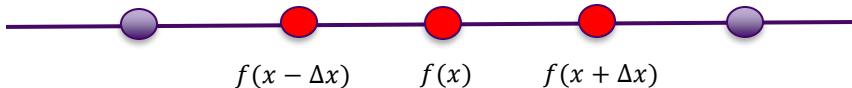


# Laplace operator

- Euclidean

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

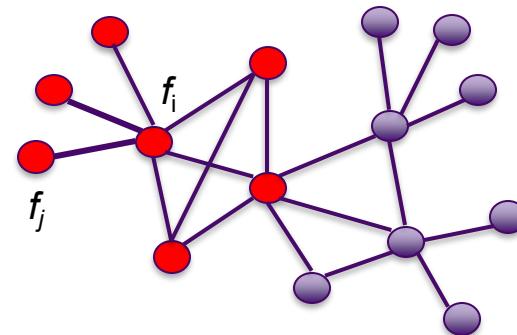
$$\Delta f \propto [f(x) - f(x + \Delta x)] + [f(x) - f(x - \Delta x)]$$



- Non-Euclidean

- $\Delta : L^2(\mathcal{V}) \rightarrow L^2(\mathcal{V})$
- On graphs:  $(\Delta f)_i = \sum_{j:(i,j) \in \mathcal{E}} a_{ij} (f_i - f_j)$
- Matrix representation:  

$$\Delta = D - A$$
, where  $D = \text{diag}(\sum_{j \neq i} a_{ij})$



# 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

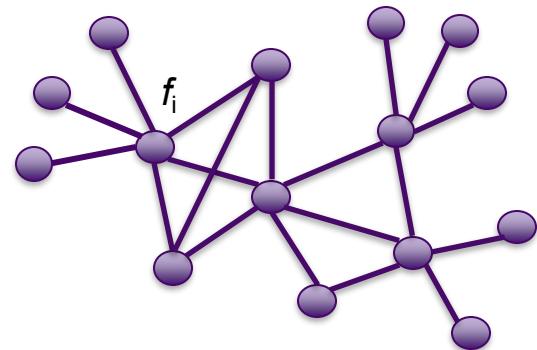
- Non-Euclidean

- Normalized Laplacian:  $\hat{\Delta} = \mathbf{D}^{-\frac{1}{2}} \Delta \mathbf{D}^{-\frac{1}{2}} = \mathbf{I}_n - \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$
- Eigendecomposition:  $\hat{\Delta} \phi_i = \lambda_i \phi_i$ ;
  - Non-negative **eigenvalues**:  $0 \leq \lambda_1 \leq \lambda_2 \dots \leq \lambda_n \leq 2$ ;  
Matrix form:  $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$
  - Corresponding orthogonal **eigenvectors**:  $\langle \phi_i, \phi_j \rangle = \delta_{ij}$ ; Matrix form:  $\Phi = (\phi_1, \dots, \phi_n)$
- Eigendecomposition in matrix form:  $\hat{\Delta} = \Phi \Lambda \Phi^T$
- Fourier transform:  $\hat{f}(\lambda_l) = \langle f, \phi_l \rangle = \sum_{i=1}^n f_i \phi_{li}^*$ ;  
Matrix-vector form:  $\hat{\mathbf{f}} = \Phi^T \mathbf{f}$
- Inverse transform:  $f_i = \langle f, \phi_l^* \rangle = \sum_{l=1}^n \hat{f}(\lambda_l) \phi_{li}$ ;  
Matrix-vector form:  $\mathbf{f} = \Phi \hat{\mathbf{f}}$

# 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)  $\Rightarrow$  compute convolution in the Fourier domain
- Convolution theorem:  $\widehat{(f * g)}(\xi) = \hat{f}(\xi) \circ \hat{g}(\xi)$   $\Rightarrow$  take inverse
- Convolution of signals on a graph (in matrix-vector notation):  $\mathbf{f} * \mathbf{w} = \mathcal{F}^{-1}\{\hat{\mathbf{f}} \circ \hat{\mathbf{w}}\} = \Phi(\hat{\mathbf{w}} \circ \Phi^T \mathbf{f})$
- $\mathbf{f} * \mathbf{w} = \Phi diag(\hat{w}_1, \dots, \hat{w}_n) \Phi^T \mathbf{f}$

**Filter coefficients depend on Laplacian eigenvectors!**



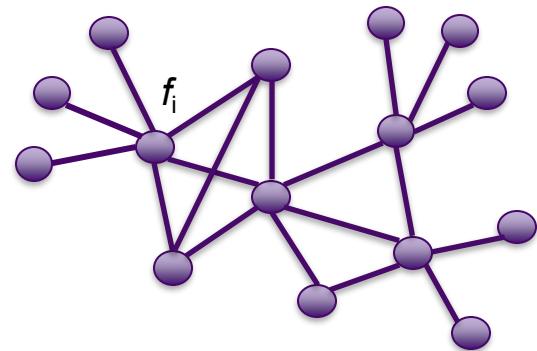
# Spectral GraphConv layer

Convolution in spectral domain:

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

where  $\mathbf{W}$  is  $n \times n$  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 😞

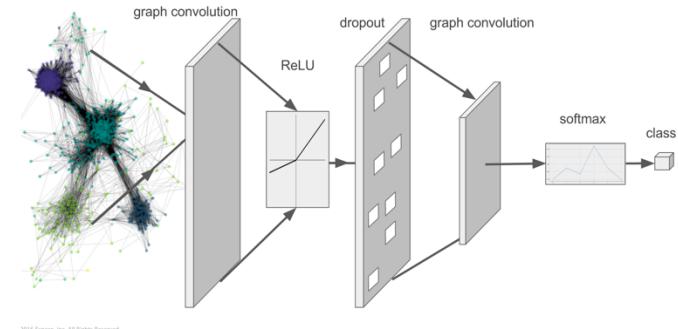


# GraphConv with polynomial filters

- Parametrize the filter using a smooth function:  
 $\hat{g}_{\mathbf{w}}(\lambda) = \sum_{k=1}^K w_k \lambda^k$
- Spectral filters represented by  $K^{th}$  order polynomials,  
 $\hat{g}_{\mathbf{w}}(\Lambda) = \sum_{k=0}^K w_k \Lambda^k$ , are **exactly  $K$ -localized**, where  
 $\mathbf{w} = (w_1, \dots, w_K)$  is the vector of filter parameters
- Application of the filter:  $\mathbf{h} = \Phi \hat{g}_{\mathbf{w}}(\Lambda) \Phi^T \mathbf{f} = \hat{g}_{\mathbf{w}}(\hat{\Delta}) \mathbf{f}$
- Number of parameters  $K$  (fixed) is independent of graph size  $n$  (same learning complexity as regular CNNs)
- No explicit computation of  $\Phi$ ,  $\Phi^T$  because  $\mathbf{h} = \hat{g}_{\mathbf{w}}(\hat{\Delta}) \mathbf{f}$   
 $\implies O(K|\mathcal{E}|)$  computational complexity

# Graph Convolutional Networks

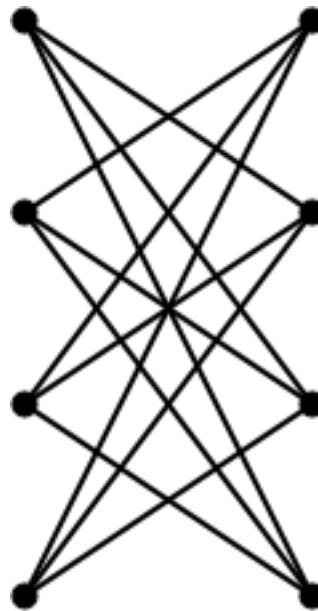
- First-order approximation ( $K = 1$ ) of spectral graph convolution
- $\mathbf{h} = g_w(\hat{\Delta})\mathbf{x} = \sum_{k=0}^1 w_k \hat{\Delta}^k \mathbf{x} = w(\mathbf{I}_n + \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}) \mathbf{x}$ , for signal  $\mathbf{x} \in \mathbb{R}^n$
- Renormalization 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



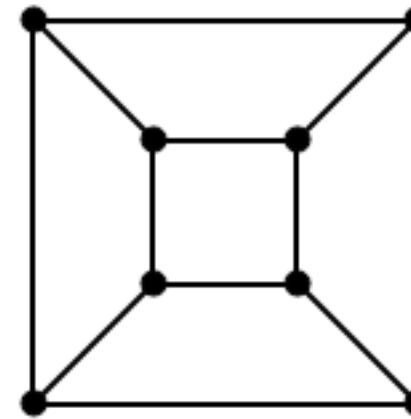
2016 Expara, Inc. All Rights Reserved

# Graph Isomorphism

---

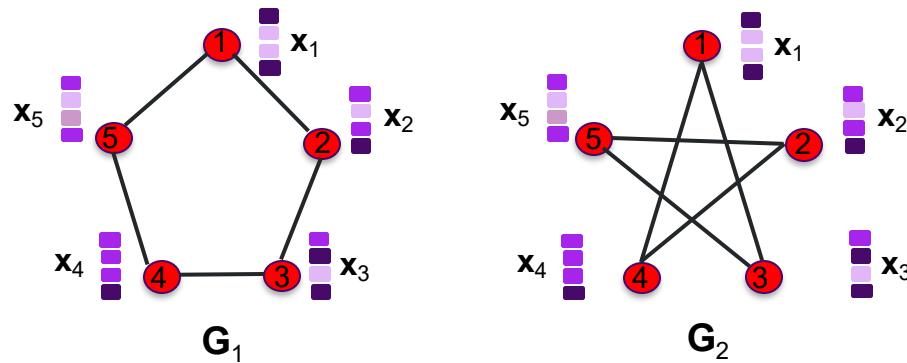


A



B

# Permutation invariance and equivariance



- The order of the nodes in a graph is not fixed
- GraphNN should produce the same result for  $\mathbf{G}_1$  &  $\mathbf{G}_2$
- i.e., the result of any function/NN shouldn't depend on the node ordering

**Node feature matrix:**

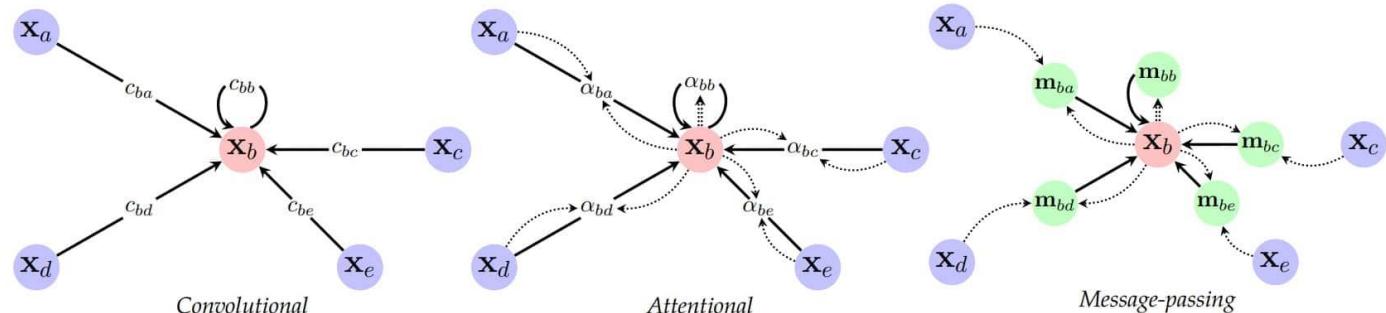
$$\mathbf{X} = \begin{bmatrix} \text{x}_1 \\ \text{x}_2 \\ \text{x}_3 \\ \text{x}_4 \\ \text{x}_5 \end{bmatrix} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5)^T$$

**Invariance:**  $f(\mathbf{P}\mathbf{X}, \mathbf{P}\mathbf{A}\mathbf{P}^T) = f(\mathbf{X}, \mathbf{A})$

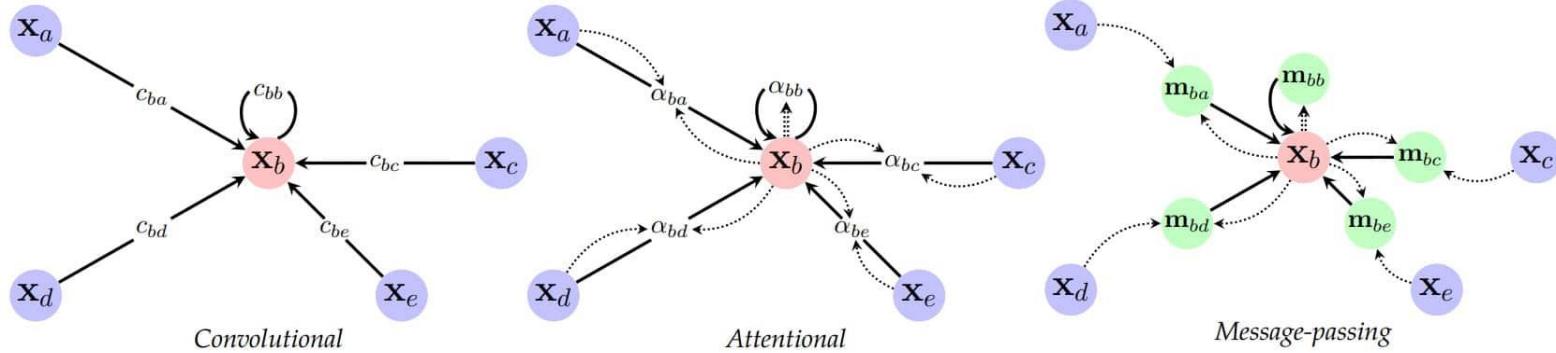
**Equivariance:**  $f(\mathbf{P}\mathbf{X}, \mathbf{P}\mathbf{A}\mathbf{P}^T) = \mathbf{P}f(\mathbf{X}, \mathbf{A})$

## GNN 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  
 $a_v^{(k)} = MEAN$



# GNN variants



$$h_i^{(l+1)} = \phi\left(h_i^{(l)}, \bigoplus_{j \in \mathcal{N}(i)} c_{ij} \psi(h_j^{(l)})\right)$$

$$h_i^{(l+1)} = \phi\left(h_i^{(l)}, \bigoplus_{j \in \mathcal{N}(i)} \alpha(h_i^{(l)}, h_j^{(l)}) \psi(h_j^{(l)})\right)$$

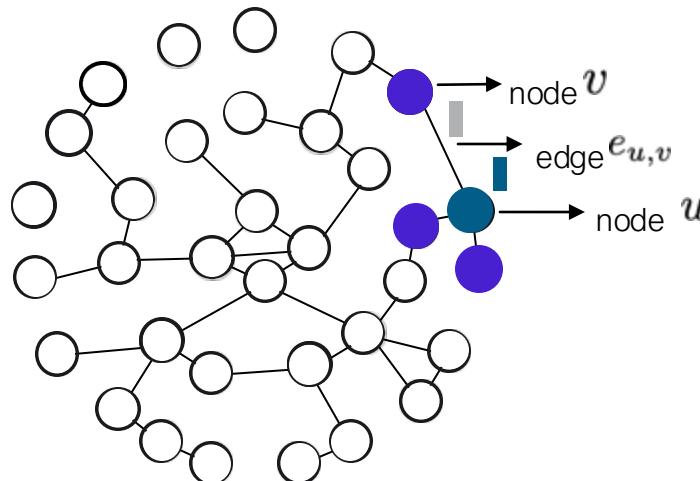
$$h_i^{(l+1)} = \phi\left(h_i^{(l)}, \bigoplus_{j \in \mathcal{N}(i)} \psi(h_i^{(l)}, h_j^{(l)})\right)$$

# Some graphs concepts

Graph  $G = (\mathcal{V}, \mathcal{E})$

$\mathcal{V}$  set of nodes

$\mathcal{E}$  set of edges



node feature  $\mathbf{x}_u \in \mathbb{R}^d$

1.5	26	28	105	61.3	5.8
-----	----	----	-----	------	-----

$$d = 6$$

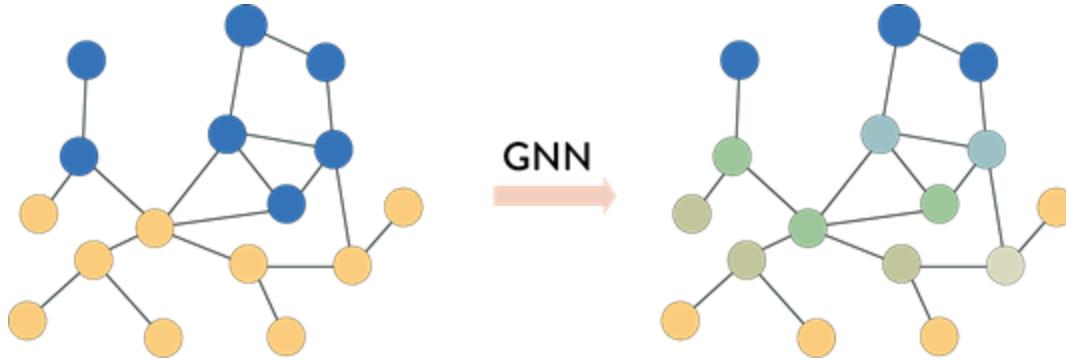
edge feature  $\mathbf{x}_{u,v} \in \mathbb{R}^c$

5.3	26	105	32.1
-----	----	-----	------

$$c = 4$$

local neighborhood  $\mathcal{N}(u)$

# Prevailing GNN paradigm | Message passing

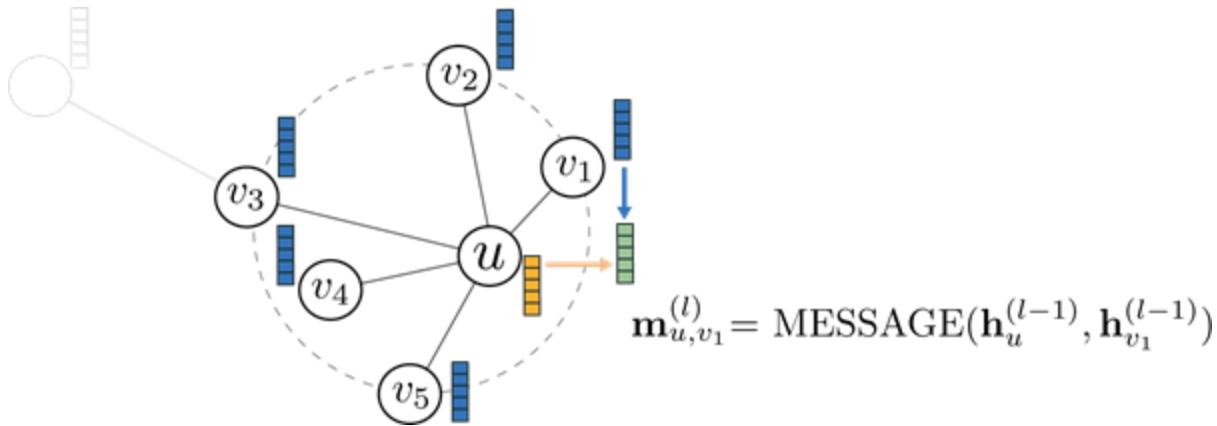


$$\mathbf{m}_{u,v_i}^{(l)} = \text{MESSAGE}(\mathbf{h}_u^{(l-1)}, \mathbf{h}_{v_i}^{(l-1)})$$

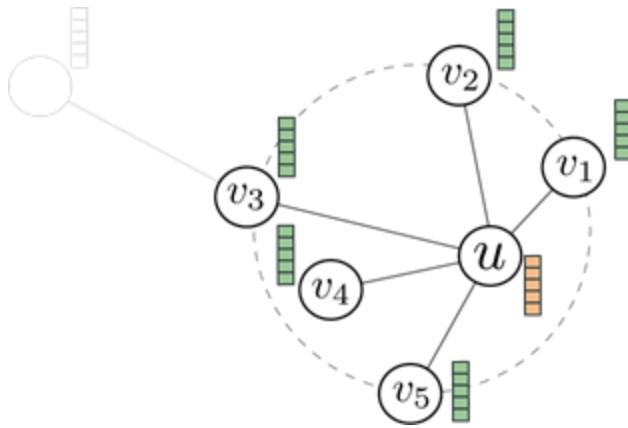
$$\mathbf{m}_u^{(l)} = \text{AGGREGATE}(\mathbf{m}_{u,v_i}^{(l)} \mid v_i \in \mathcal{N}(u))$$

$$\mathbf{h}_u^{(l)} = \text{UPDATE}(\mathbf{m}_u^{(l)}, \mathbf{h}_u^{(l-1)})$$

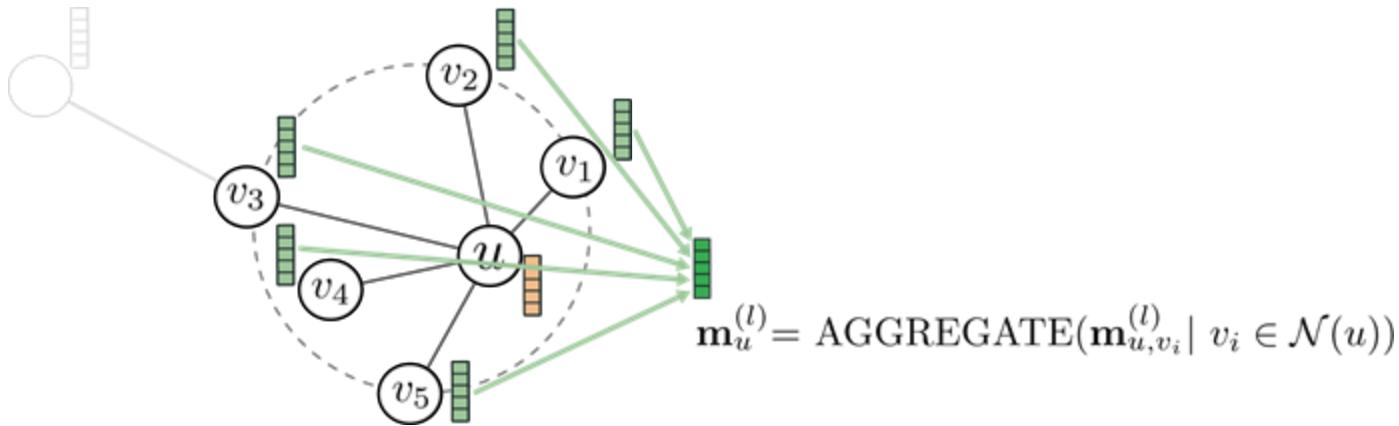
# Prevailing GNN paradigm | Message passing



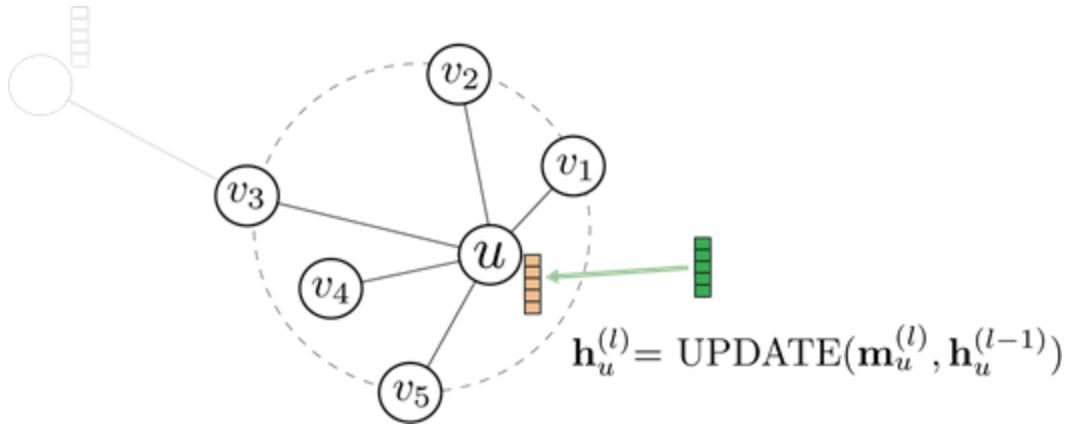
# Prevailing GNN paradigm | Message passing



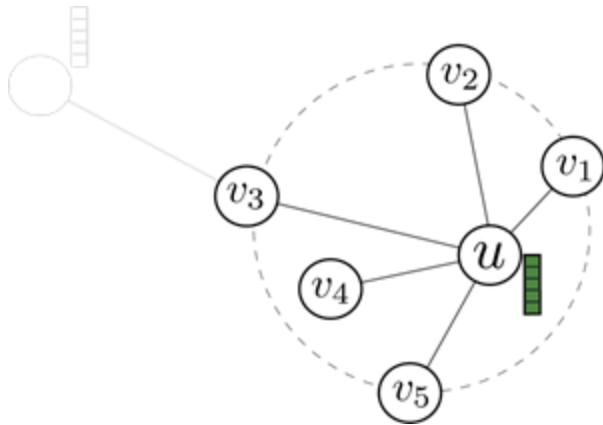
# Prevailing GNN paradigm | Message passing



# Prevailing GNN paradigm | Message passing



# Prevailing GNN paradigm | Message passing

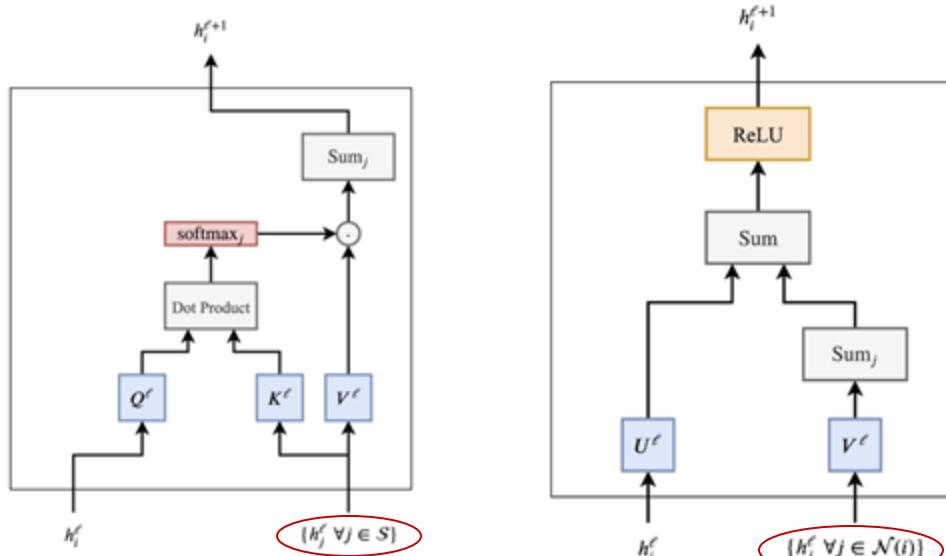


# Graph transformers

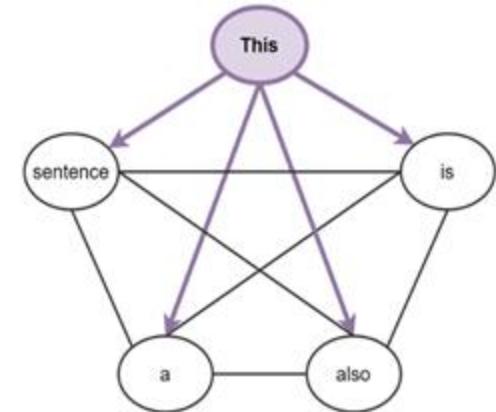
Chaitanya Joshi

Feb 12, 2020

## Transformers are Graph Neural Networks



Sentences are fully-connected word graphs!



# Graph transformers

Transformers are Graph Neural

|

C  
F

## A Generalization of Transformer Networks to Graphs

Vijay Prakash Dwivedi,<sup>¶</sup> Xavier Bresson<sup>¶</sup>

<sup>¶</sup> School of Computer Science and Engineering, Nanyang Technological University, Singapore  
vijaypra001@e.ntu.edu.sg, xbresson@ntu.edu.sg

**Graph Transformer Network:** Develops a theoretical framework for generalizing the transformer model to graph data, addressing how **self-attention and positional encodings** can be adapted for graphs.

# Graph transformers

Transformers are Graph Neural

|

C  
F

## A Generalization of Transformer Networks to G

Sch

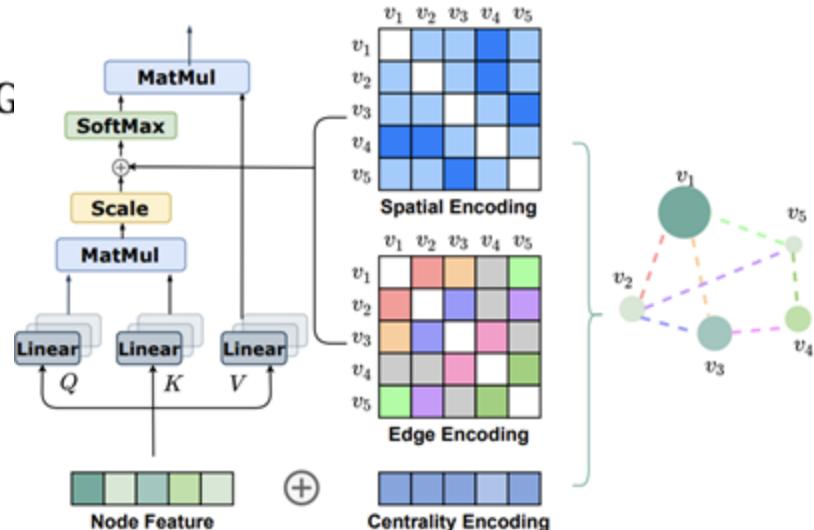
### Do Transformers Really Perform Bad for Graph Representation?

Chengxuan Ying<sup>1,\*</sup>, Tianle Cai<sup>2</sup>, Shengjie Luo<sup>3\*</sup>,  
Shuxin Zheng<sup>4†</sup>, Guolin Ke<sup>4</sup>, Di He<sup>4‡</sup>, Yanming Shen<sup>1</sup>, Tie-Yan Liu<sup>4</sup>

<sup>1</sup>Dalian University of Technology    <sup>2</sup>Princeton University

<sup>3</sup>Peking University    <sup>4</sup>Microsoft Research Asia

yingchengsyuan@gmail.com, tianle.cai@princeton.edu, luosj@stu.pku.edu.cn  
{shuz, guoke, dihe, tyliu}@microsoft.com, shen@dlut.edu.cn



**Graphomer** introduces **spatial encoding** to capture graph structure, **edge encoding** to incorporate edge features into the attention mechanism, and **centrality encoding** to enhance node importance representation.

## Limitations of GNNs

---

- To capture long range information, we need GNNs with sufficient depth
- Current GNNs suffer from performance degradation at higher depth, which limits their application
- This degradation happens because:
  - Over-smoothing (nodes' features become similar to each other as # layers  $\rightarrow \inf$  )
  - Over-squashing (graph bottlenecks cause messages between distant nodes to become overly-squashed)

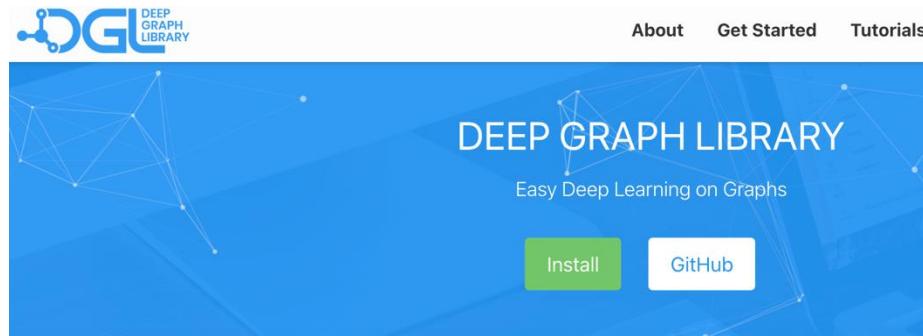
# Resources for implementing GNNs



PyG Documentation

## PyG Documentation

PyG (PyTorch Geometric) is a library built upon PyTorch to easily write and train Graph Neural Networks (GNNs) for a wide range of applications related to structured data.



About   Get Started   Tutorials

## DEEP GRAPH LIBRARY

Easy Deep Learning on Graphs

Install

GitHub

Name	SparseTensor	edge_weight	edge_attr	bipartite	static	lazy
<a href="#">SimpleConv</a> ( <a href="#">Paper</a> )	✓	✓		✓	✓	
<a href="#">GCNConv</a> ( <a href="#">Paper</a> )	✓	✓			✓	✓
<a href="#">ChebConv</a> ( <a href="#">Paper</a> )		✓			✓	✓
<a href="#">SAGEConv</a> ( <a href="#">Paper</a> )	✓			✓	✓	✓
<a href="#">CuGraphSAGEConv</a> ( <a href="#">Paper</a> )					✓	
<a href="#">GraphConv</a> ( <a href="#">Paper</a> )	✓	✓		✓	✓	✓
<a href="#">GatedGraphConv</a> ( <a href="#">Paper</a> )	✓	✓			✓	
<a href="#">ResGatedGraphConv</a> ( <a href="#">Paper</a> )	✓		✓	✓	✓	✓
<a href="#">GATConv</a> ( <a href="#">Paper</a> )	✓		✓	✓		
<a href="#">CuGraphGATConv</a> ( <a href="#">Paper</a> )					✓	
<a href="#">FusedGATConv</a> ( <a href="#">Paper</a> )					✓	
<a href="#">GATv2Conv</a> ( <a href="#">Paper</a> )	✓		✓	✓		

# Reading list

- Li, M.M., Huang, K. & Zitnik, M. Graph representation learning in biomedicine and healthcare. *Nat. Biomed. Eng* 6, 1353–1369 (2022). URL: <https://www.nature.com/articles/s41551-022-00942-x>
- Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A Comprehensive Survey on Graph Neural Networks. 1 2019. URL: <http://arxiv.org/abs/1901.00596>
- Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. 9 2016. URL: <http://arxiv.org/abs/1609.02907>
- Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. 10 2017. URL: <http://arxiv.org/abs/1710.10903>
- William L Hamilton, Rex Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. URL: <https://arxiv.org/abs/1706.02216>
- Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling Relational Data with Graph Convolutional Networks. 3 2017. URL: <http://arxiv.org/abs/1703.06103>.

## Related reading

- Review on current geometric ML approaches for structure-based small molecule design:  
<https://arxiv.org/pdf/2306.11768>
- RNNs explained: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- Great explanation (intuitive and with equations) of transformers and their connection with GNNs:  
<https://graphdeeplearning.github.io/post/transformers-are-gnns/>
- Introduction to graph representation learning and GNNs: <https://prescient.gene.com/blog/2023-08-16-graph-representation-learning/>



**Prescient**  
Design

A Genentech Accelerator



# Generative AI

---

## In Drug Design

**Genentech**  
*A Member of the Roche Group*

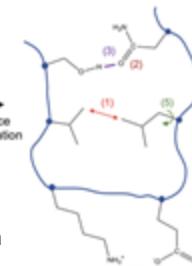
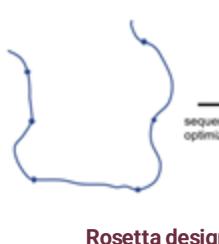
# Computational protein design/optimization problem

**Key challenge:** Searching through a *prohibitively large* combinatorial discrete space,  $\mathcal{S}$ , for a solution that max scoring function,  $f(s)$

$$\operatorname{argmax}_{s \in \mathcal{S}} f(s)$$

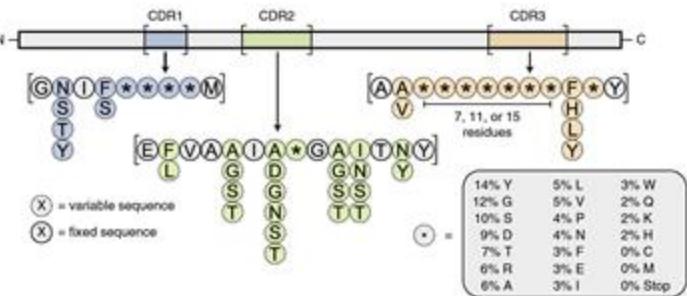
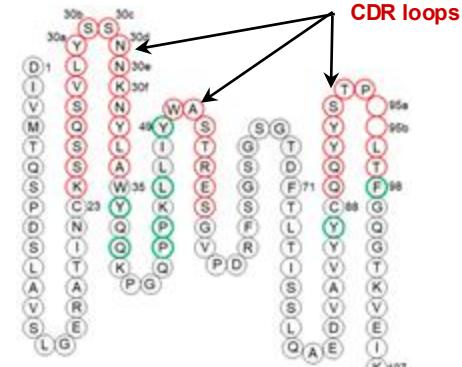
**Problem:**  $\mathcal{S}$  is sparsely functional

- Rational design (David Baker) and directed evolution (Frances Arnold)



66

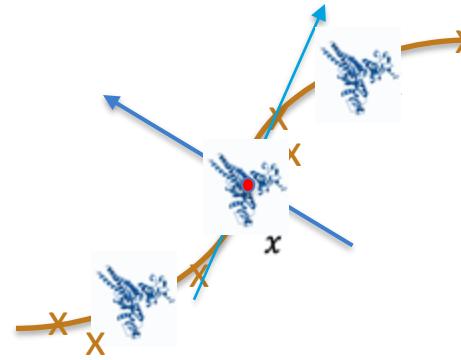
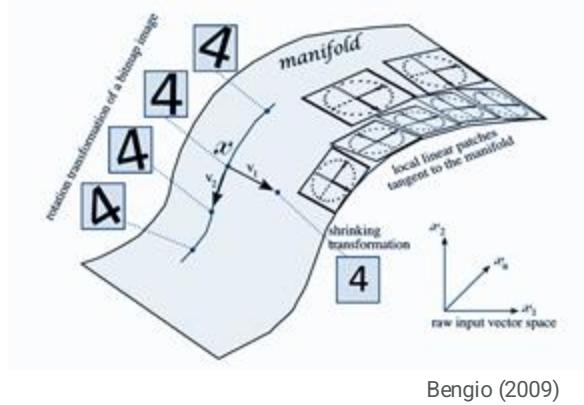
Kuhlman (2019), JBC



McMahon, C., ... & Kruse, A. C. (2018), *Nature structural & molecular biology*.

# The Curse of Dimensionality and the Manifold Assumption

Real-world, high-dimensional data lie (roughly) on a low-dimensional manifold (Chapelle 2006)



Instead of the (**large**) combinatorial space, search for a novel sequence  $\boldsymbol{x}$  is restricted on the (**small**) manifold space, i.e.,  $\boldsymbol{x} \in \mathcal{M}$

**Function-guided design:**  $\underset{\mathbf{x} \in \mathcal{M}}{\operatorname{argmax}} f(\mathbf{x})$  for a scoring function  $f(\mathbf{x})$  trained on  $\mathcal{M}$  where  $f(\mathbf{x})$  is an oracle.

# Learning and Sampling from the Data Manifold: Denoising Autoencoders (DAE)

Vincent et al. (2010) introduced DAEs:

$$\min_{\varphi, \theta} \frac{1}{N} \sum_{t=1}^N \|x - P_\varphi(Q_\theta(\tilde{x}_t))\|^2 \text{ where } \tilde{x}_t \sim C(\tilde{x} | x_t)$$

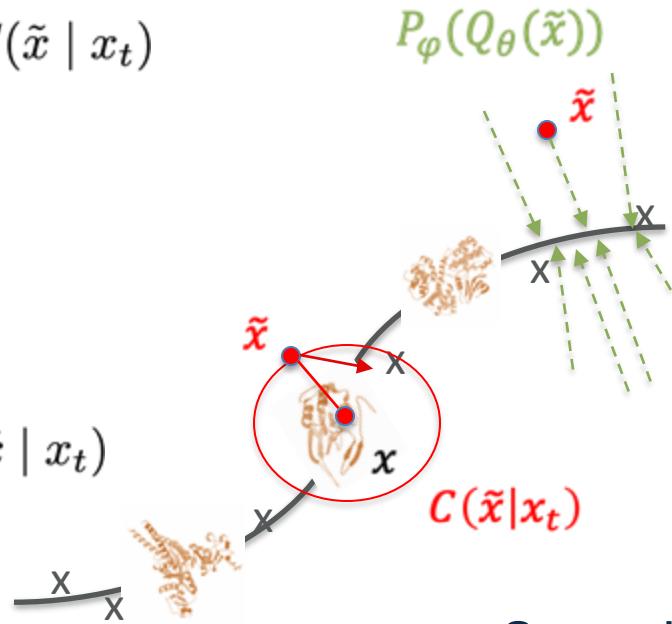
**Encoder**

**Decoder**

and  $C(\tilde{x} | x_t)$  is unstructured noise centered on  $x_t$ .

Then  $x' \sim P(x) \iff x' \sim P_\varphi(Q_\theta(\tilde{x}_t))$ , where  $\tilde{x}_t \sim C(\tilde{x} | x_t)$

Admits **Markov Chain Monte Carlo (MCMC)-like**  
iterative sampling procedure



# Deep Manifold Sampler

## Our model: architecture details

NeurIPS 2021 Machine Learning for Structural Biology -  
Oral Presentation

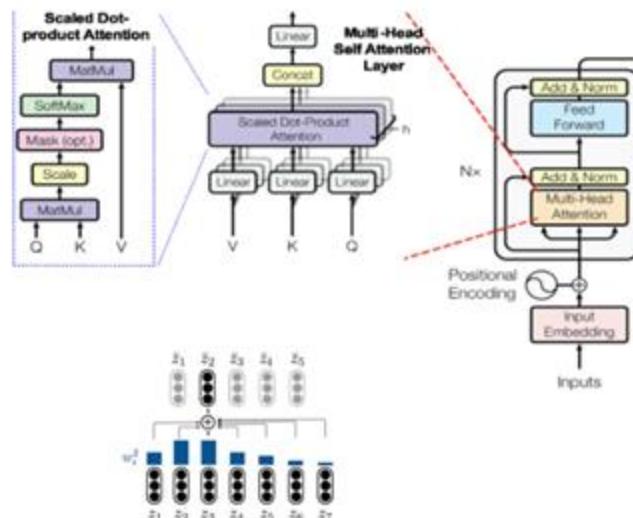
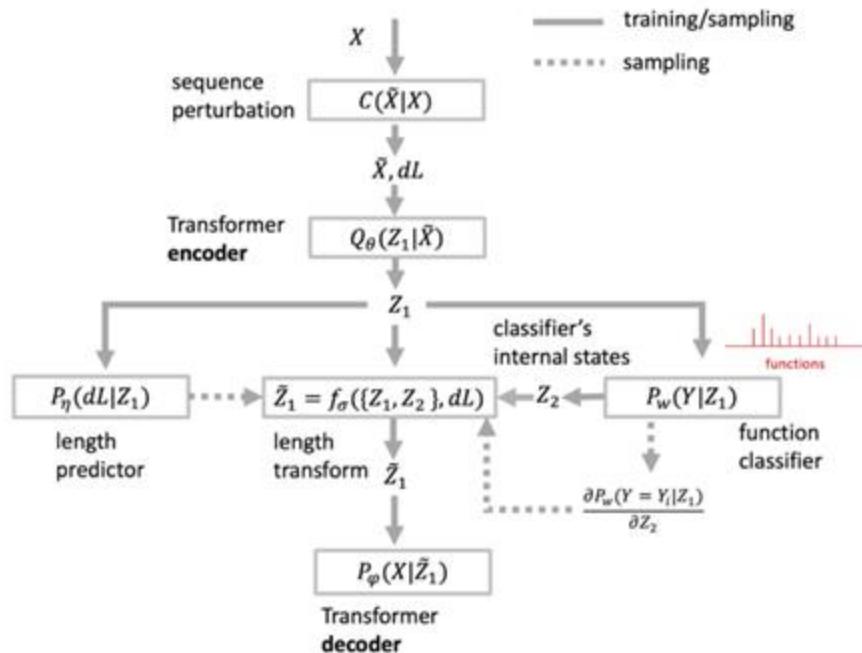


Figure 2: Illustration of the length transformation mecha-nism.

Shu, Raphael, ..K. Cho "Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34, No. 05, 2020.

# How Generative AI can help in Sequence/Structure Design

## Generative models in a nutshell.

Learn  $p(\mathbf{x}) = p(x_1, x_2, \dots, x_L)$   
 $p(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$

### Alignment-based models:

- Hidden Markov Models (HMM)
- Potts model

### Latent variable models:

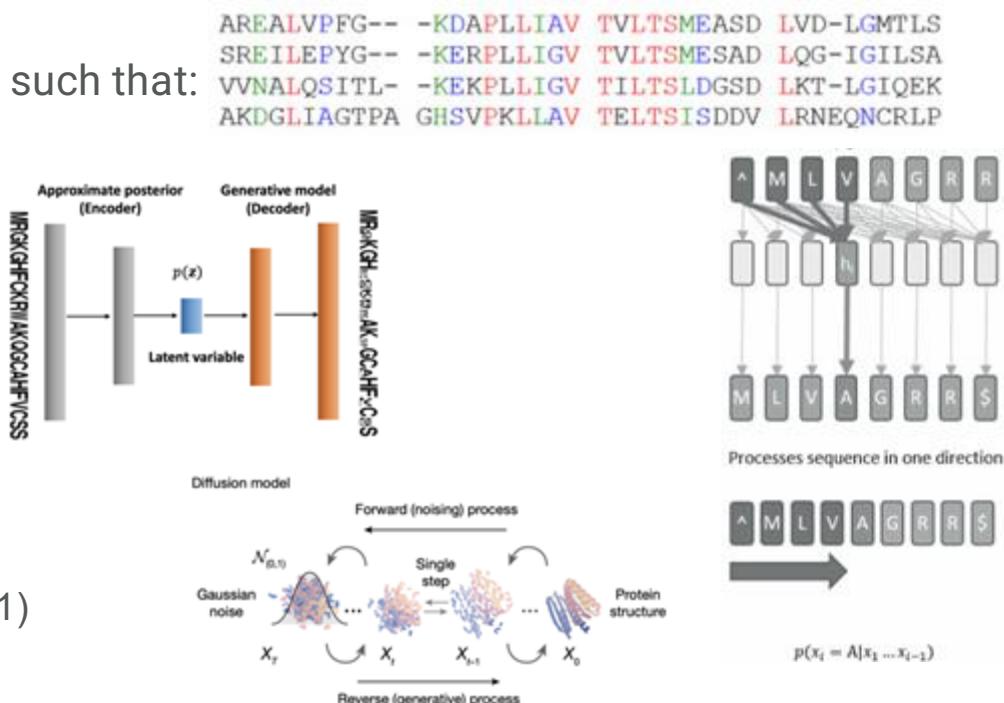
- GANs (e.g., Namrata *et al.*, 2018)
- VAEs (e.g., Riesselman *et al.*, 2018)

### Language models:

- ProGen (e.g., Madani *et al.*, 2020)
- Manifold Sampler (Gligorijevic, et al. 2021)

### Diffusion Models

- RFDiffusion (Chroma, 2023)
- WJS, AbDiffuser, (Prescient, 2023)



# Model: Overview

Given a protein (target) sequence  $Y = y_1, \dots, y_n$ , learn a Seq2Seq DAE model  $X \rightarrow Y$  where  $X$  is a corrupted version of  $Y$ :  $\mathbf{X} \sim C(\mathbf{X}|\mathbf{Y})$

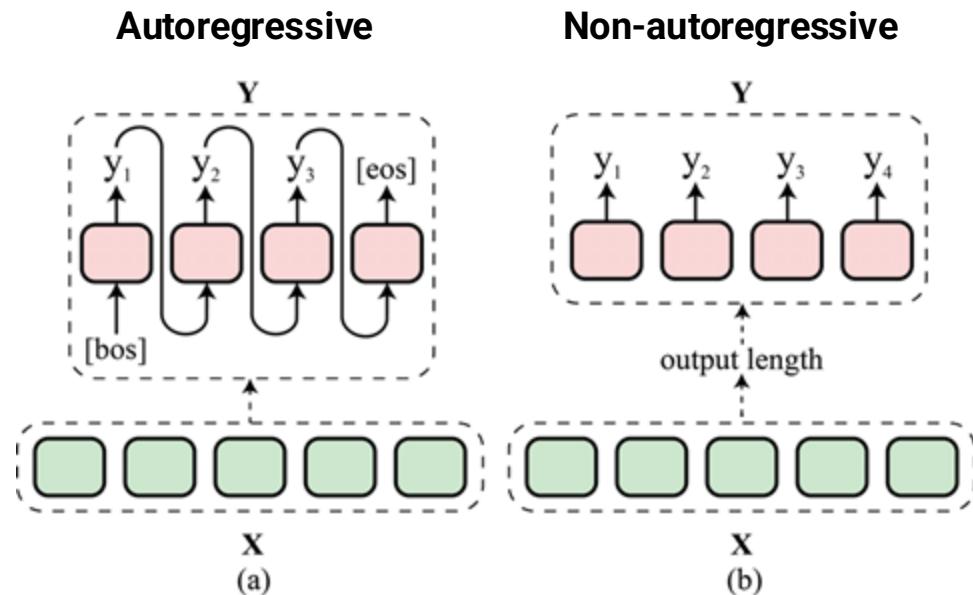
Two approaches:

1. Autoregressive Seq2Seq:

$$\log p(\mathbf{Y}|\mathbf{X}) = \sum \log p(y_t, y_{<t}, \mathbf{X})$$

1. Non-autoregressive Seq2Seq:

$$\log p(\mathbf{Y}|\mathbf{X}) = \sum_t \log p(y_t, \mathbf{X})$$



**Non-autoregressive sampling.** Makes changes in multiple positions of a target sequence enabling **effective** exploration of the overall fitness landscape and resulting <sup>71</sup> in **diverse** sequence designs

Lee, Jason, Elman Mansimov, and Kyunghyun Cho. "Deterministic non-autoregressive neural sequence modeling by iterative refinement."(2018).

# Approaches to Conditional Design (exploring original sequence space)

Guiding design by an oracle (e.g., Yang et al. 2019, Alley et al. 2019, Brookes et al. 2019, Biswas et al. 2021)



## Machine-learning-guided directed evolution for protein engineering

Kevin K. Yang, Zachary Wu and Frances H. Arnold\*



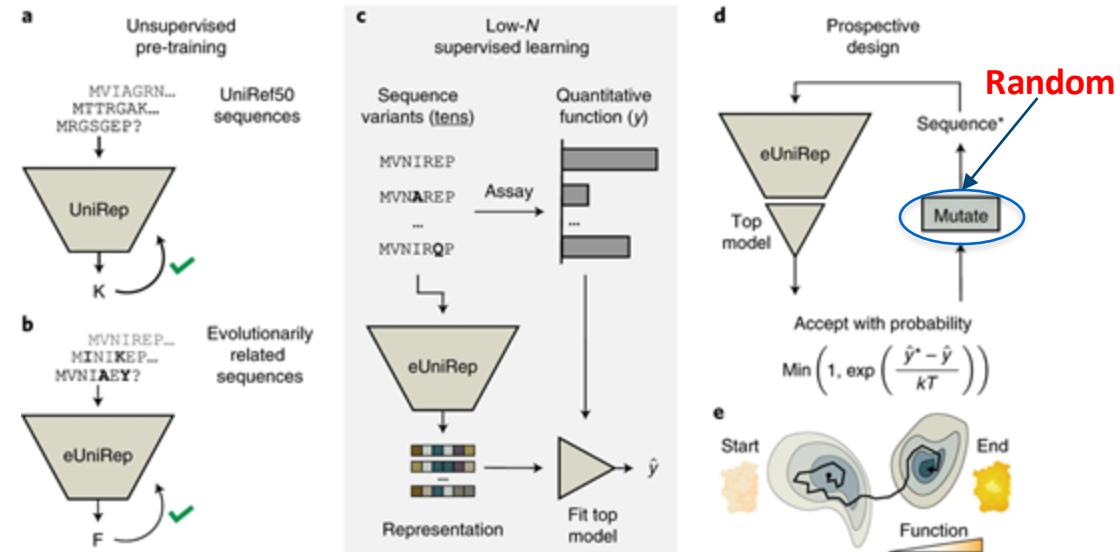
## Unified rational protein engineering with sequence-based deep representation learning

Ethan C. Alley<sup>1,2\*</sup>, Grigory Khimulya<sup>3,4</sup>, Surojit Biswas<sup>1,5</sup>, Mohammed AlQuraishi<sup>3,4</sup> and George M. Church<sup>3,5</sup>



## Low- $N$ protein engineering with data-efficient deep learning

Surojit Biswas<sup>1,2\*</sup>, Grigory Khimulya<sup>3,4</sup>, Ethan C. Alley<sup>4,5</sup>, Kevin M. Esvelt<sup>3,4</sup> and George M. Church<sup>3,5,6</sup>

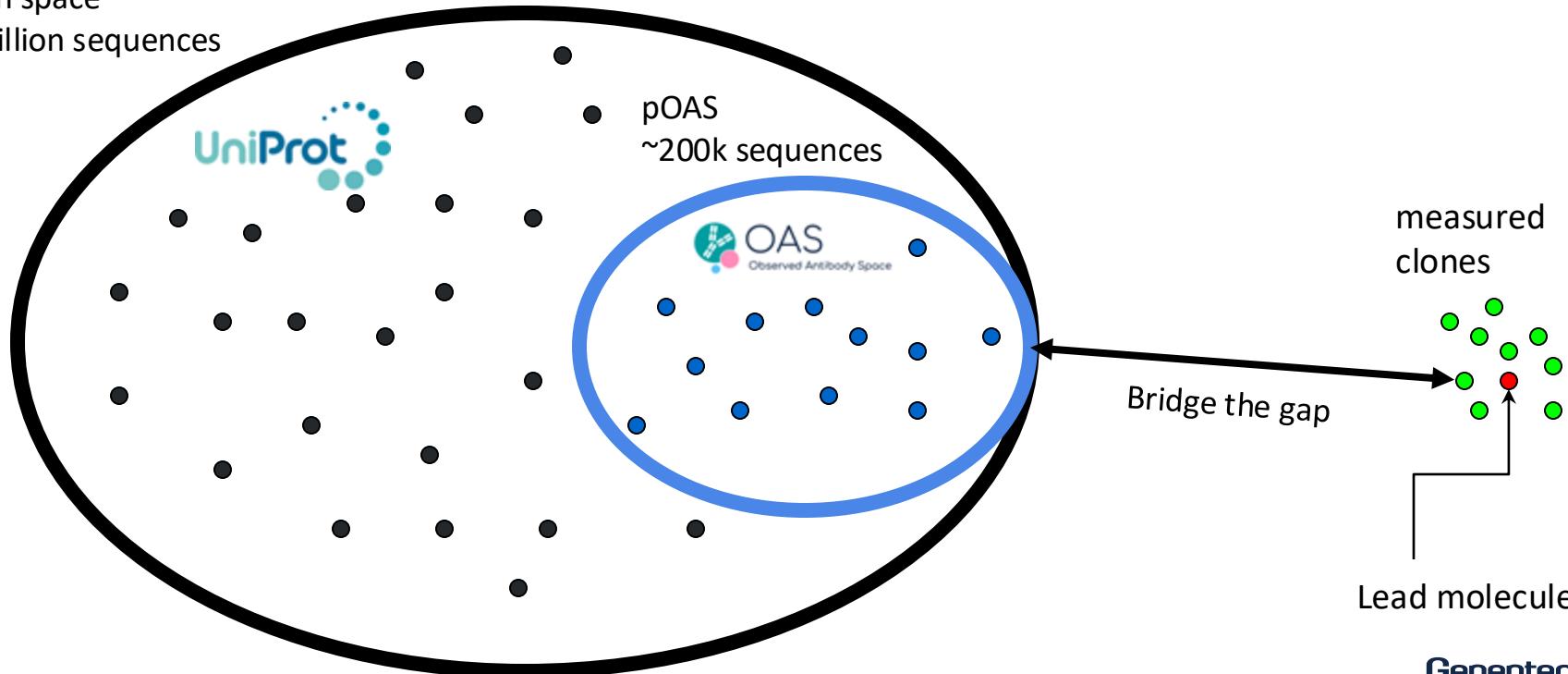


Biswas et al., 2021

# Publicly available data vs. reality

Protein space

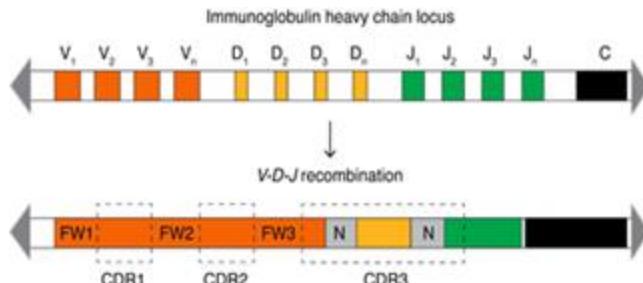
~2.4 billion sequences



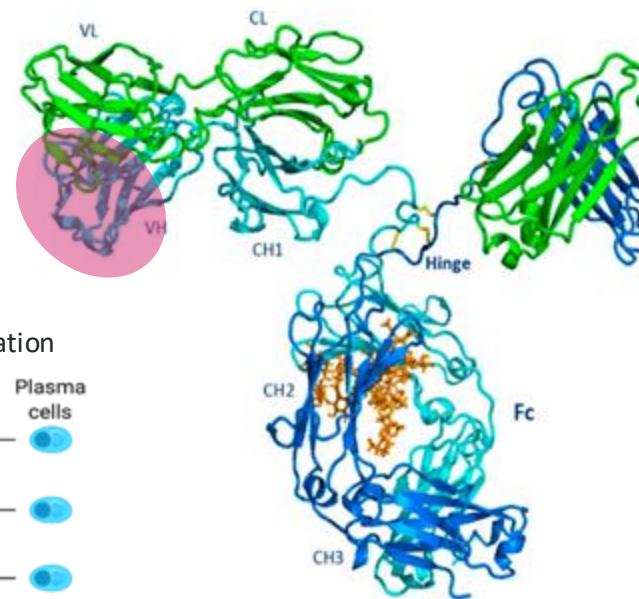
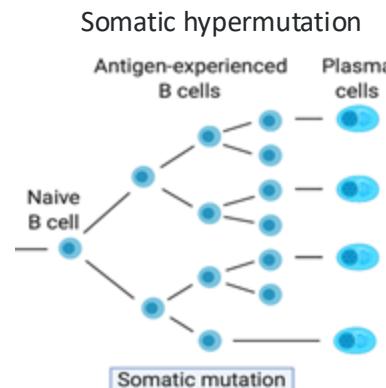
# How nature generates antibodies

Genome contains multiple copies of 3 different types of gene segments for the variable regions of Abs

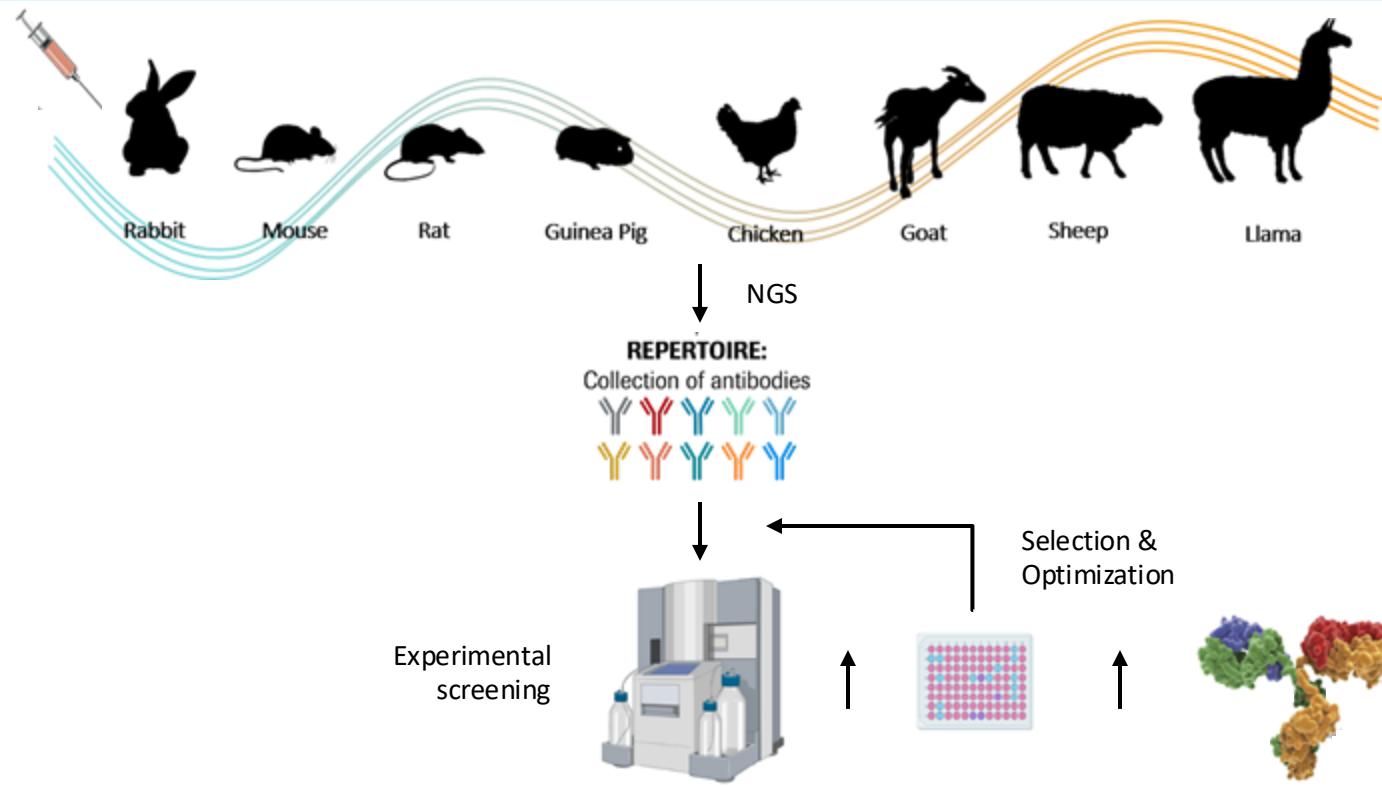
DNA rearrangement causes one copy of each type of gene segment to go in any given B-cell, generating an enormous antibody repertoire; roughly  $3 \times 10^{11}$ .



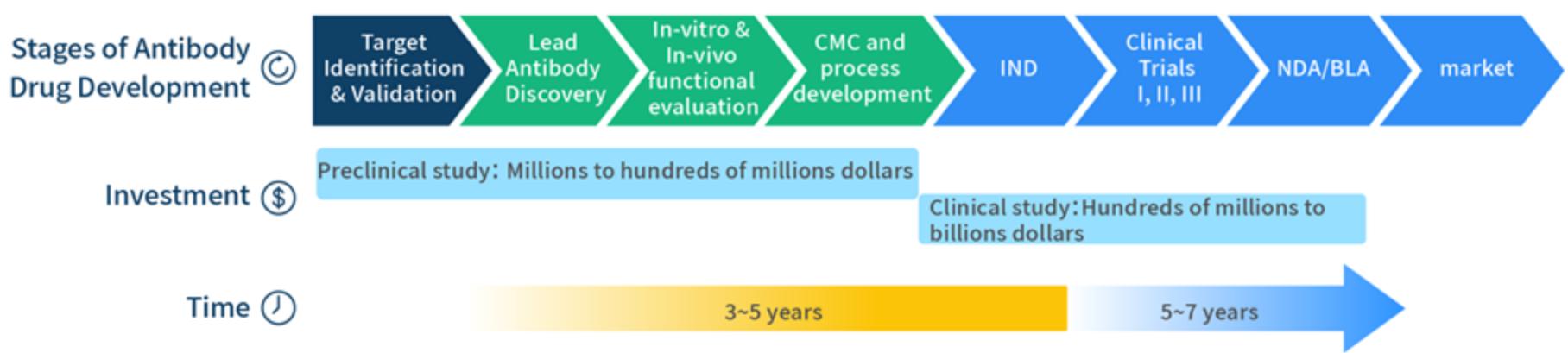
1987 Nobel Prize in Physiology or Medicine  
by Susumu Tonegawa



# How pharma typically discovers antibodies

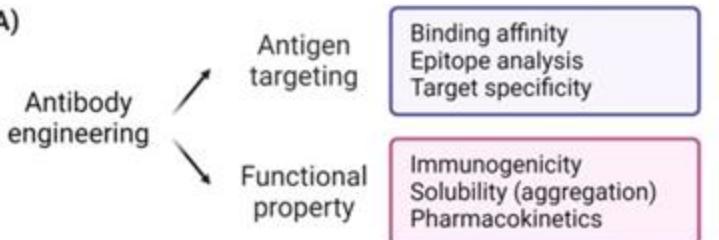


# Stages of antibody drug development process

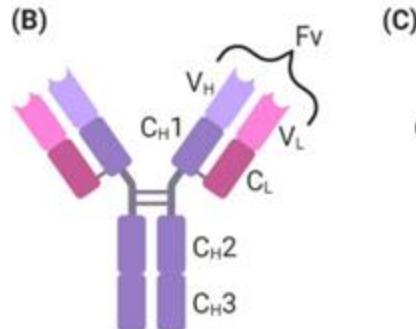


# Antibody developability properties

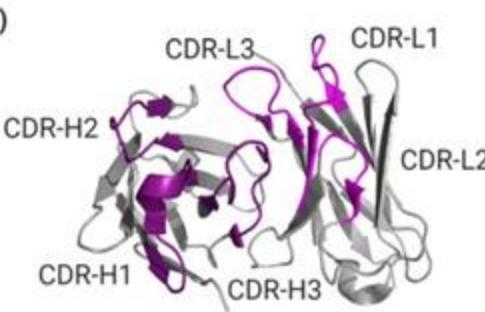
(A)



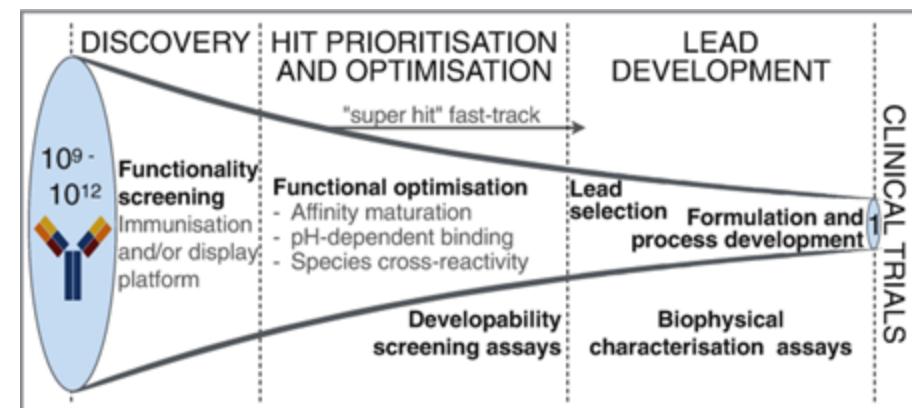
(B)



(C)



Trends in Pharmacological Sciences



# How AI can help antibody design & discovery ?

## Generative models:

- can help in hit finding and generating proposals (clever library design instead of combinatorial search or random diversification)
- sequence multi-property optimization (generating proposal that can satisfy multiple objectives at the same time)
- *de novo* design (that might replace the entire immunization/discovery process)

## Discriminative models (property predictors/oracles):

- antibody-antigen affinity prediction ( $K_d$  predictors)
- developability property predictors for *in silico* screening (faster & cheaper)
- replacing functional assays
- guiding generative models (e.g., guiding library design)

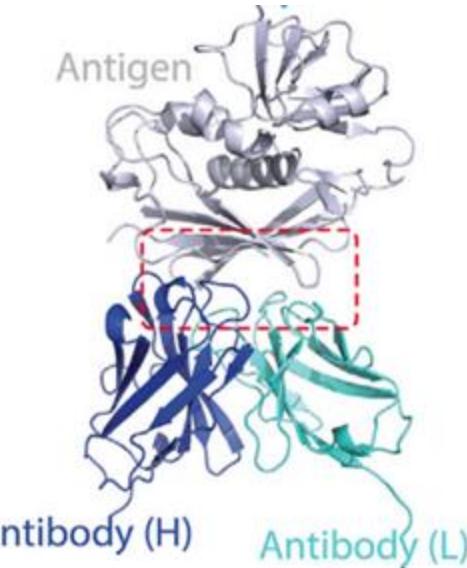
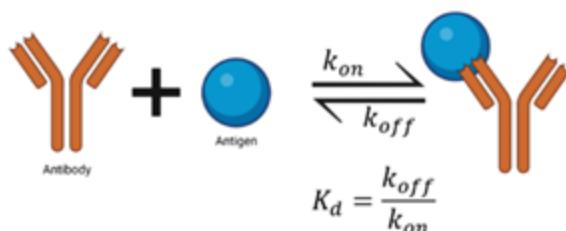
# Affinity/binding prediction

## Predict binding affinity

- Input type: unbound Ab+Ag (sequence or folded structures)
- Output type: (regression) equilibrium dissociation constant Kd; (classification) bind/no bind?

## Related tasks:

- Predict interface (epitope/paratope)
- Docking



## Experimental techniques:

- enzyme-linked immunosorbent assay (ELISA)
- surface plasmon resonance (SPR)

# Why is the problem hard ?

## Machine learning



Sequence space is extremely sparsely functional

**Formally:** hard to learn an affinity prediction function that is Lipschitz continuous

$$f : D \rightarrow \mathbb{R} \text{ is Lipschitz continuous}$$

$$\iff \exists L \geq 0 \forall x, \tilde{x} \in D : |f(x) - f(\tilde{x})| \leq L \cdot |x - \tilde{x}|$$

## Biology

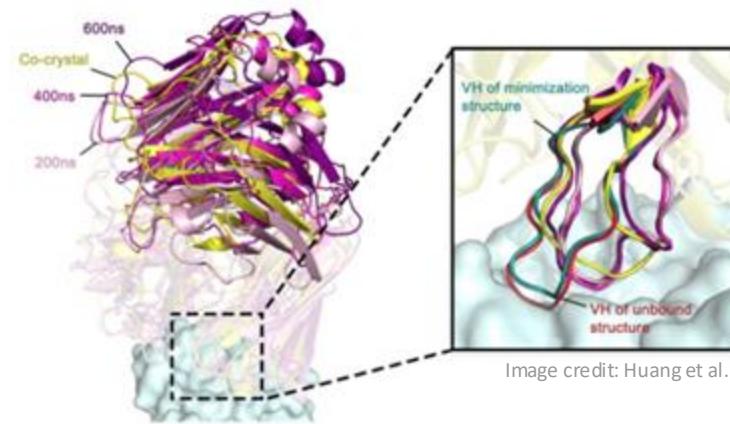


Image credit: Huang et al. 2022

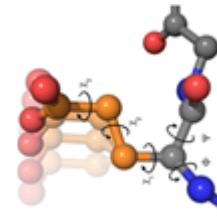
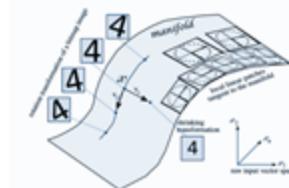
Function depends on dynamics of loops and solvent, which are hard to simulate

# How we overcome these challenges with ML ?

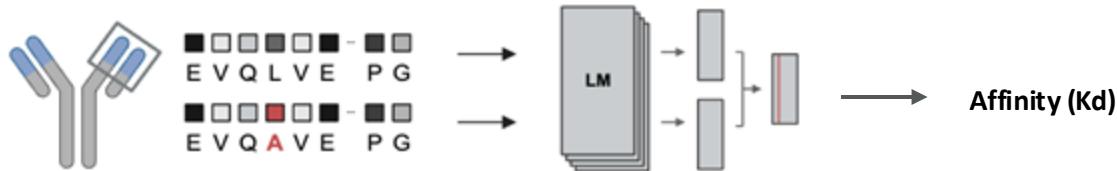
Real-world high-dimensional data lie (roughly) on a **low-dimensional manifold** (Chapelle 2006). Easy to learn with deep learning.

Protein representation & inductive biases help deal with data scarcity:

- PLM embeddings, equivariant models, internal geometry

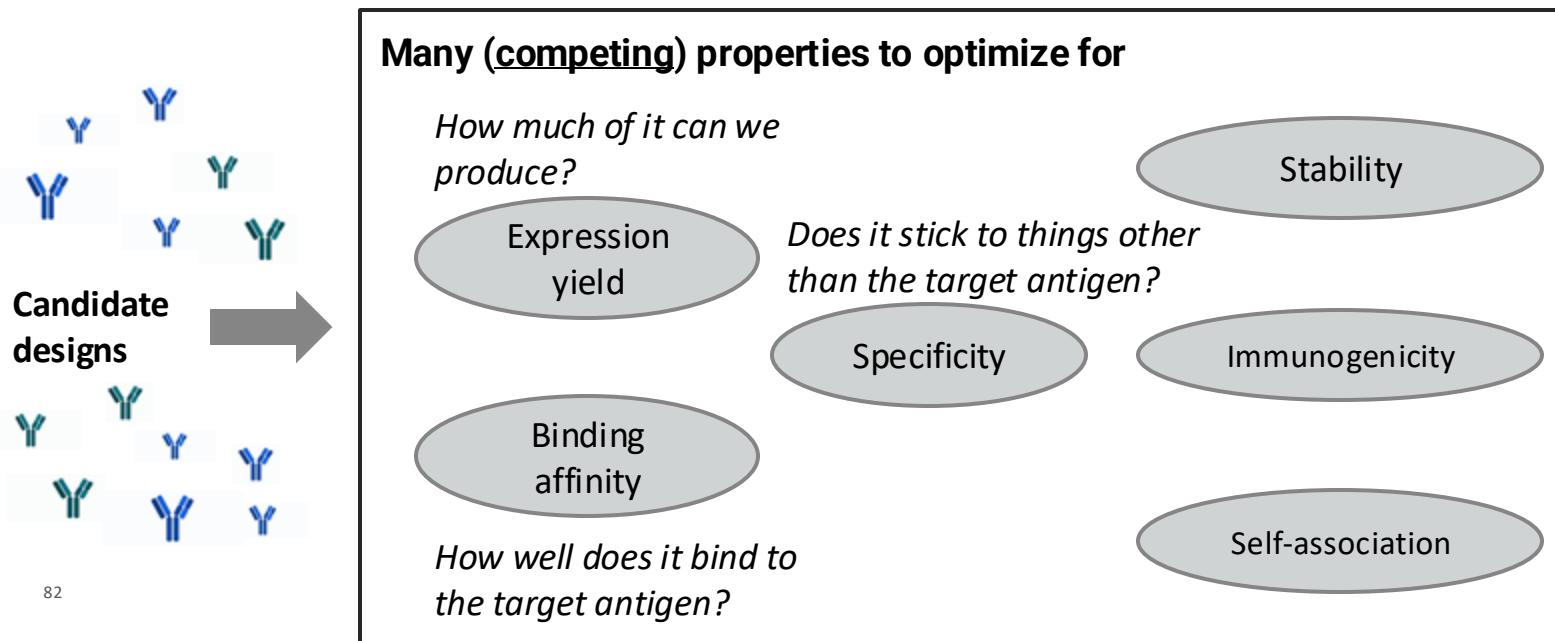


## Sequence-based antibody-antigen affinity prediction

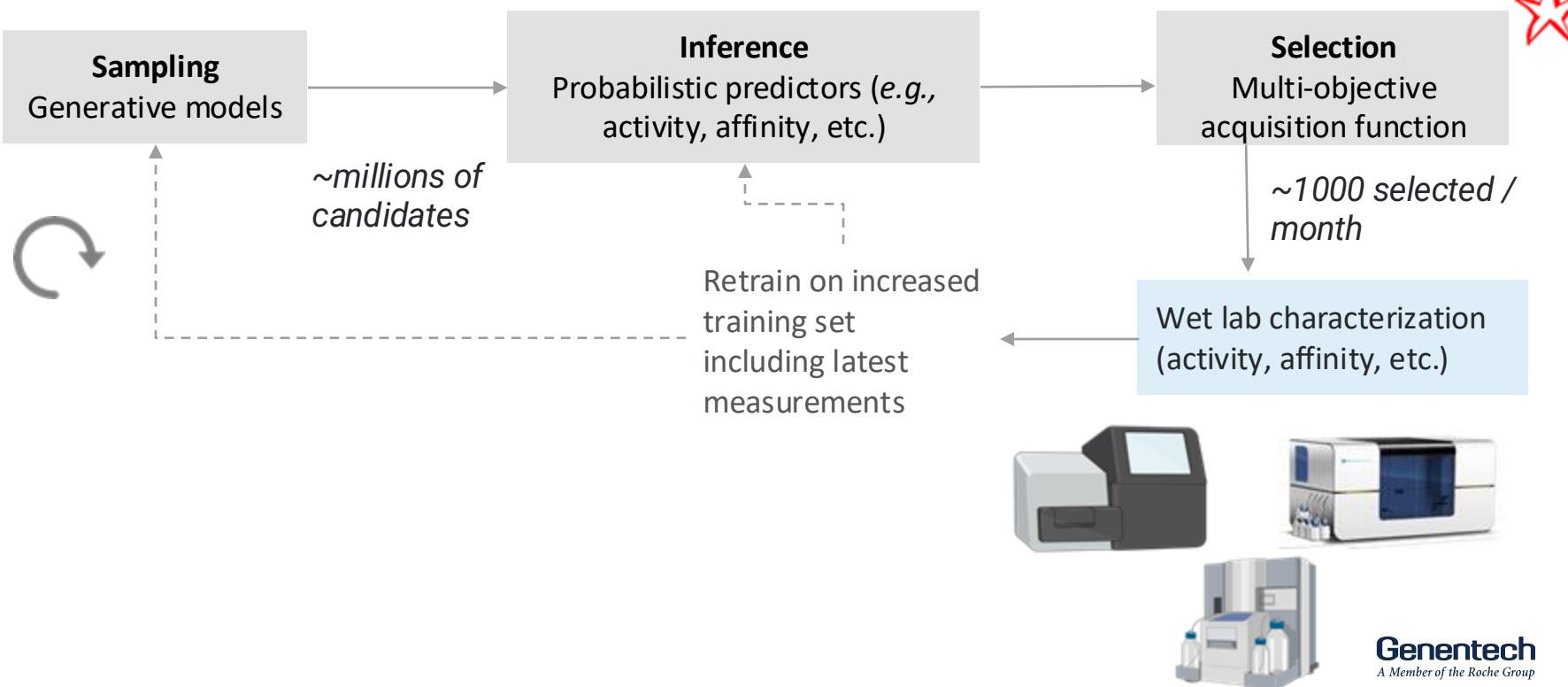


# Multi-objective optimization for lead molecule optimization

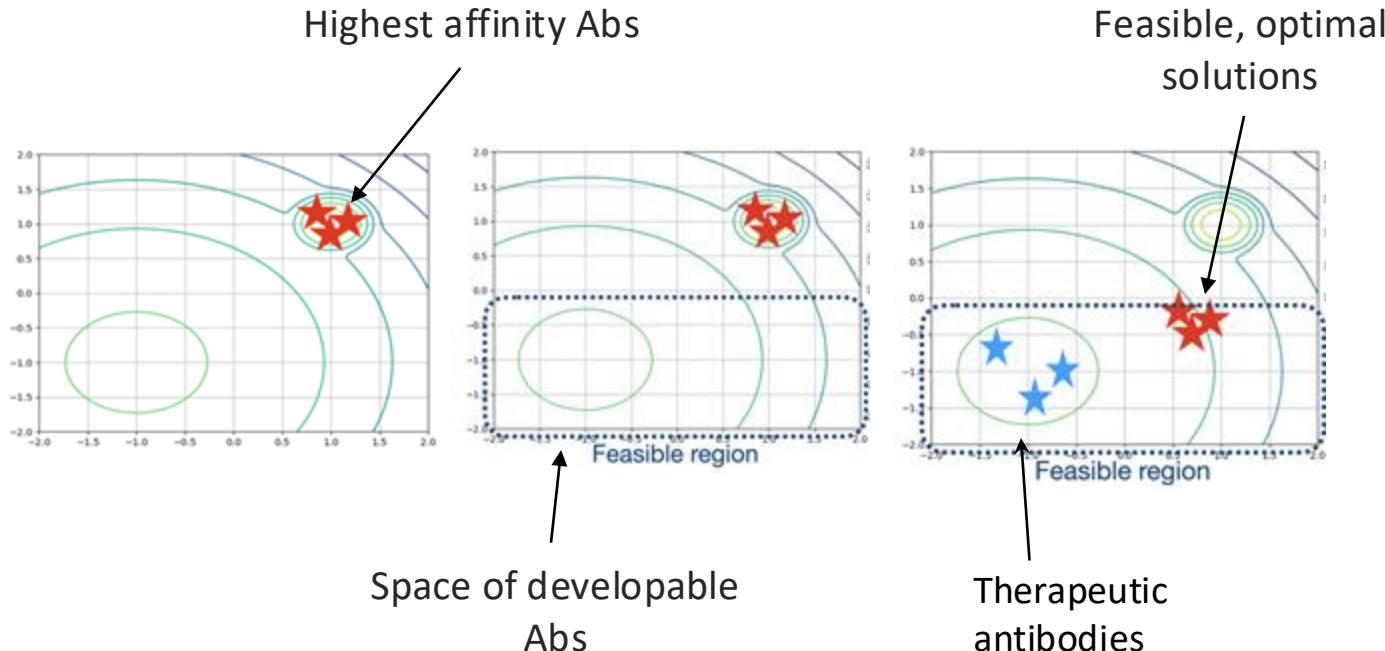
How do we design molecules with better properties to: 1) reduce time-to-clinic; 2) increase clinical success rates; and 3) target previously undruggable diseases



# Lab-in-the-Loop: active learning selection and , optimizing for multiple properties



# Space of developable antibodies



The only way of making **therapeutic antibodies** is considering all the objective functions at the same time!

# Pareto optimality & therapeutic antibodies

Find set of sequences at Pareto front:

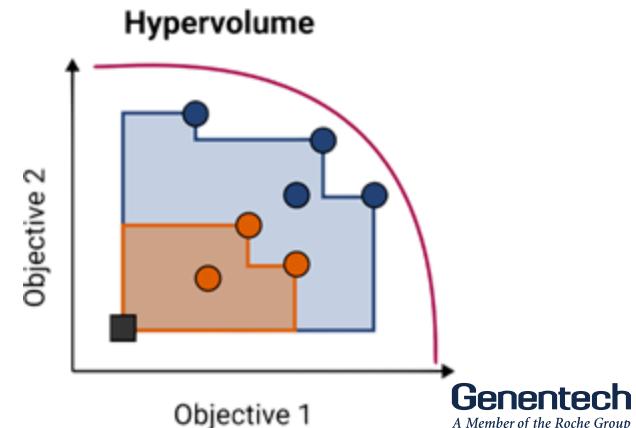
$$\mathcal{P} = \{S \mid \{S' \mid S' > S\} = \emptyset\}$$

$S' > S$  if  $f_k(S') > f_k(S)$  for all  $k = 1, \dots, K$  (S' strictly dominates S)

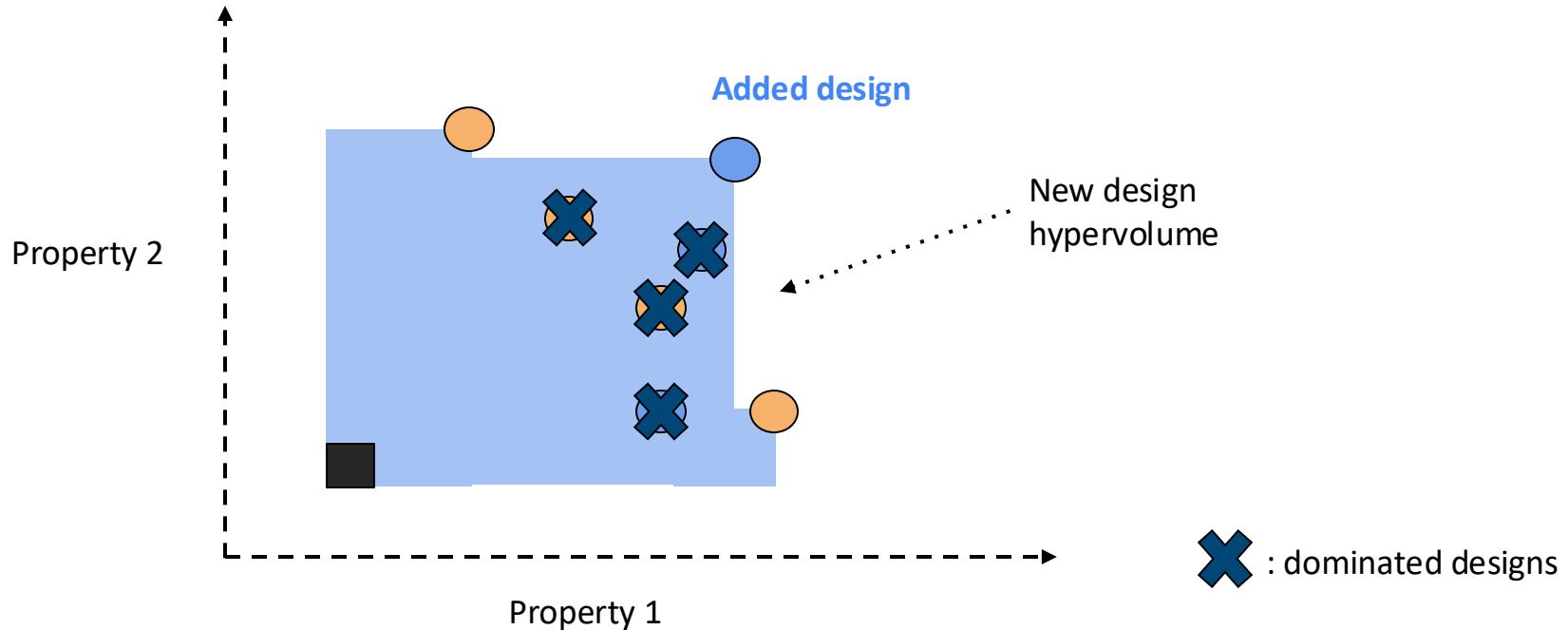
where,  $f_k(S) \in \mathbb{R}$  is the k-th objective function (affinity, activity, specificity, etc.)

How to measure the quality of the frontier (acquisition function)?

- Volume: Expected hypervolume improvement (EHVI; Emmerich *et al.* 2011)
- Expected HVI is defined over the posterior distribution of our property predictors (pseudo-oracles) (Daulton *et al.*, 2020)



# Multi-property optimization: The best designs increase the “hypervolume” the most

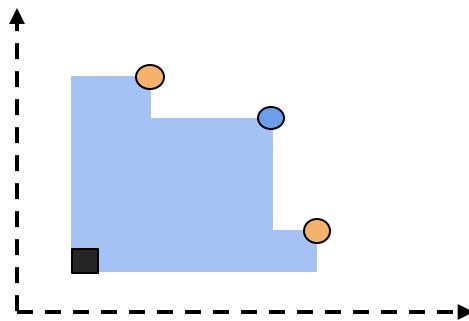


Properties are **jointly** maximized!

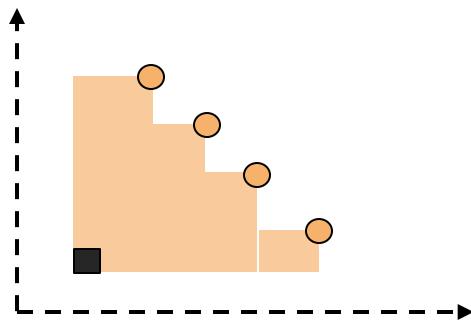
Contrast with sequential maximization or arbitrary weighting of properties

# What is hypervolume *improvement*?

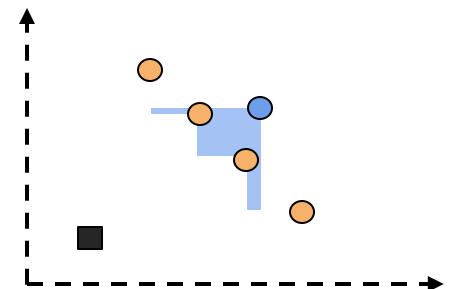
Take the difference of the hypervolumes before and after adding the candidate design.  
The best designs have the highest HVI.



After adding **design**



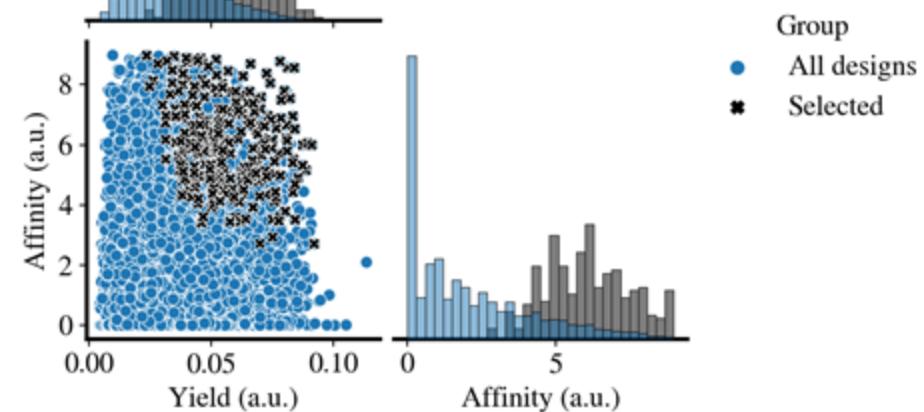
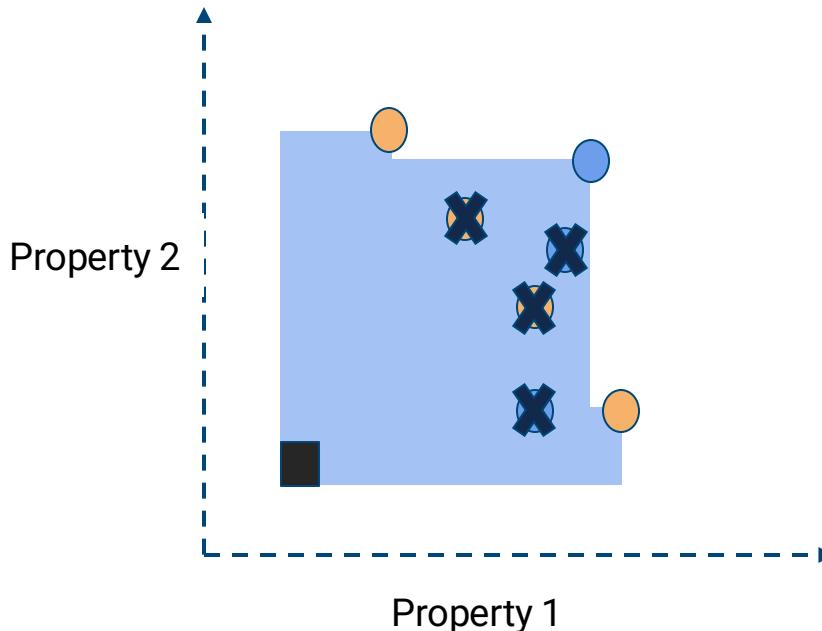
Before, with just **known points**



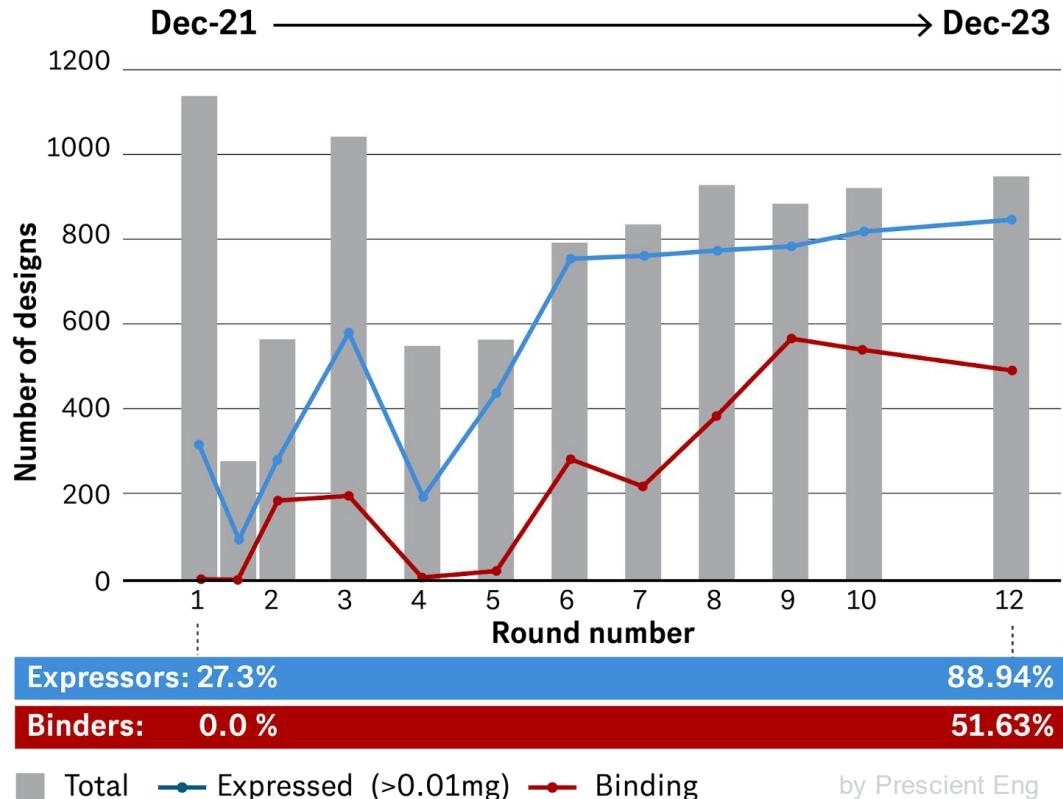
Hypervolume *Improvement* (HVI)

# Active learning: multi-objective sample selection through expanding the Pareto frontier

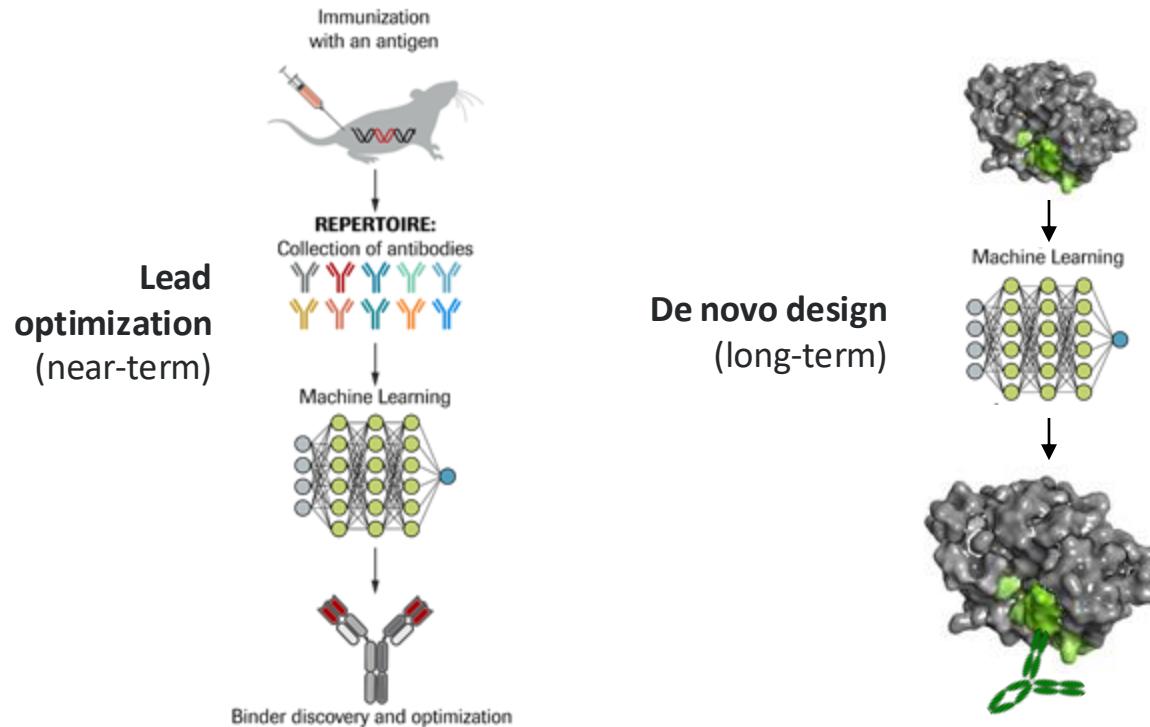
BOtied: Expected hypervolume improvement (EHVI) guides exploration vs. exploitation balance towards expanding the Pareto frontier over multiple developability properties.



# LitL importance evidenced by expression & affinity round-over-round improvement

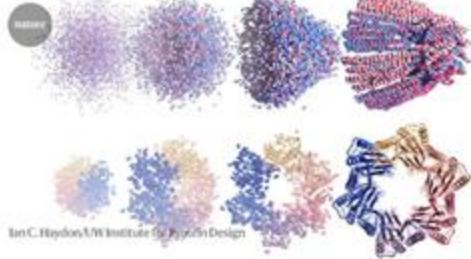


# How will *de novo* generative AI impact antibody discovery in the future ?

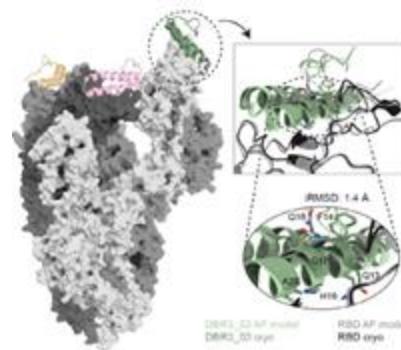


# Great progress in protein design with generative AI

## General protein design



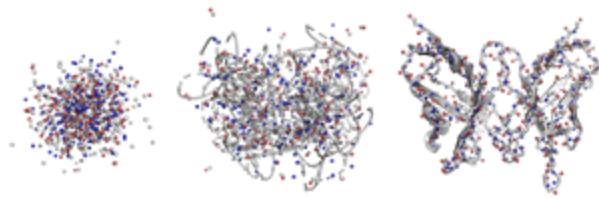
De novo design of protein structure and function with **RFDiffusion**. Watson et al. 2023



De novo design of site-specific protein interactions with learned surface fingerprints. Gainza et al. Nature 2023 (**MaSIF**)

Ingraham et al 2022 (**Chroma**), Yim et al 2023 (**FrameDiff**), ..

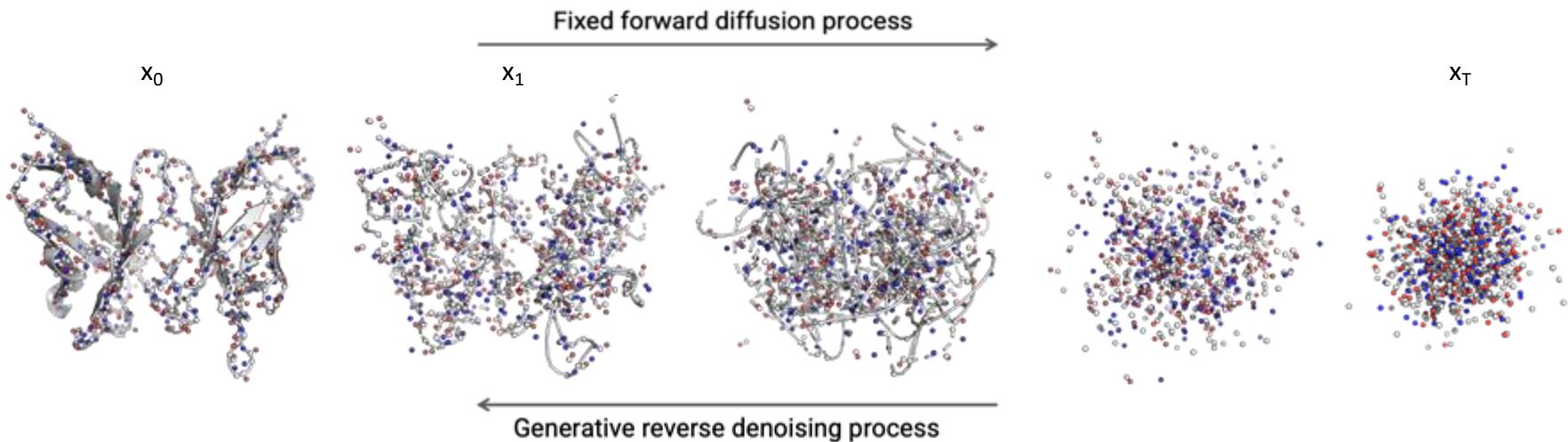
## Antibody-specific models



- Iterative Refinement Graph Neural Network for Antibody Sequence-Structure Co-design. ICLR 2022 (**RefineGNN**)
- *Protein Structure and Sequence Generation with Equivariant Denoising Diffusion Probabilistic Models*. Namrata & Tudor. arXiv 2022
- End-to-End Full-Atom Antibody Design. ICML 2023 (**dyMEAN**)
- Antigen-Specific Antibody Design and Optimization with Diffusion-Based Generative Models. Luo et al. NeurIPS 2023. (**DiffAb**)

Diffusion dominates!

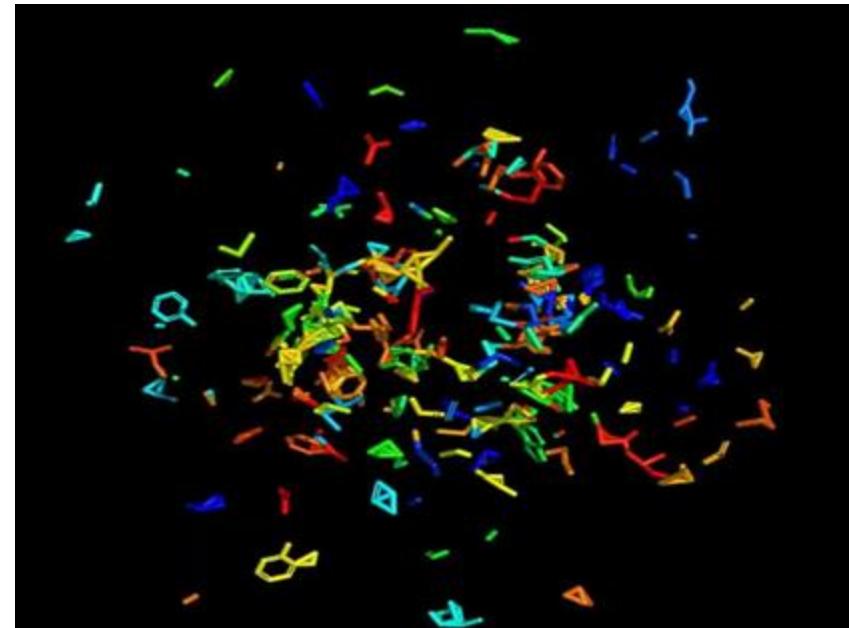
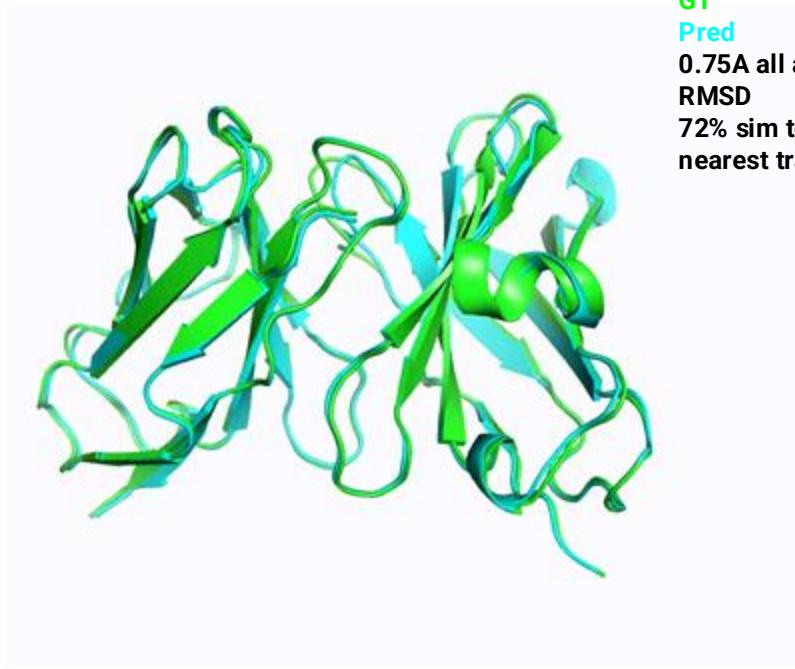
# Denoising Diffusion on proteins



Steps from a sampling trajectory. Beginning with randomly initialised positions, our model iteratively updates the position of the backbone atoms, producing a recognisable antibody backbone at the end of the process.

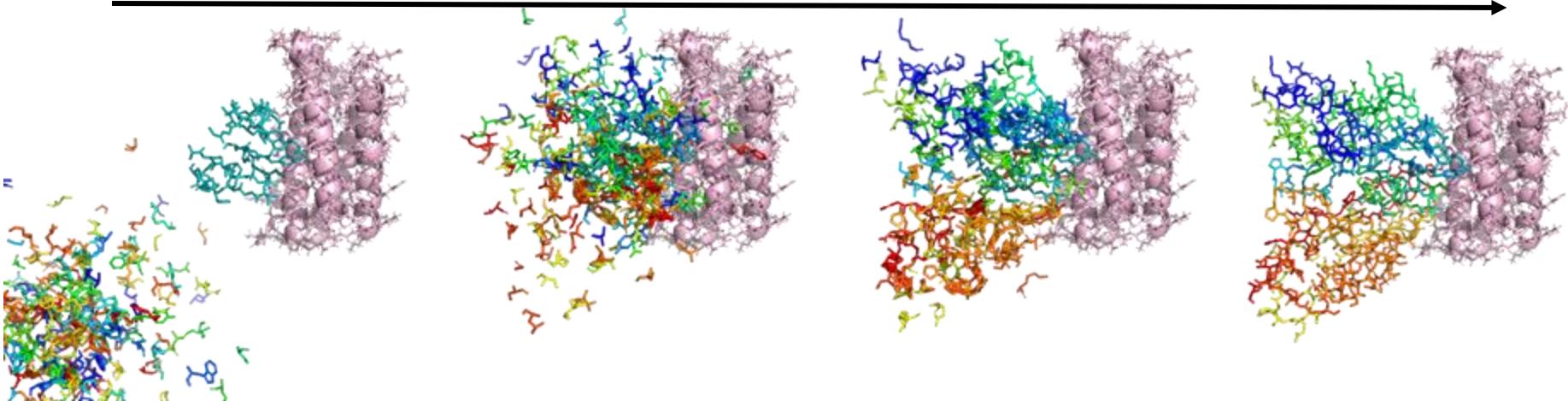
Martinkus, Karolis, et al. "Abdiffuser: Full-atom generation of in-vitro functioning antibodies." *NeurIPS* (2023).

# EquiFold: antibody structure prediction & design



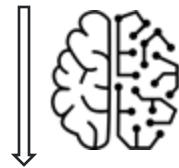
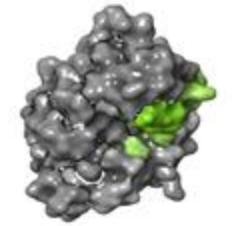
# Guided diffusion sampling

Direction of guided generative process

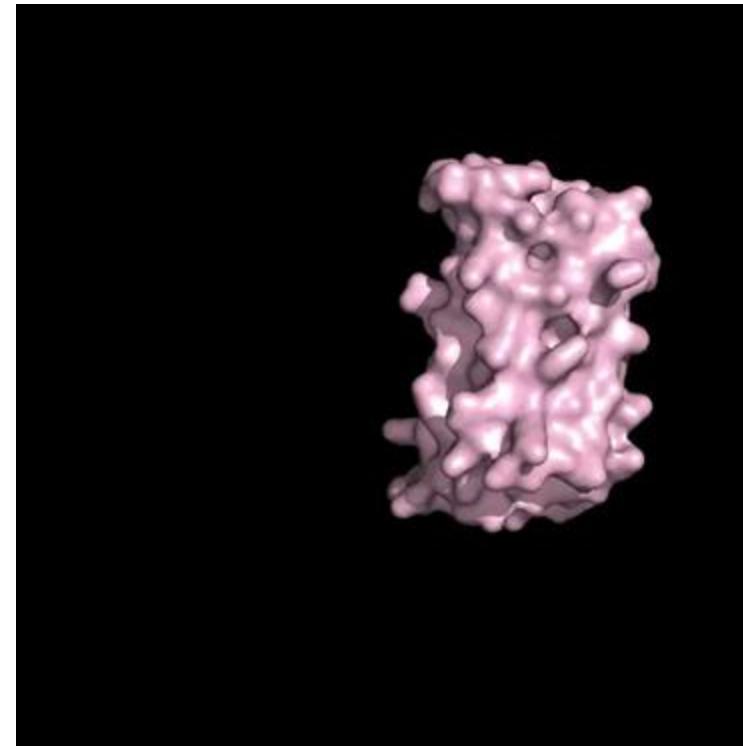
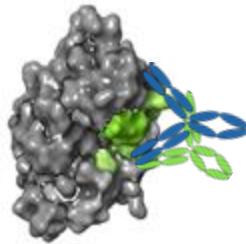


**Next steps:** Further improvements of the generative algorithm working together with the Genentech AE team for experimental testing

# Generative AI for *de novo* design is the future!



**Generative AI**



# Thank you!

---

## Bonus slides - AlphaFold 3

### Article

## Improved protein structure prediction using potentials from deep learning

<https://doi.org/10.1038/s41586-019-1923-7>

Received: 2 April 2019

Accepted: 10 December 2019

Published online: 15 January 2020

Andrew W. Senior<sup>1,4\*</sup>, Richard Evans<sup>1,5</sup>, John Jumper<sup>1,6</sup>, James Kirkpatrick<sup>1,6</sup>, Laurent Sifre<sup>1,6</sup>, Tim Green<sup>1</sup>, Chongli Qin<sup>1</sup>, Augustin Žídek<sup>1</sup>, Alexander W. R. Nelsen<sup>1</sup>, Alex Bridgland<sup>1</sup>, Hugo Penedones<sup>1</sup>, Stig Petersen<sup>1</sup>, Karen Simonyan<sup>1</sup>, Steve Crossan<sup>1</sup>, Pushmeet Kohli<sup>1</sup>, David T. Jones<sup>1,3</sup>, David Silver<sup>1</sup>, Koray Kavukcuoglu<sup>1</sup> & Demis Hassabis<sup>1</sup>

2018

2020

2024

### Article

## Highly accurate protein structure prediction with AlphaFold

<https://doi.org/10.1038/s41586-021-03819-2>

Received: 11 May 2021

Accepted: 12 July 2021

Published online: 15 July 2021

Open access

Check for updates

John Jumper<sup>1,4,5</sup>, Richard Evans<sup>1,5</sup>, Alexander Pritzel<sup>1,6</sup>, Tim Green<sup>1,6</sup>, Michael Figurnov<sup>1,6</sup>, Olaf Ronneberger<sup>1,6</sup>, Kathryn Tunyasuvunakool<sup>1,6</sup>, Russ Bates<sup>1,6</sup>, Augustin Žídek<sup>1,6</sup>, Anna Potapenko<sup>1,6</sup>, Alex Bridgland<sup>1,6</sup>, Clemens Meyer<sup>1,6</sup>, Simon A. A. Kohl<sup>1,6</sup>, Andrew J. Ballard<sup>1</sup>, Andrew Cowie<sup>1</sup>, Bernardino Romera-Paredes<sup>1</sup>, Stanislav Nikolov<sup>1,6</sup>, Rishabh Jain<sup>1,6</sup>, Jonas Adler<sup>1</sup>, Trevor Back<sup>1</sup>, Stig Petersen<sup>1</sup>, David Reiman<sup>1</sup>, Ellen Clancy<sup>1</sup>, Michal Zeliński<sup>1</sup>, Martin Steinegger<sup>1,2</sup>, Michałina Pacholska<sup>1</sup>, Tamás Berg hammer<sup>1</sup>, Sebastian Bodenstein<sup>1</sup>, David Silver<sup>1</sup>, Oriol Vinyals<sup>1</sup>, Andrew W. Senior<sup>1</sup>, Koray Kavukcuoglu<sup>1</sup>, Pushmeet Kohli<sup>1</sup> & Demis Hassabis<sup>1,6,7</sup>

### nature

#### Accelerated Article Preview

## Accurate structure prediction of biomolecular interactions with AlphaFold 3

Received: 19 December 2023

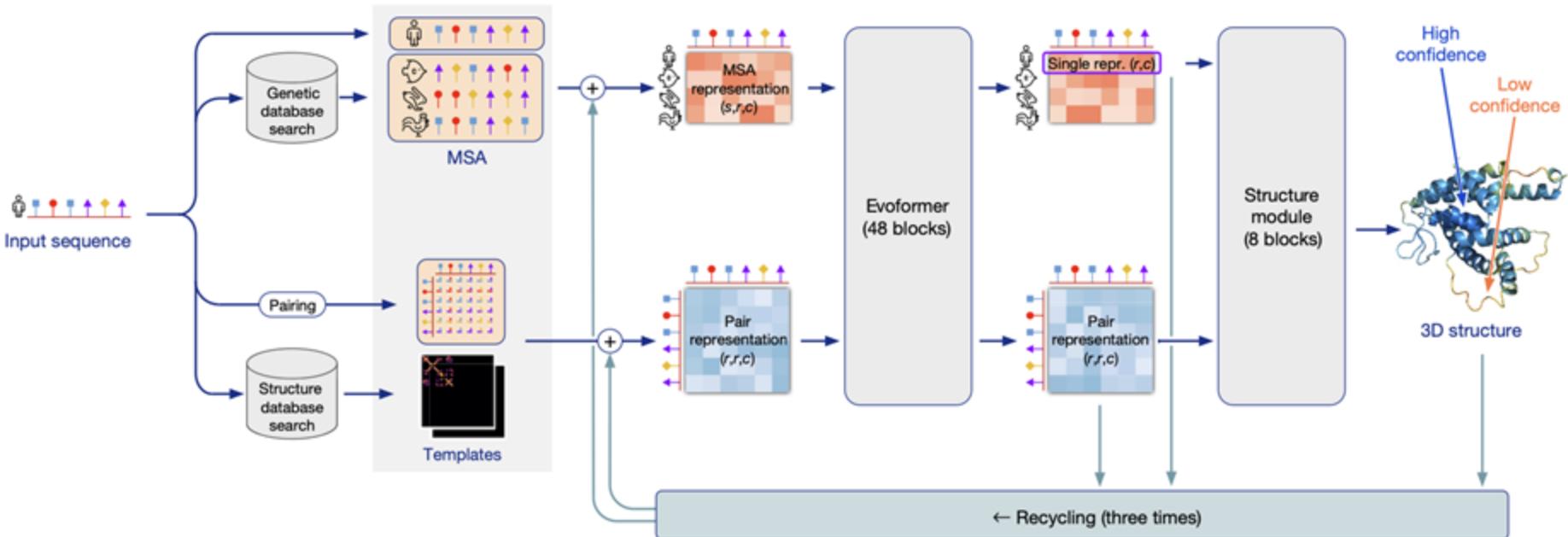
Accepted: 29 April 2024

Accelerated Article Preview

Cite this article as: Abramson, J. et al. Accurate structure prediction of biomolecular interactions with AlphaFold 3. *Nature* <https://doi.org/10.1038/s41586-024-03487-w> (2024)

Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Williams, Andrew J. Ballard, Joshua Bambica, Sebastian W. Bodenstein, David A. Evans, Chi-Chun Huang, Michael O'Neill, David Reiman, Kathryn Tunyasuvunakool, Zachary Wu, Alvia Zengulya, Enriq Arivio, Charles Beattie, Ottavia Bertoldi, Alex Bridgland, Alexey Cherezmanov, Miles Cognetti, Alexander I. Cowen-Rivers, Andrew Cowie, Michael Figurnov, Fabian B. Fuchs, Hannah Gladman, Rishabh Jain, Yousef A. Khan, Caroline M. R. Low, Kubra Perlin, Anna Potapenko, Pascual Savo, Sukhdeep Singh, Adrian Stecula, Ashok Thilaisundaram, Catherine Tong, Sergei Yakhnenko, Ellen D. Zhong, Michal Zeliński, Augustin Žídek, Victor Bapst, Pushmeet Kohli, Max Jaderberg, Dennis Hassabis & John M. Jumper

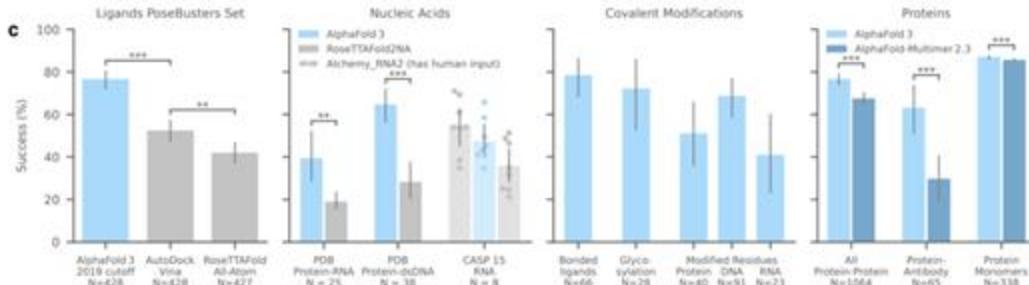
# AlphaFold 3



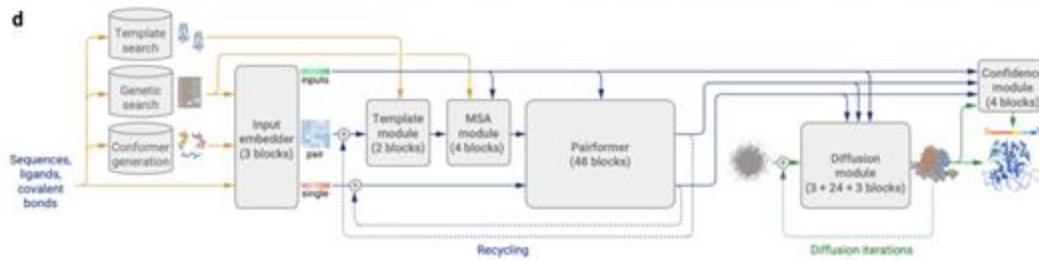
# AlphaFold 3



Increased structure prediction accuracy



Increased complex (binding) prediction accuracy



Generation is done all at the same time

# Weisfeiler Leman graph isomorphism test

---

## Algorithm 1: WL-1 algorithm (Weisfeiler & Lehmann, 1968)

---

**Input:** Initial node coloring  $(h_1^{(0)}, h_2^{(0)}, \dots, h_N^{(0)})$

**Output:** Final node coloring  $(h_1^{(T)}, h_2^{(T)}, \dots, h_N^{(T)})$

$t \leftarrow 0;$

**repeat**

**for**  $v_i \in \mathcal{V}$  **do**

$h_i^{(t+1)} \leftarrow \text{hash} \left( \sum_{j \in \mathcal{N}_i} h_j^{(t)} \right);$

$t \leftarrow t + 1;$

**until** *stable node coloring is reached*;

---