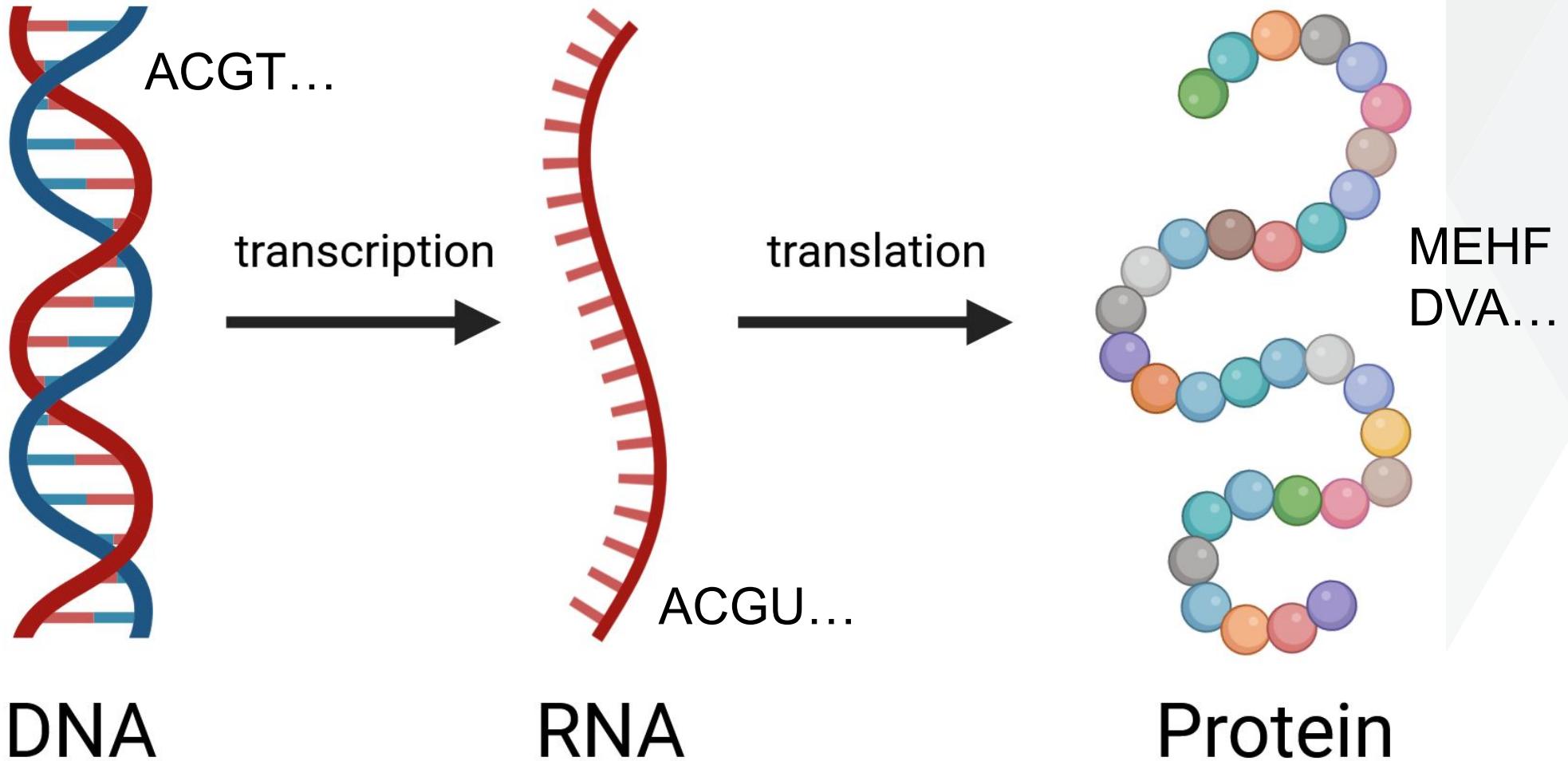


Deep Learning for Protein Structure

Content

1. Models for solving the protein folding problem
 - a. AlphaFold2
 - b. OmegaFold
 - c. ESM
2. RMSD for 3D protein structure comparison
3. Exercise 0: Predicting protein structure from amino acid sequence
4. Large Language Models (LLMs)
 - a. Transformers (BERT)
 - b. HuggingFace
5. Case study: proteins with knots in their 3D structure
6. Exercise 1: Protein embedding extraction + classification idea guidance (CNN)
7. Exercise 2: Fine-tuning a ProtBert-BFD model for knot prediction

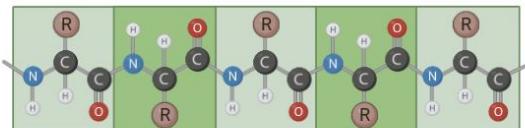
Central dogma of biology



Protein folding problem

Primary structure

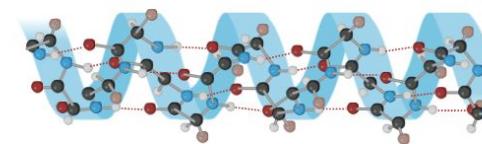
Polypeptide chain



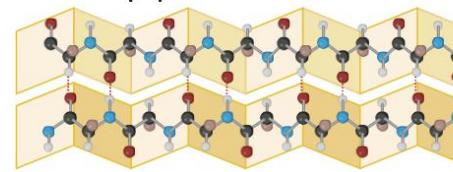
Amino acid

Secondary structure

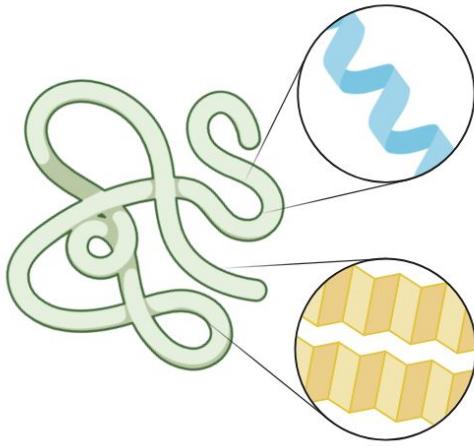
α -helix



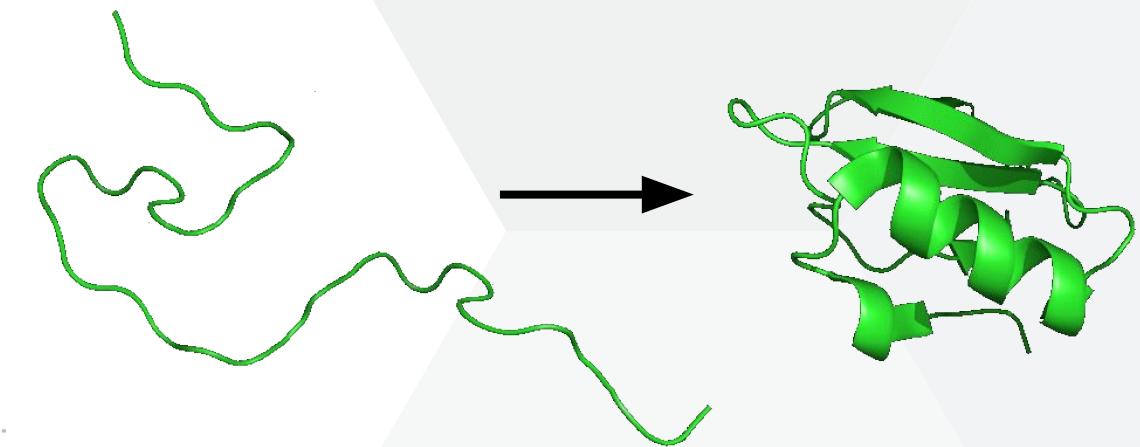
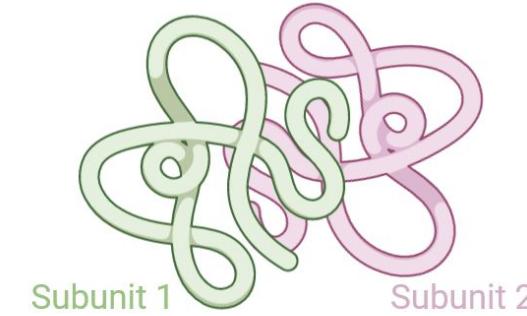
β -pleated sheet



Tertiary structure



Quaternary structure

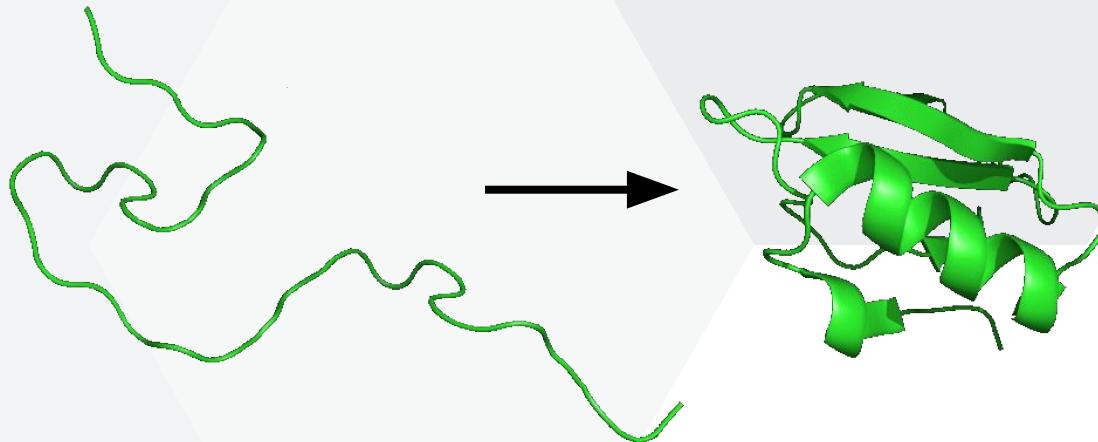


protein before and after folding

https://en.wikipedia.org/wiki/Protein_folding



Protein folding problem

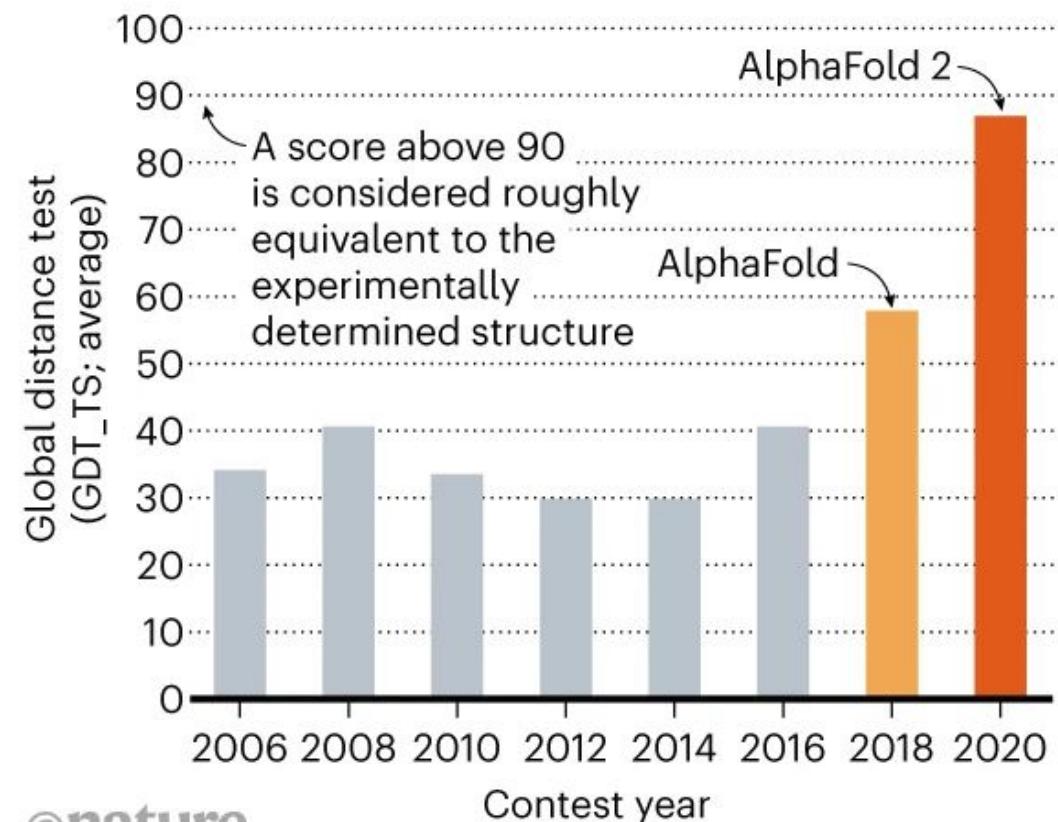


protein before and after folding

https://en.wikipedia.org/wiki/Protein_folding

STRUCTURE SOLVER

DeepMind's AlphaFold 2 algorithm significantly outperformed other teams at the CASP14 protein-folding contest — and its previous version's performance at the last CASP.



©nature

<https://elearning.vib.be/courses/alphafold/lessons/introduction-to-alphaFold/topic/critical-assessment-for-structure-prediction-casp/>

Anfisen's dogma ("thermodynamics hypothesis")

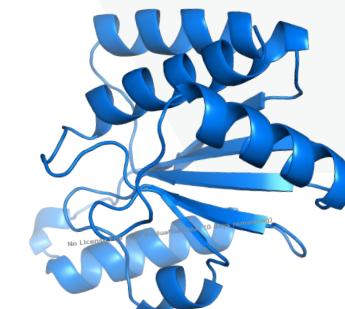
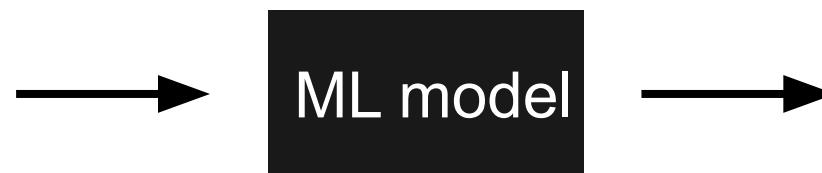
"at least for a small globular protein in its standard physiological environment, the native structure is determined only by the protein's amino acid sequence" - Christian B. Anfinsen

=> we can treat proteins as their amino acid sequences

=> we can use other methods that have been widely used on text (NLP)

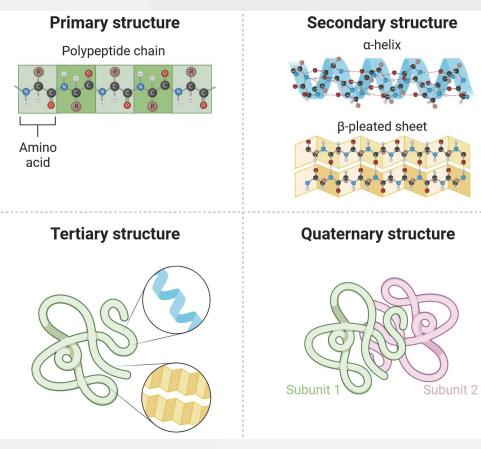
=> deep learning models, Transformers, ...

MKFTLLVVGRTVEK
HYITAINDYVERTR
HFISFDMEVIP...



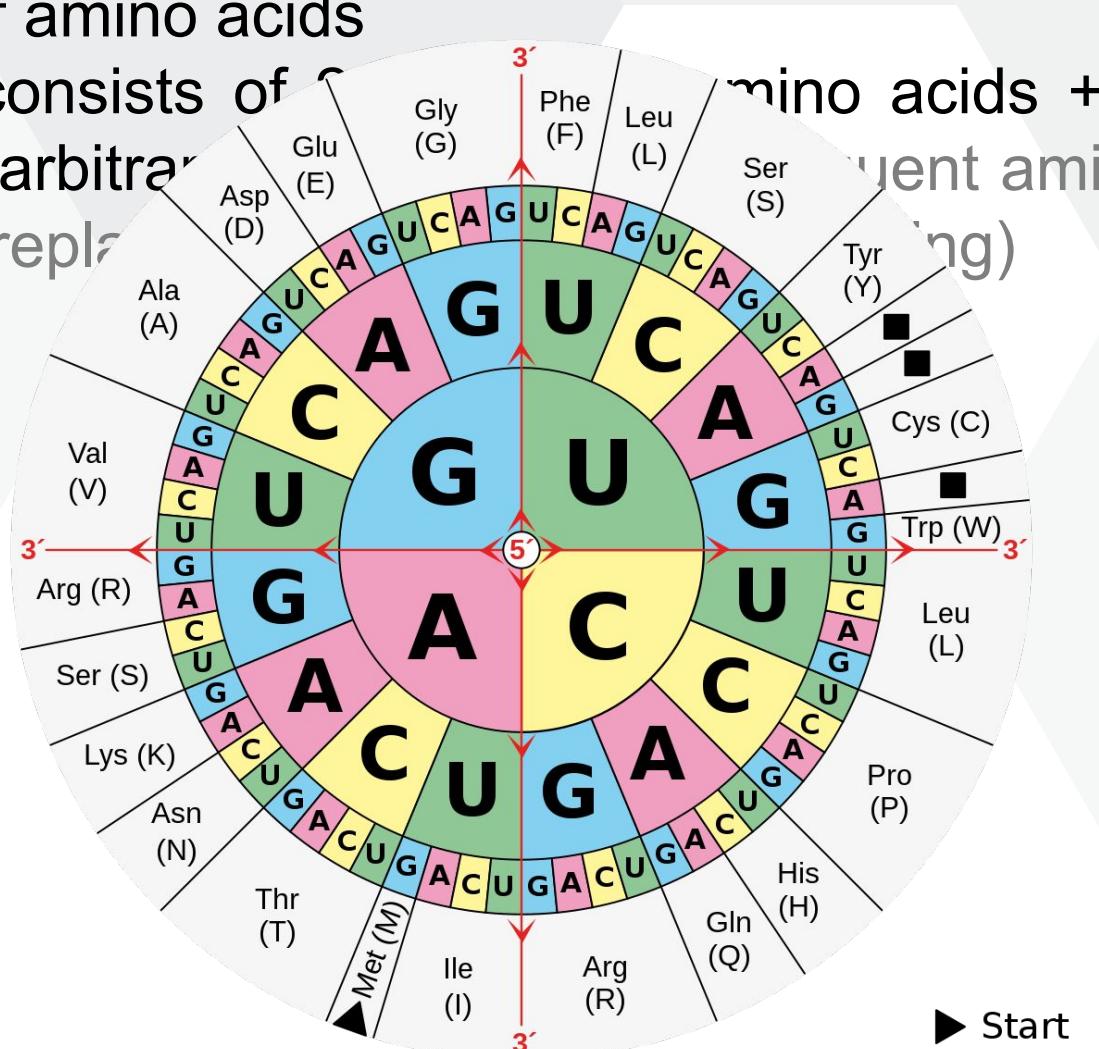
Protein (data format) from NLP point of view

- protein == string of amino acids
- alphabet usually consists of 20 common amino acids + X
that stands for an arbitrary amino acid (less frequent amino acids (UZOB) are replaced by X during preprocessing)

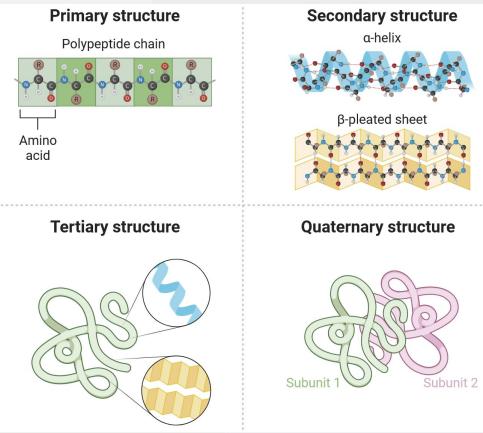


Protein (data format) from NLP point of view

- protein == string of amino acids
 - alphabet usually consists of
that stands for an arbitrary
amino acid (UZOB) are replaced by
the corresponding amino acid



 Start Stop

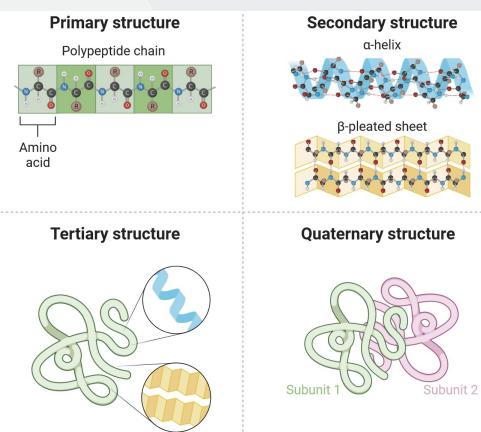


Protein (data format) from NLP point of view

- protein == string of amino acids
- alphabet usually consists of 20 common amino acids + X
that stands for an arbitrary amino acid (less frequent amino acids (UZOB) are replaced by X during preprocessing)

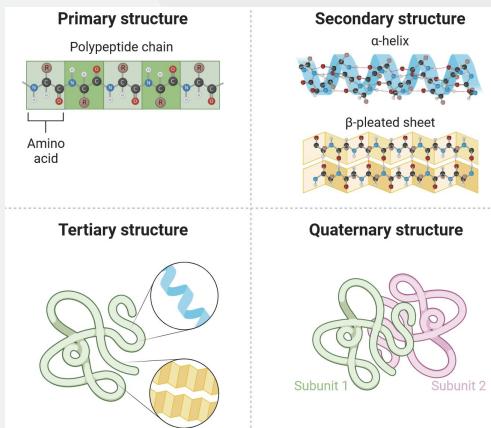
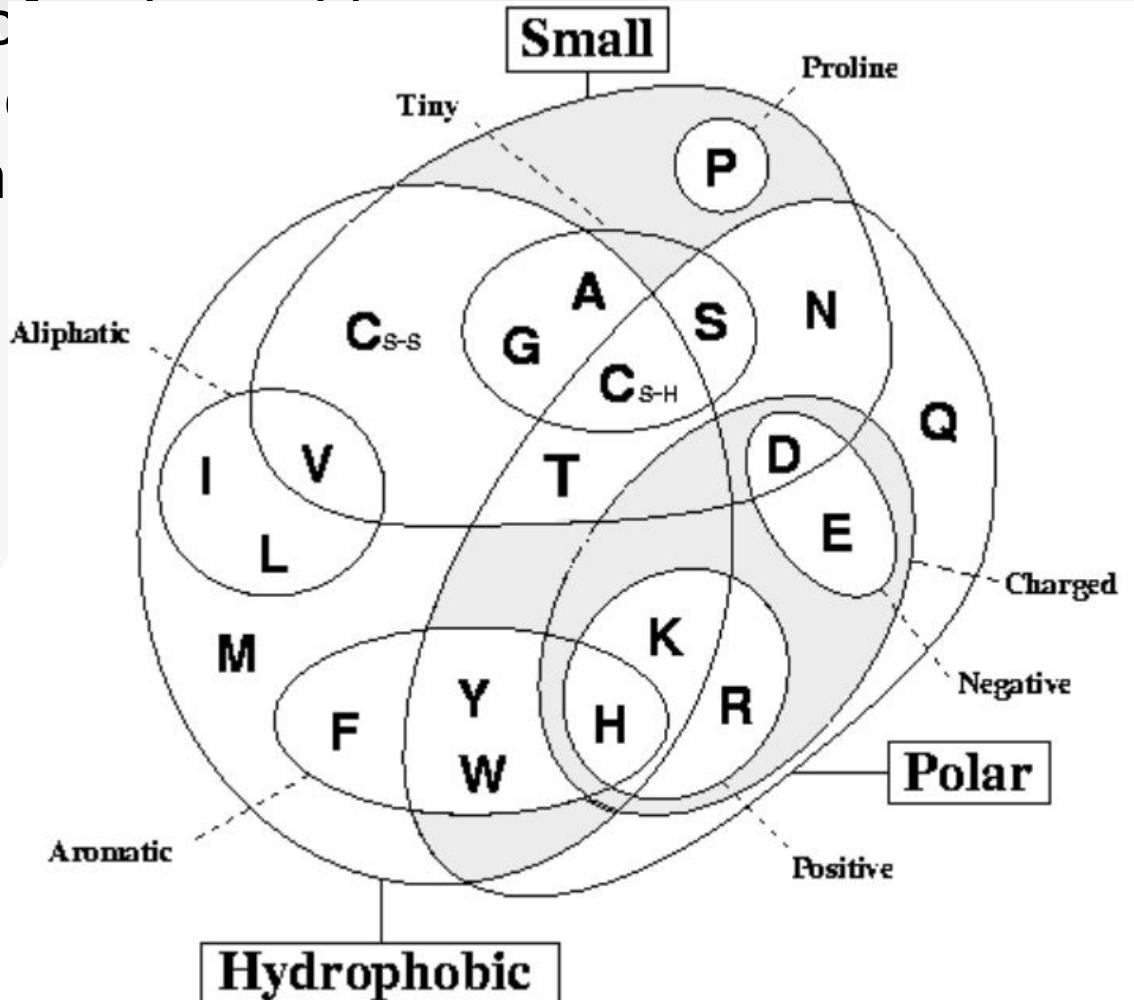
M	Met	Methionine	F	Phe	Phenylalanine
I	Ile	Isoleucine	W	Trp	Tryptophan
L	Leu	Leucine	Y	Tyr	Tyrosine
V	Val	Valine	G	Gly	Glycine
A	Ala	Alanine	P	Pro	Proline
C	Cys	Cysteine	S	Ser	Serine
D	Asp	Aspartic acid	T	Thr	Threonine
E	Glu	Glutamic acid	H	His	Histidine
N	Asn	Asparagine	K	Lys	Lysine
Q	Gln	Glutamine	R	Arg	Arginine
O	Pyl	Pyrrolysine	U	Sec	Selenocysteine
B	Asx	Asparagine or Aspartic acid	Z	Glx	Glutamine or Glutamic acid
J	Xle	Leucine or Isoleucine	X	Xaa	Any amino acid

http://www.h-invitational.jp/csmview/aa_table.html



Protein (data format) from NLP point of view

- protein == string of amino acids
- alphabet usually 20 letters that stands for amino acids (UZOB) are



X
no

Overview: <http://www.russelllab.org/aas/>

.PDB file format

1. header with information about the protein
2. amino acid sequence
3. structure coordination

HEADER 01-JUN-22
TITLE ALPHAFOOLD MONOMER V2.0 PREDICTION FOR SPOUT METHYLTRANSFERASE
TITLE 2 SUPERFAMILY PROTEIN (K1SW45)
COMPND MOL_ID: 1;
COMPND 2 MOLECULE: SPOUT METHYLTRANSFERASE SUPERFAMILY PROTEIN;
COMPND 3 CHAIN: A
SOURCE MOL_ID: 1;
SOURCE 2 ORGANISM_SCIENTIFIC: HUMAN GUT METAGENOME;
SOURCE 3 ORGANISM_TAXID: 408170
REMARK 1
REMARK 1 REFERENCE 1
REMARK 1 AUTH JOHN JUMPER, RICHARD EVANS, ALEXANDER PRITZEL, TIM GREEN,

...
SEQRES 1 A 147 MET LYS PHE THR LEU LEU VAL VAL GLY ARG THR VAL GLU
SEQRES 2 A 147 LYS HIS TYR ILE THR ALA ILE ASN ASP TYR VAL GLU ARG
SEQRES 3 A 147 THR ARG HIS PHE ILE SER PHE ASP MET GLU VAL ILE PRO
SEQRES 4 A 147 GLU LEU LYS ASN THR LYS ASN LEU SER MET GLU GLN GLN
SEQRES 5 A 147 LYS GLU LYS GLU GLY GLU LEU ILE LEU LYS ALA LEU LEU
SEQRES 6 A 147 PRO GLY ASP VAL VAL LEU LEU ASP GLU HIS GLY LYS
SEQRES 7 A 147 GLU PHE ARG SER VAL GLU PHE ALA ASN TRP ILE GLU ARG
SEQRES 8 A 147 LYS MET HIS THR VAL ASN LYS ARG LEU VAL PHE ILE ILE
SEQRES 9 A 147 GLY GLY PRO TYR GLY PHE ALA PRO LYS ILE TYR ASP ALA
SEQRES 10 A 147 ALA GLN GLU LYS ILE SER LEU SER LYS MET THR PHE SER
SEQRES 11 A 147 HIS GLN MET ILE ARG LEU ILE PHE VAL GLU GLN LEU TYR
SEQRES 12 A 147 ARG ALA MET THR

...
MODEL 1
ATOM 1 N MET A 1 6.248 6.075 -15.215 1.00 92.75 N
ATOM 2 CA MET A 1 5.220 5.606 -14.264 1.00 92.75 C
ATOM 3 C MET A 1 5.642 5.969 -12.845 1.00 92.75 C
ATOM 4 CB MET A 1 5.038 4.093 -14.419 1.00 92.75 C
ATOM 5 O MET A 1 6.834 5.932 -12.565 1.00 92.75 O
ATOM 6 CG MET A 1 4.081 3.491 -13.384 1.00 92.75 C
ATOM 7 SD MET A 1 4.031 1.700 -13.362 1.00 92.75 S
ATOM 8 CE MET A 1 5.762 1.344 -13.069 1.00 92.75 C
ATOM 9 N LYS A 2 4.691 6.320 -11.976 1.00 96.94 N
ATOM 10 CA LYS A 2 4.910 6.571 -10.545 1.00 96.94 C
ATOM 11 C LYS A 2 4.182 5.393 -9.797 1.00 96.94 C

.PDB file format

1. header with information about the protein
2. amino acid sequence
3. structure coordination

alpha-N atom of the first residue (**MET**) of peptide chain **A**



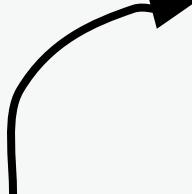
HEADER
TITLE ALPHAFOOLD MONOMER V2.0 PREDICTION FOR SPOUT METHYLTRANSFERASE
TITLE 2 SUPERFAMILY PROTEIN (K1SW45)
COMPND MOL_ID: 1;
COMPND 2 MOLECULE: SPOUT METHYLTRANSFERASE SUPERFAMILY PROTEIN;
COMPND 3 CHAIN: A
SOURCE MOL_ID: 1;
SOURCE 2 ORGANISM_SCIENTIFIC: HUMAN GUT METAGENOME;
SOURCE 3 ORGANISM_TAXID: 408170
REMARK 1
REMARK 1 REFERENCE 1
REMARK 1 AUTH JOHN JUMPER, RICHARD EVANS, ALEXANDER PRITZEL, TIM GREEN,

...
SEQRES 1 A 147 MET LYS PHE THR LEU LEU VAL VAL GLY ARG THR VAL GLU
SEQRES 2 A 147 LYS HIS TYR ILE THR ALA ILE ASN ASP TYR VAL GLU ARG
SEQRES 3 A 147 THR ARG HIS PHE ILE SER PHE ASP MET GLU VAL ILE PRO
SEQRES 4 A 147 GLU LEU LYS ASN THR LYS ASN LEU SER MET GLU GLN GLN
SEQRES 5 A 147 LYS GLU LYS GLU GLY GLU LEU ILE LEU LYS ALA LEU LEU
SEQRES 6 A 147 PRO GLY ASP VAL VAL VAL LEU LEU ASP GLU HIS GLY LYS
SEQRES 7 A 147 GLU PHE ARG SER VAL GLU PHE ALA ASN TRP ILE GLU ARG
SEQRES 8 A 147 LYS MET HIS THR VAL ASN LYS ARG LEU VAL PHE ILE ILE
SEQRES 9 A 147 GLY GLY PRO TYR GLY PHE ALA PRO LYS ILE TYR ASP ALA
SEQRES 10 A 147 ALA GLN GLU LYS ILE SER LEU SER LYS MET THR PHE SER
SEQRES 11 A 147 HIS GLN MET ILE ARG LEU ILE PHE VAL GLU GLN LEU TYR
SEQRES 12 A 147 ARG ALA MET THR
...

MODEL 1
ATOM 1 N MET A 1 6.248 6.075 -15.215 1.00 92.75 N
ATOM 2 CA MET A 1 5.220 5.606 -14.264 1.00 92.75 C
ATOM 3 C MET A 1 5.642 5.969 -12.845 1.00 92.75 C
ATOM 4 CB MET A 1 5.038 4.093 -14.419 1.00 92.75 C
ATOM 5 O MET A 1 6.834 5.932 -12.565 1.00 92.75 O
ATOM 6 CG MET A 1 4.081 3.491 -13.384 1.00 92.75 C
ATOM 7 SD MET A 1 4.031 1.700 -13.362 1.00 92.75 S
ATOM 8 CE MET A 1 5.762 1.344 -13.069 1.00 92.75 C
ATOM 9 N LYS A 2 4.691 6.320 -11.976 1.00 96.94 N
ATOM 10 CA LYS A 2 4.910 6.571 -10.545 1.00 96.94 C
ATOM 11 C LYS A 2 4.182 5.393 -9.797 1.00 96.94 C

.PDB file format

x, y, and z coordinates
(Angstrom units == 10^{-10} m)



MODEL	1	ATOM	1	N	MET	A	1	6.248	6.075	-15.215	1.00	92.75	N
		ATOM	2	CA	MET	A	1	5.220	5.606	-14.264	1.00	92.75	C
		ATOM	3	C	MET	A	1	5.642	5.969	-12.845	1.00	92.75	C
		ATOM	4	CB	MET	A	1	5.038	4.093	-14.419	1.00	92.75	C
		ATOM	5	O	MET	A	1	6.834	5.932	-12.565	1.00	92.75	O
		ATOM	6	CG	MET	A	1	4.081	3.491	-13.384	1.00	92.75	C
		ATOM	7	SD	MET	A	1	4.031	1.700	-13.362	1.00	92.75	S
		ATOM	8	CE	MET	A	1	5.762	1.344	-13.069	1.00	92.75	C
		ATOM	9	N	LYS	A	2	4.691	6.320	-11.976	1.00	96.94	N
		ATOM	10	CA	LYS	A	2	4.910	6.571	-10.545	1.00	96.94	C
		ATOM	11	C	LYS	A	2	4.102	5.393	-9.707	1.00	96.94	C

alpha-N atom of the
first residue (MET)
of peptide chain A

.PDB file format

x, y, and z coordinates
(Angstrom units == 10^{-10} m)

...

MODEL	1	x, y, and z coordinates (Angstrom units == 10^{-10} m)								
ATOM	1	N	MET	A	1	6.248	6.075	-15.215	1.00 92.75	N
ATOM	2	CA	MET	A	1	5.220	5.606	-14.264	1.00 92.75	C
ATOM	3	C	MET	A	1	5.642	5.969	-12.845	1.00 92.75	C
ATOM	4	CB	MET	A	1	5.038	4.093	-14.419	1.00 92.75	C
ATOM	5	O	MET	A	1	6.834	5.932	-12.565	1.00 92.75	O
ATOM	6	CG	MET	A	1	4.081	3.491	-13.384	1.00 92.75	C
ATOM	7	SD	MET	A	1	4.031	1.700	-13.362	1.00 92.75	S
ATOM	8	CE	MET	A	1	5.762	1.344	-13.069	1.00 92.75	C
ATOM	9	N	LYS	A	2	4.691	6.320	-11.976	1.00 96.94	N
ATOM	10	CA	LYS	A	2	4.910	6.571	-10.545	1.00 96.94	C
ATOM	11	C	LYS	A	2	4.102	5.393	-9.707	1.00 96.94	C

← element name

alpha-N atom of the first residue (MET) of peptide chain A

.PDB file format

alpha-N atom of the first residue (MET) of peptide chain A

MODEL	1	...									
ATOM	1	N	MET	A	1	6.248	6.075	-15.215	1.00	92.75	N
ATOM	2	CA	MET	A	1	5.220	5.606	-14.264	1.00	92.75	C
ATOM	3	C	MET	A	1	5.642	5.969	-12.845	1.00	92.75	C
ATOM	4	CB	MET	A	1	5.038	4.093	-14.419	1.00	92.75	C
ATOM	5	O	MET	A	1	6.834	5.932	-12.565	1.00	92.75	O
ATOM	6	CG	MET	A	1	4.081	3.491	-13.384	1.00	92.75	C
ATOM	7	SD	MET	A	1	4.031	1.700	-13.362	1.00	92.75	S
ATOM	8	CE	MET	A	1	5.762	1.344	-13.069	1.00	92.75	C
ATOM	9	N	LYS	A	2	4.691	6.320	-11.976	1.00	96.94	N
ATOM	10	CA	LYS	A	2	4.910	6.571	-10.545	1.00	96.94	C
ATOM	11	C	LYS	A	2	4.102	5.393	-9.707	1.00	96.94	C

x, y, and z coordinates
(Angstrom units == 10^{-10} m)

element name

temperature
== a measure of
confidence in the
location of atom

.PDB file format

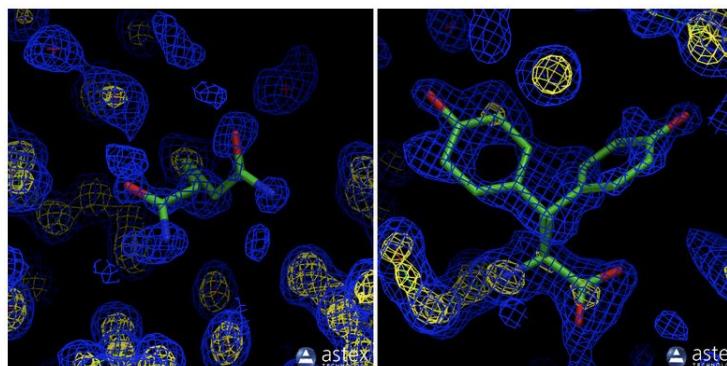
alpha-N atom of the first residue (MET) of peptide chain A

MODEL	1	x, y, and z coordinates (Angstrom units == 10^{-10} m)										
ATOM	1	N	MET	A	1	6.248	6.075	-15.215	1.00	92.75	N	
ATOM	2	CA	MET	A	1	5.220	5.606	-14.264	1.00	92.75	C	
ATOM	3	C	MET	A	1	5.642	5.969	-12.845	1.00	92.75	C	
ATOM	4	CB	MET	A	1	5.038	4.093	-14.419	1.00	92.75	C	
ATOM	5	O	MET	A	1	6.834	5.932	-12.565	1.00	92.75	O	
ATOM	6	CG	MET	A	1	4.081	3.491	-13.384	1.00	92.75	C	
ATOM	7	SD	MET	A	1	4.031	1.700	-13.362	1.00	92.75	S	
ATOM	8	CE	MET	A	1	5.762	1.344	-13.069	1.00	92.75	C	
ATOM	9	N	LYS	A	2	4.691	6.320	-11.976	1.00	96.94	N	
ATOM	10	CA	LYS	A	2	4.910	6.571	-10.545	1.00	96.94	C	
ATOM	11	C	LYS	A	2	4.102	5.393	-9.707	1.00	96.94	C	

occupancy

== 1 if atom is found in the same place in the crystal in all molecules

temperature
== a measure of confidence in the location of atom



alternate conformations
in myoglobin

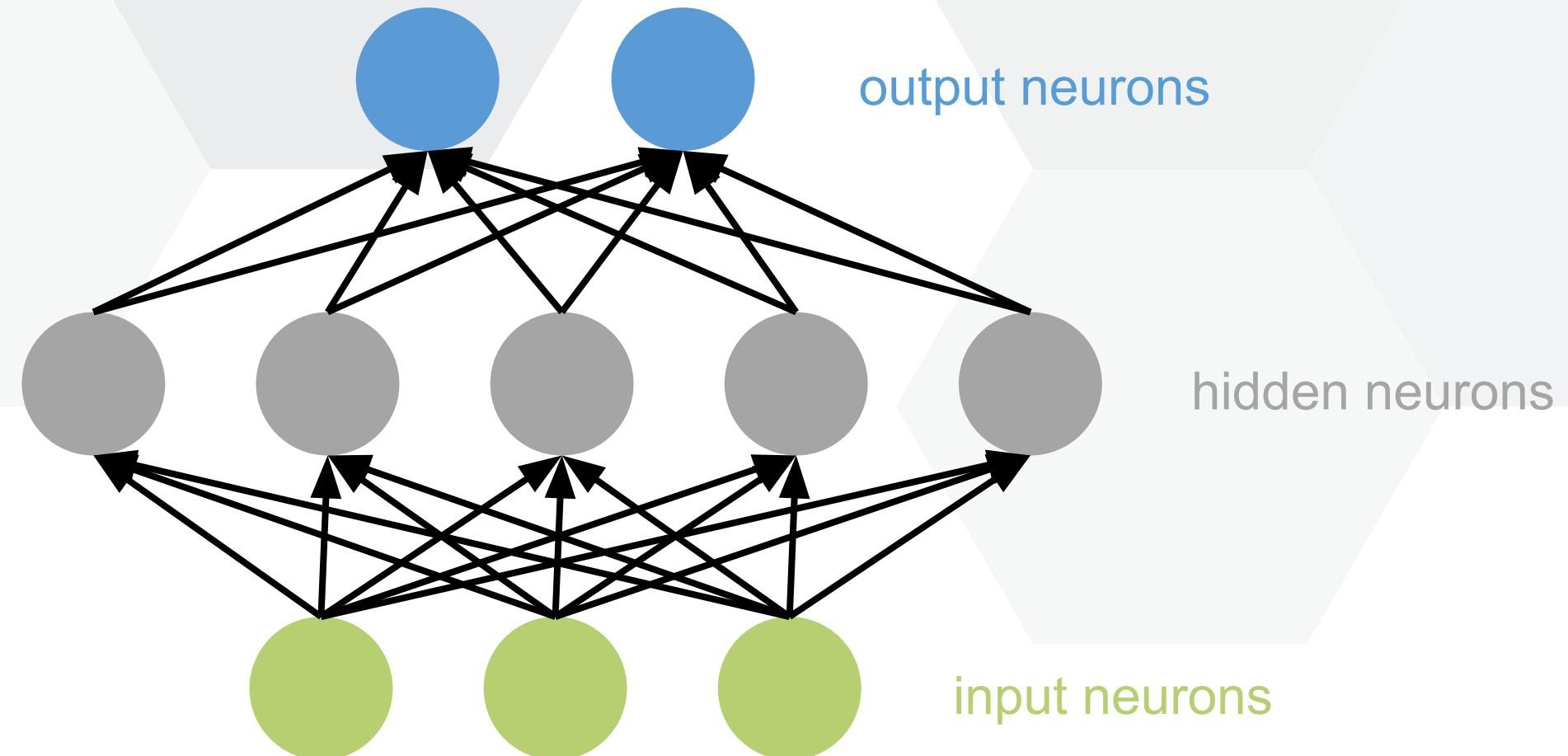
[https://pdb101.rcsb.org/learn/guide-to-understanding-pdb-data/
dealing-with-coordinates](https://pdb101.rcsb.org/learn/guide-to-understanding-pdb-data/dealing-with-coordinates)

Models for solving the protein folding problem

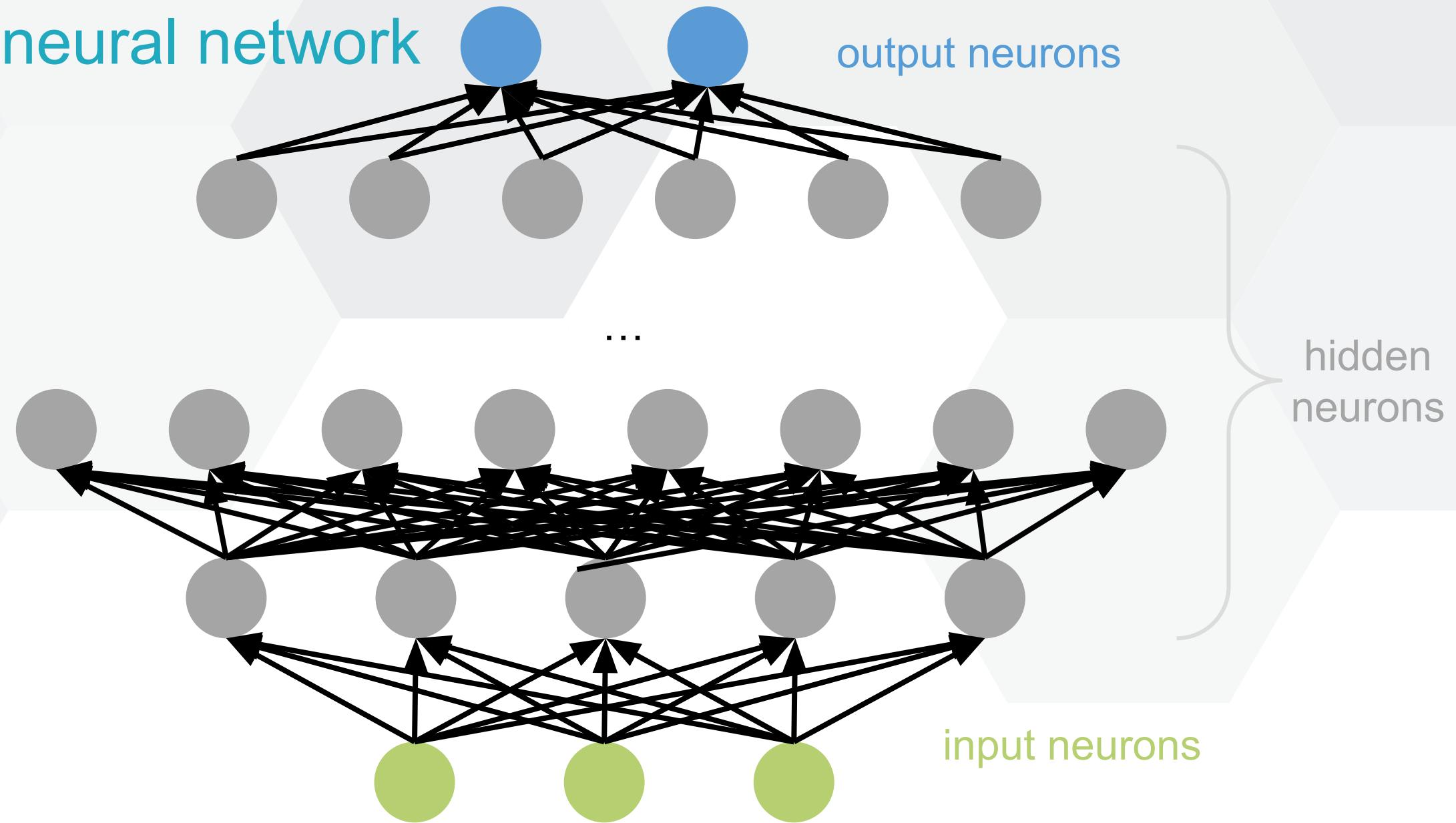
Model

== a machine learning approach that has been trained to solve some specific task

Multi-layer
Perceptron

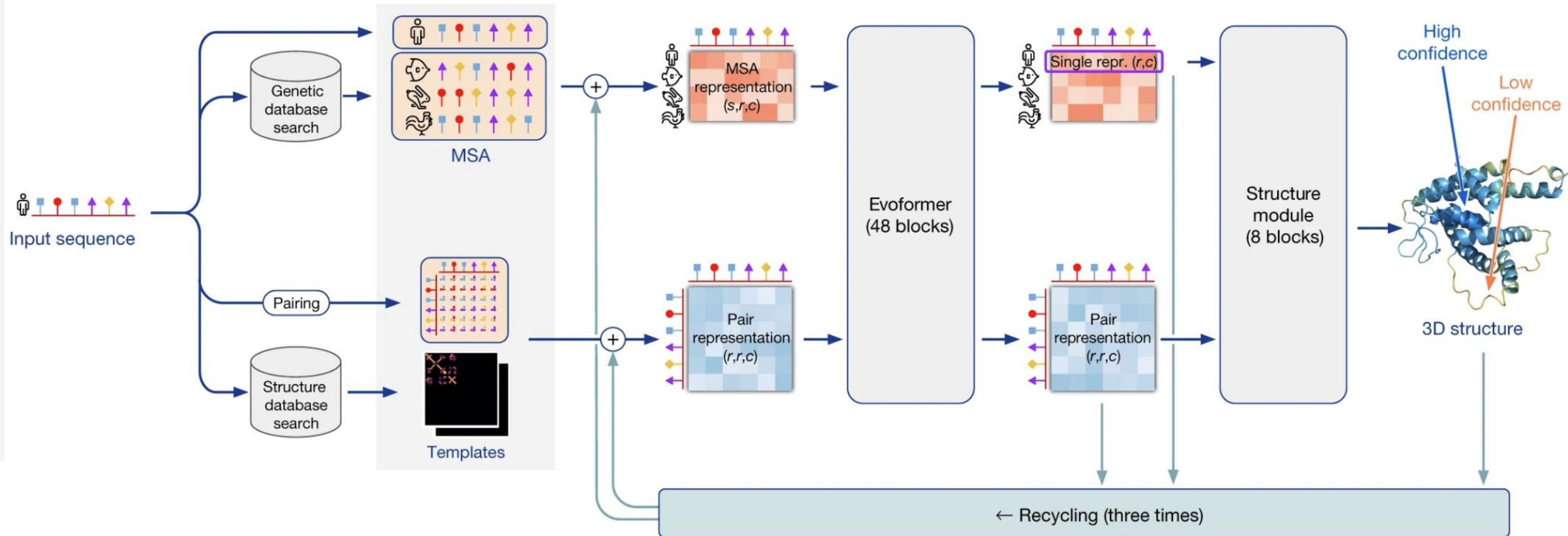


Deep neural network



AlphaFold2 [1]

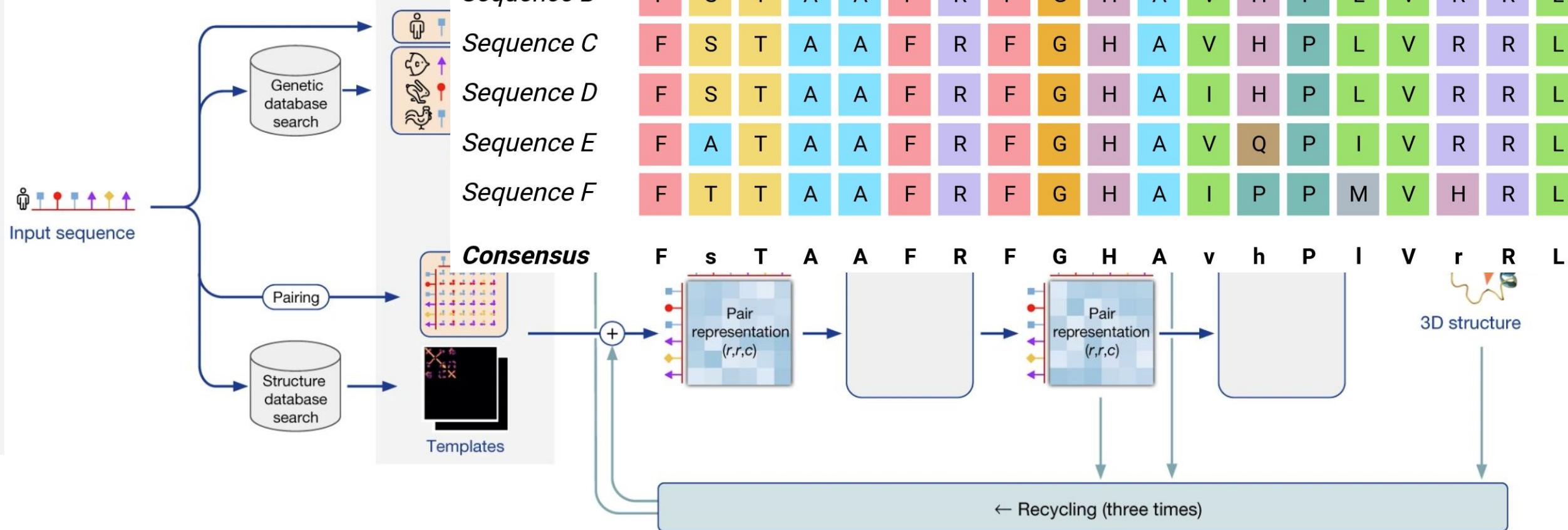
- orphan proteins



Multiple Sequence Alignment

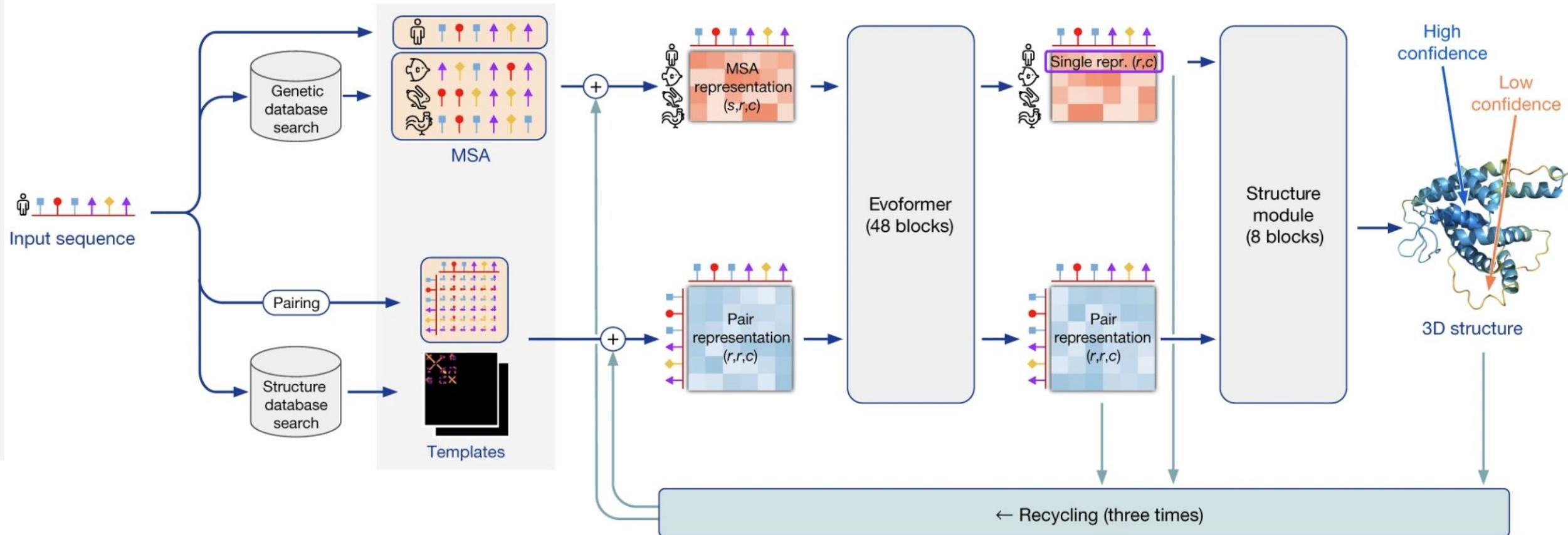
AlphaFold2 [1]

- orphan proteins



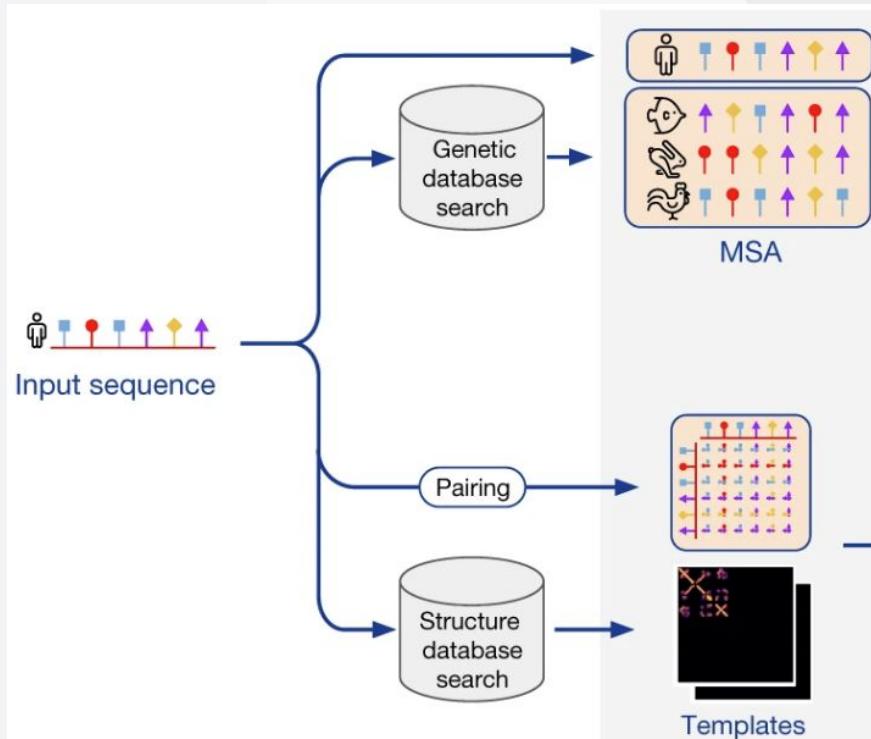
AlphaFold2 [1]

- orphan proteins



AlphaFold2 [1]

- orphan proteins



AlphaFold2 fails to predict protein fold switching

Devlina Chakravarty ¹, Lauren L Porter ^{1,2}

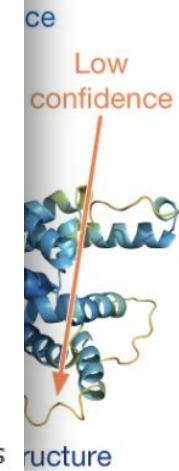
Affiliations + expand

PMID: 35634782 PMCID: PMC9134877 DOI: 10.1002/pro.4353

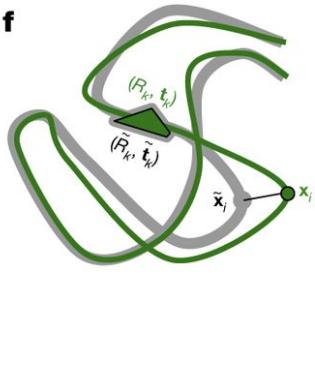
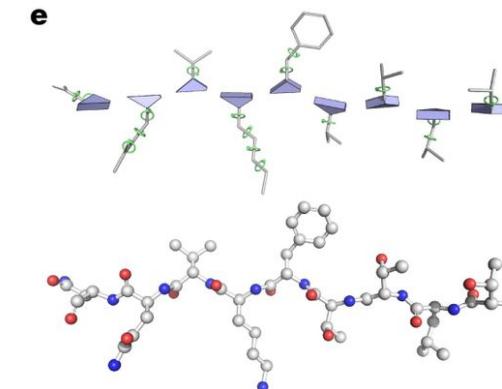
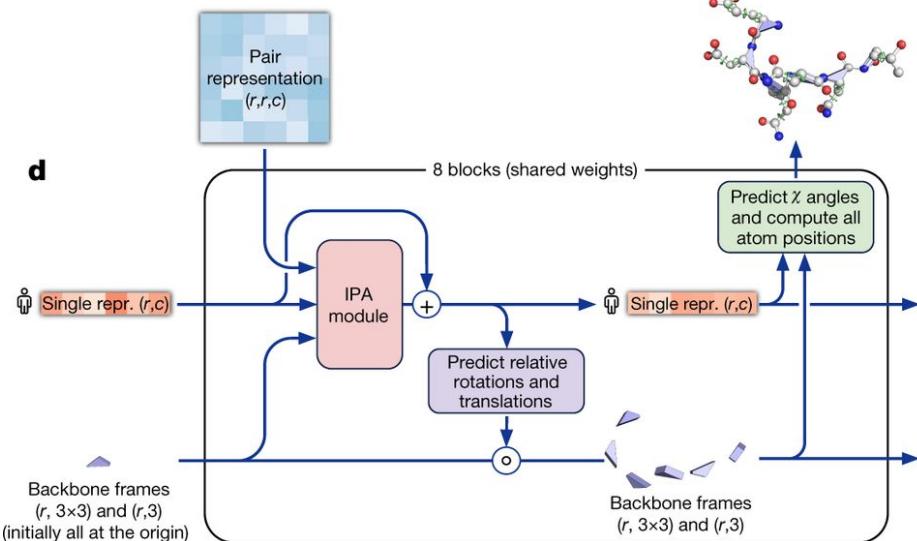
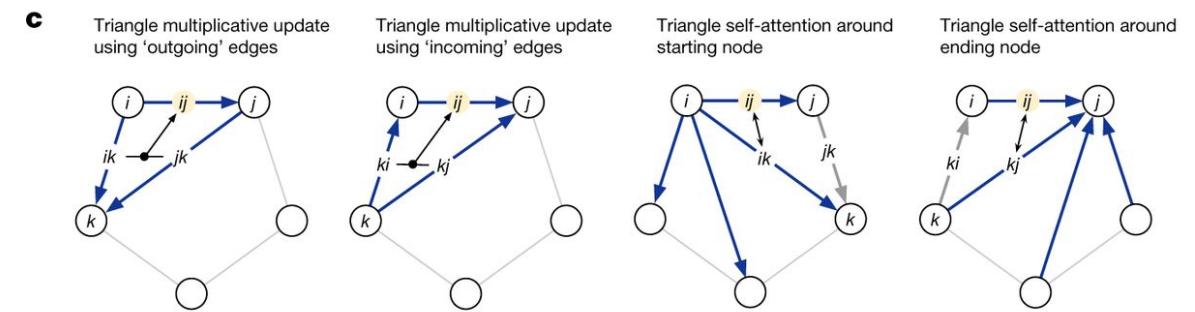
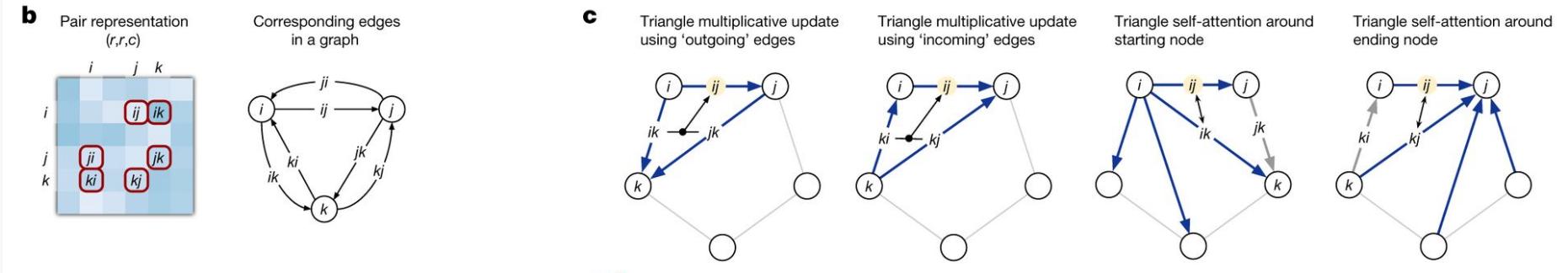
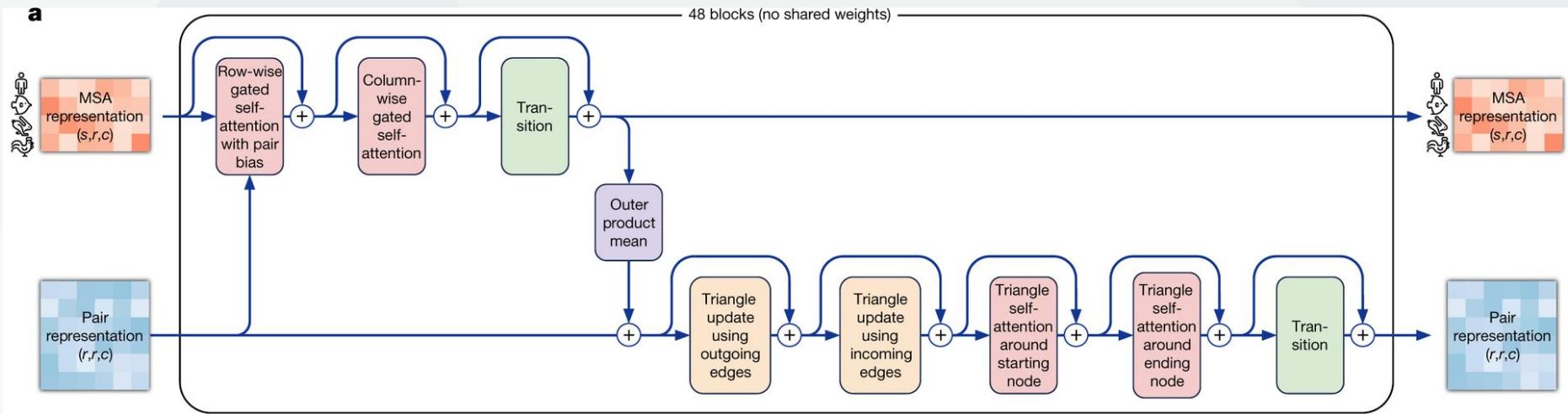
Free PMC article

Abstract

AlphaFold2 has revolutionized protein structure prediction by leveraging sequence information to rapidly model protein folds with atomic-level accuracy. Nevertheless, previous work has shown that these predictions tend to be inaccurate for structurally heterogeneous proteins. To systematically assess factors that contribute to this inaccuracy, we tested AlphaFold2's performance on 98-fold-switching proteins, which assume at least two distinct-yet-stable secondary and tertiary structures. Topological similarities were quantified between five predicted and two experimentally determined structures of each fold-switching protein. Overall, 94% of AlphaFold2 predictions captured one experimentally determined conformation but not the other. Despite these biased results, AlphaFold2's



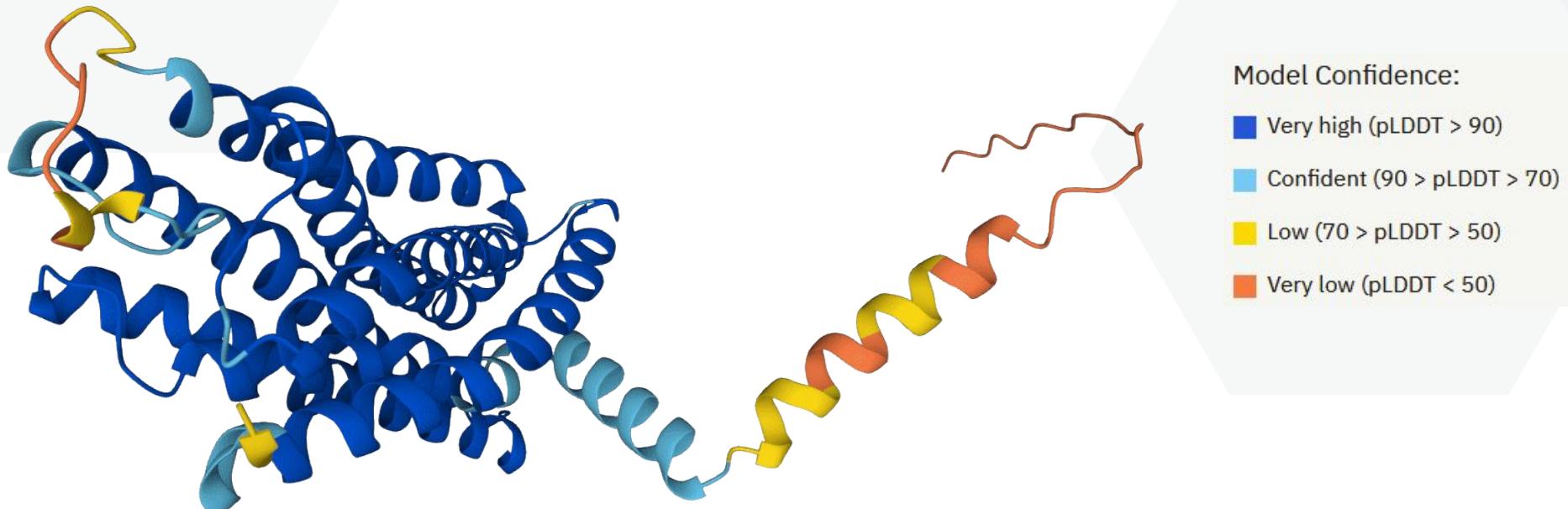
AlphaFold2



Prediction confidence [1]

pLDDT == per-residue estimate of its confidence on a scale from 0 - 100

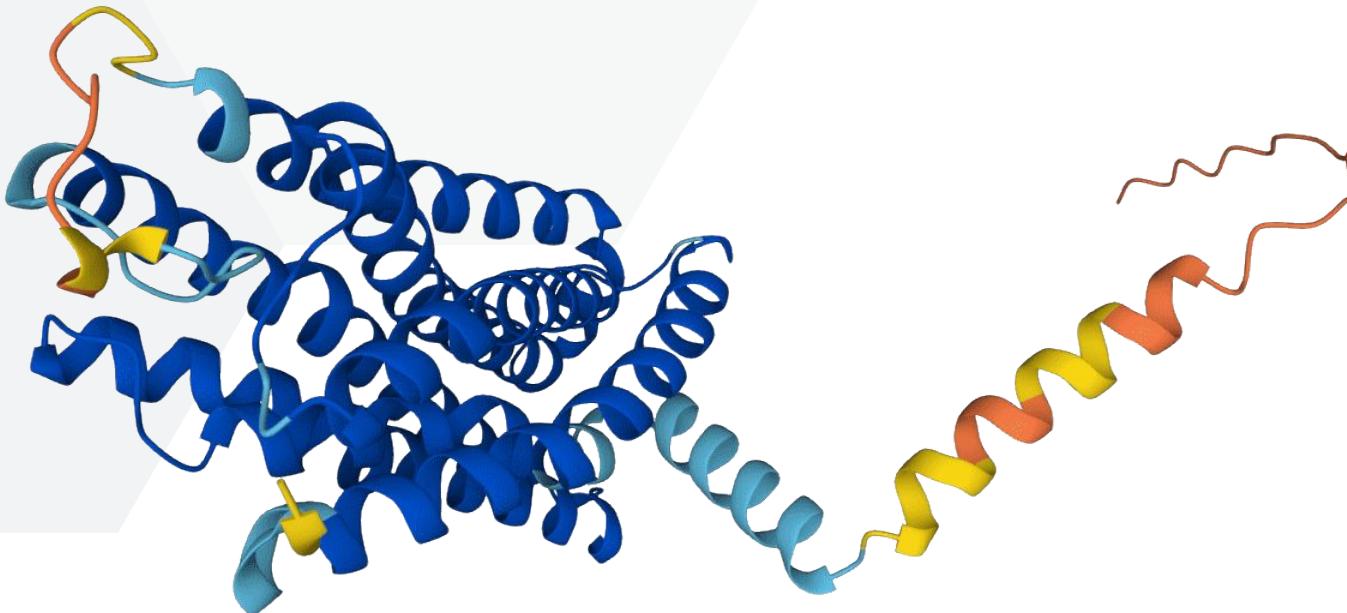
- pLDDT > 90 is considered to be modelled to high accuracy
- pLDDT < 50 often have a ribbon-like appearance and should not be interpreted
 - *suggests that such region is either unstructured in physiological conditions or only structured as a part of a complex (analysed on well-studied proteins)*



Prediction confidence

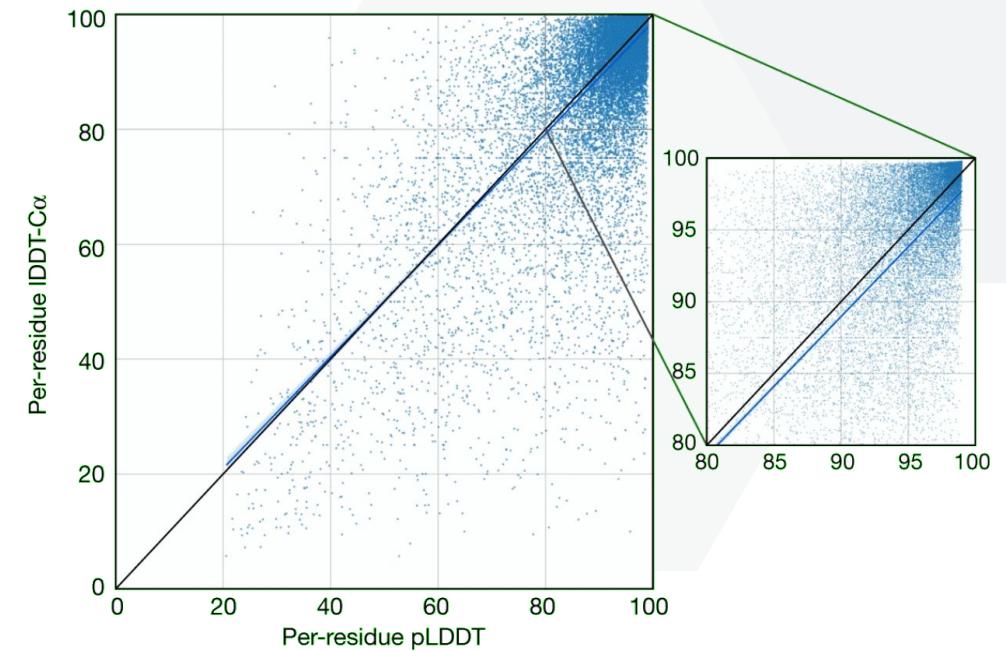
pLDDT == per-residue estimate of its confidence on a scale from 0 - 100

- how well the prediction would agree with an experimental structure based on the local distance difference test C α (IDDT-C α)

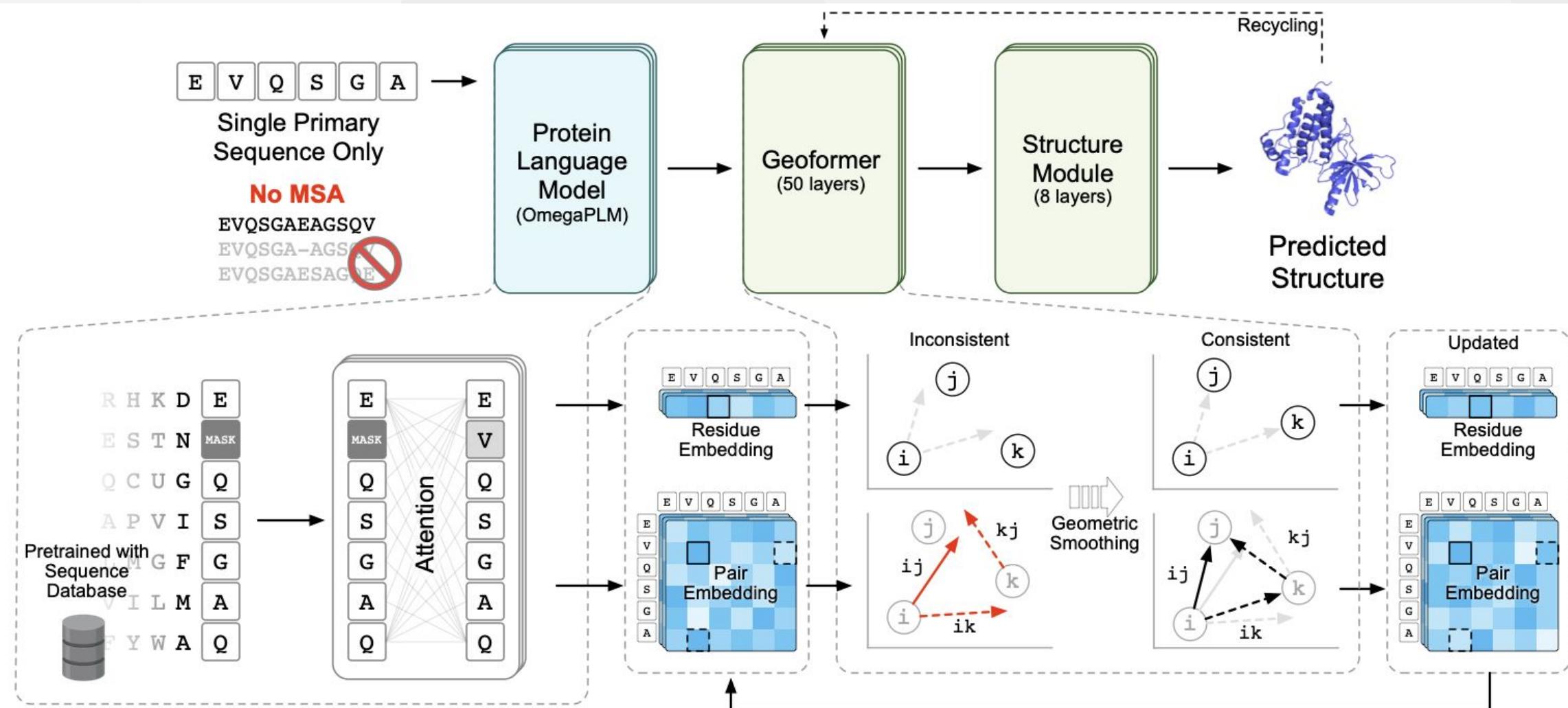


<https://alphafold.ebi.ac.uk/entry/Q8VCK6>

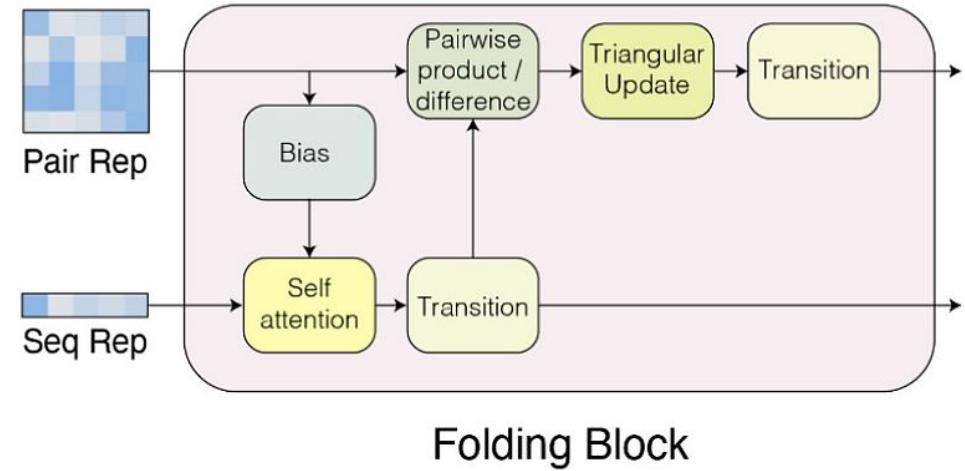
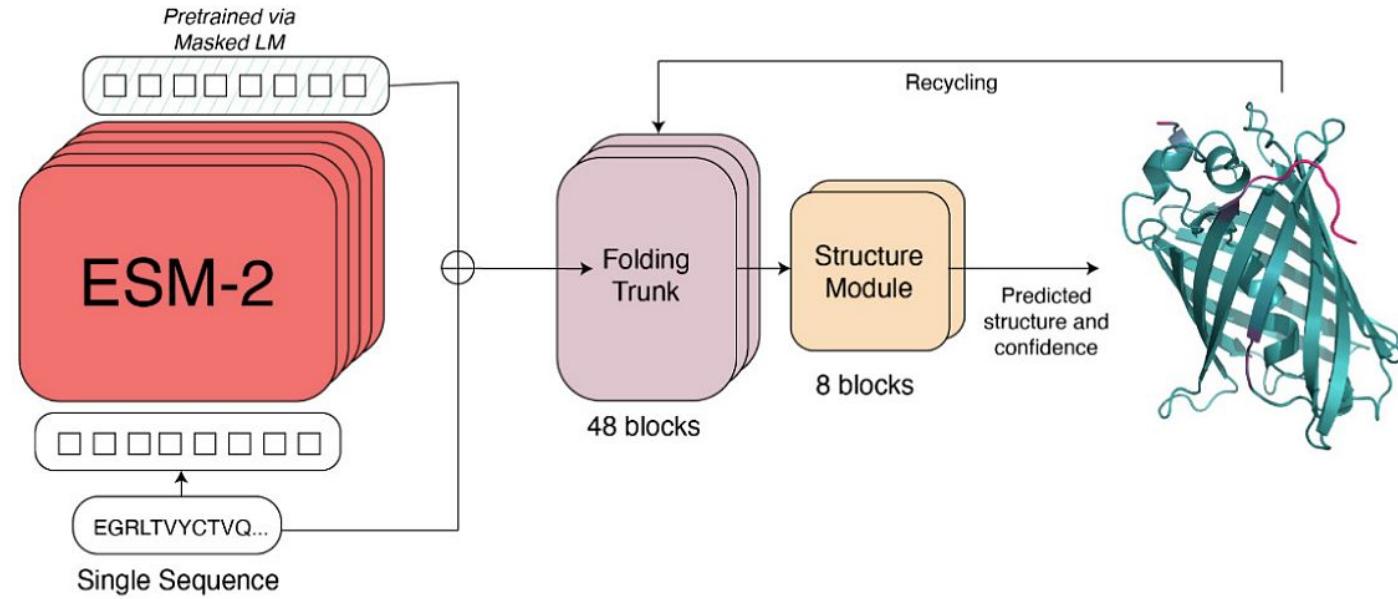
correlation between per-residue pLDDT and IDDT-C α :



OmegaFold [2]



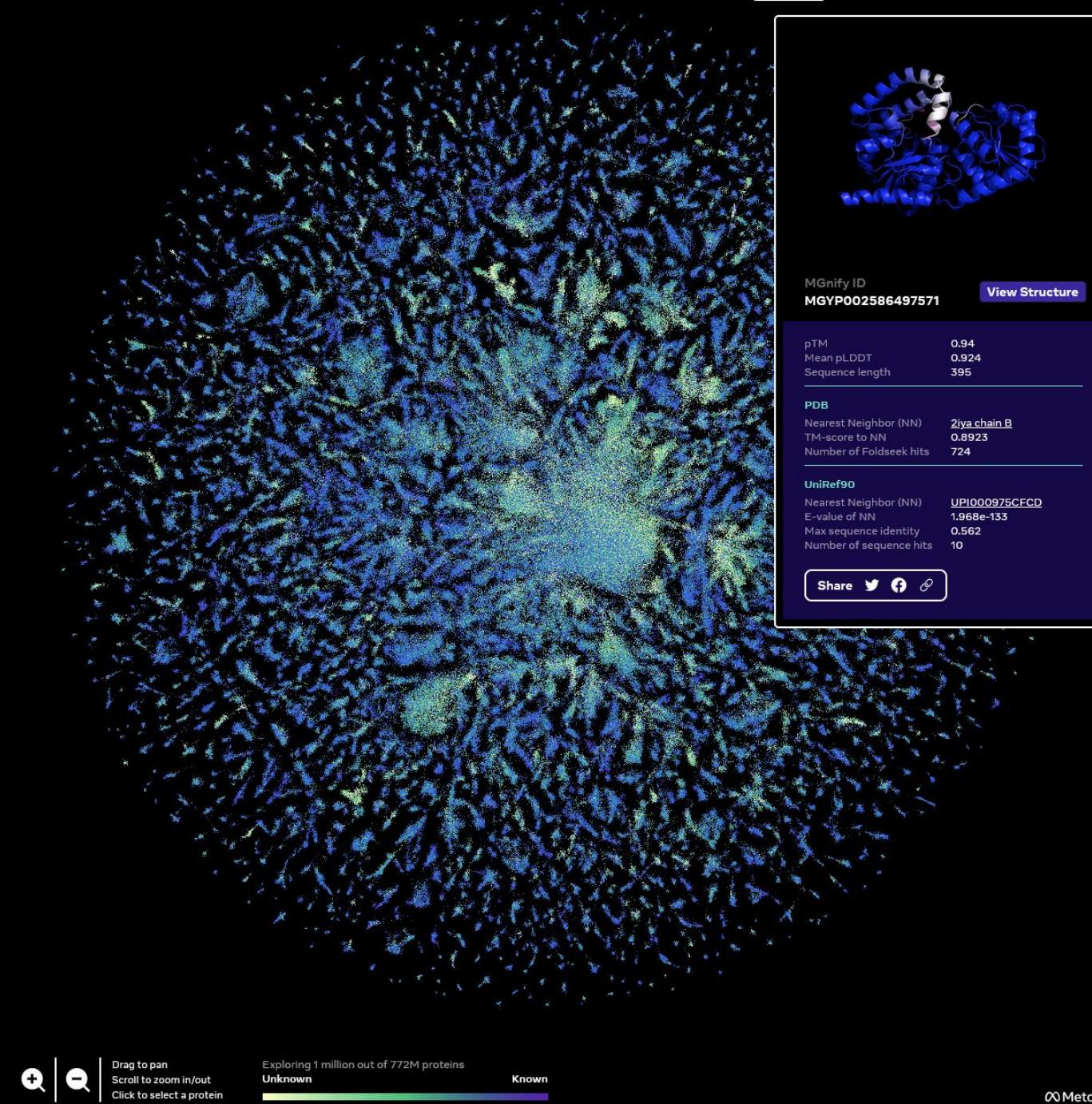
ESMFold [3]





ESMFold

<https://esmatlas.com/>



Comparison of the models [1,2,3,4]

- benchmark results:

AlphaFold (ColabFold)					OmegaFold					ESMFold				
Length	Running time	pLDDT	CPU memory	GPU memory	Length	Running time	pLDDT	CPU memory	GPU memory	Length	Running time	pLDDT	CPU memory	GPU memory
50	45	0.89	10 GB	10 GB	50	3.66	0.86	10 GB	6 GB	50	1	0.84	13 GB	16 GB
					100	7.42	0.39	10 GB	7 GB					
					200	34.07	0.65	10 GB	8.5 GB					
					400	110	0.76	10 GB	10 GB					
					800	1425	0.53	10 GB	11 GB					
					1600	FAILED	-	-	17 GB					
1600	2800	0.41	10 GB	10 GB						1600	FAILED	-	-	24 GB

- since AlphaFold utilizes MSA, it is not accurate on orphan proteins, since ESMFold leverages the power of transformer models, it effectively addresses the “twilight zone” problem

predicted local distance difference test (pLDDT) = score to measure model's prediction accuracy (1 == perfect, 0 == incorrect)

Comparison of the models [1,2,3,4]

- benchmark results:

AlphaFold (ColabFold)					OmegaFold					ESMFold				
Length	Running time	pLDDT	CPU memory	GPU memory	Length	Running time	pLDDT	CPU memory	GPU memory	Length	Running time	pLDDT	CPU memory	GPU memory
50	45	0.89	10 GB	10 GB	50 100 200 400 800 1600	3.66	0.86	10 GB	6 GB	50 100 200 400 800 1600	1	0.84	13 GB	16 GB
100	55	0.38	10 GB	10 GB		7.42	0.39	10 GB	7 GB		1	0.3	13 GB	16 GB
200	91	0.55	10 GB	10 GB		34.07	0.65	10 GB	8.5 GB		4	0.77	13 GB	16 GB
400	210	0.82	10 GB	10 GB		110	0.76	10 GB	10 GB		20	0.93	13 GB	18 GB
800	810	0.54	10 GB	10 GB		1425	0.53	10 GB	11 GB		125	0.66	13 GB	20 GB
1600	2800	0.41	10 GB	10 GB		FAILED	-	-	17 GB		FAILED	-	-	24 GB

- since AlphaFold utilizes MSA, it is not accurate on orphan proteins, since ESMFold leverages the power of transformer models, it effectively addresses the “twilight zone” problem

predicted local distance difference test (pLDDT) = score to measure model's prediction accuracy (1 == perfect, 0 == incorrect)

Comparison of the models [1,2,3,4]

- benchmark results:

AlphaFold (ColabFold)					OmegaFold					ESMFold				
Length	Running time	pLDDT	CPU memory	GPU memory	Length	Running time	pLDDT	CPU memory	GPU memory	Length	Running time	pLDDT	CPU memory	GPU memory
50	45	0.89	10 GB	10 GB	50 100 200 400 800 1600	3.66	0.86	10 GB	6 GB	50 100 200 400 800 1600	1	0.84	13 GB	16 GB
100	55	0.38	10 GB	10 GB		7.42	0.39	10 GB	7 GB		1	0.3	13 GB	16 GB
200	91	0.55	10 GB	10 GB		34.07	0.65	10 GB	8.5 GB		4	0.77	13 GB	16 GB
400	210	0.82	10 GB	10 GB		110	0.76	10 GB	10 GB		20	0.93	13 GB	18 GB
800	810	0.54	10 GB	10 GB		1425	0.53	10 GB	11 GB		125	0.66	13 GB	20 GB
1600	2800	0.41	10 GB	10 GB		FAILED	-	-	17 GB		FAILED	-	-	24 GB

- since AlphaFold utilizes MSA, it is not accurate on orphan proteins, since ESMFold leverages the power of transformer models, it effectively addresses the “twilight zone” problem

predicted local distance difference test (pLDDT) = score to measure model's prediction accuracy (1 == perfect, 0 == incorrect)

Comparison of the models [1,2,3,4]

- benchmark results:

AlphaFold (ColabFold)					OmegaFold					ESMFold				
Length	Running time	pLDDT	CPU memory	GPU memory	Length	Running time	pLDDT	CPU memory	GPU memory	Length	Running time	pLDDT	CPU memory	GPU memory
50	45	0.89	10 GB	10 GB	50 100 200 400 800 1600	3.66	0.86	10 GB	6 GB	50 100 200 400 800 1600	1	0.84	13 GB	16 GB
100	55	0.38	10 GB	10 GB		7.42	0.39	10 GB	7 GB		1	0.3	13 GB	16 GB
200	91	0.55	10 GB	10 GB		34.07	0.65	10 GB	8.5 GB		4	0.77	13 GB	16 GB
400	210	0.82	10 GB	10 GB		110	0.76	10 GB	10 GB		20	0.93	13 GB	18 GB
800	810	0.54	10 GB	10 GB		1425	0.53	10 GB	11 GB		125	0.66	13 GB	20 GB
1600	2800	0.41	10 GB	10 GB		FAILED	-	-	17 GB		FAILED	-	-	24 GB

- since AlphaFold utilizes MSA, it is not accurate on orphan proteins, since ESMFold leverages the power of transformer models, it effectively addresses the “twilight zone” problem

predicted local distance difference test (pLDDT) = score to measure model's prediction accuracy (1 == perfect, 0 == incorrect)

Other folding tools

https://github.com/biolists/folding_tools

Predictors

[in alphabetical order]

- **MSA-based** (uses Multiple Sequence Alignments (MSAs) as input)

- AlphaFold2 [repo](#) [JAX](#) [DOI](#) [10.1038/s41586-021-03819-2](https://doi.org/10.1038/s41586-021-03819-2)

- The original AlphaFold 2 method
 - Features: monomer, multimer
 - Other: [Colab Notebook](#)

- ColabFold [repo](#) [JAX](#) [DOI](#) [10.1038/s41592-022-01488-1](https://doi.org/10.1038/s41592-022-01488-1)

- Faster AF2 compiling and MSA generations
 - Features: monomer, multimer
 - Other: [localcolabfold](#)

- FastFold [repo](#) [PyTorch](#) [arxiv](#) [2203.00854](https://arxiv.org/abs/2203.00854)

- Runtime improvements to OpenFold (see below)
 - Features: monomer

- HelixFold [repo](#) [PaddlePaddle](#) [arxiv](#) [2207.05477](https://arxiv.org/abs/2207.05477)

- Reimplementation of AF2 in PaddlePaddle
 - Features: monomer

- **pLM-based** (using embeddings from protein Language Models (pLMs) as input)

- ESM-Fold [repo](#) [PyTorch](#) [DOI](#) [10.1101/2022.07.20.500902](https://doi.org/10.1101/2022.07.20.500902)

- Features: monomer
 - Other: [Unofficial Colab notebook](#), [API](#)

- EMBER3D [repo](#) [PyTorch](#)

- Features: monomer

- HelixFold-single [repo](#) [PaddlePaddle](#) [arxiv](#) [2207.13921](https://arxiv.org/abs/2207.13921)

- Features: monomer
 - Resource: <https://paddlehelix.baidu.com/app/drug/protein-single/forecast>

- IgFold [repo](#) [PyTorch](#) [DOI](#) [10.1101/2022.04.20.488972](https://doi.org/10.1101/2022.04.20.488972)

- pLM focused on antibody sequences
 - Features: monomer
 - Other: [Colab Notebook](#)

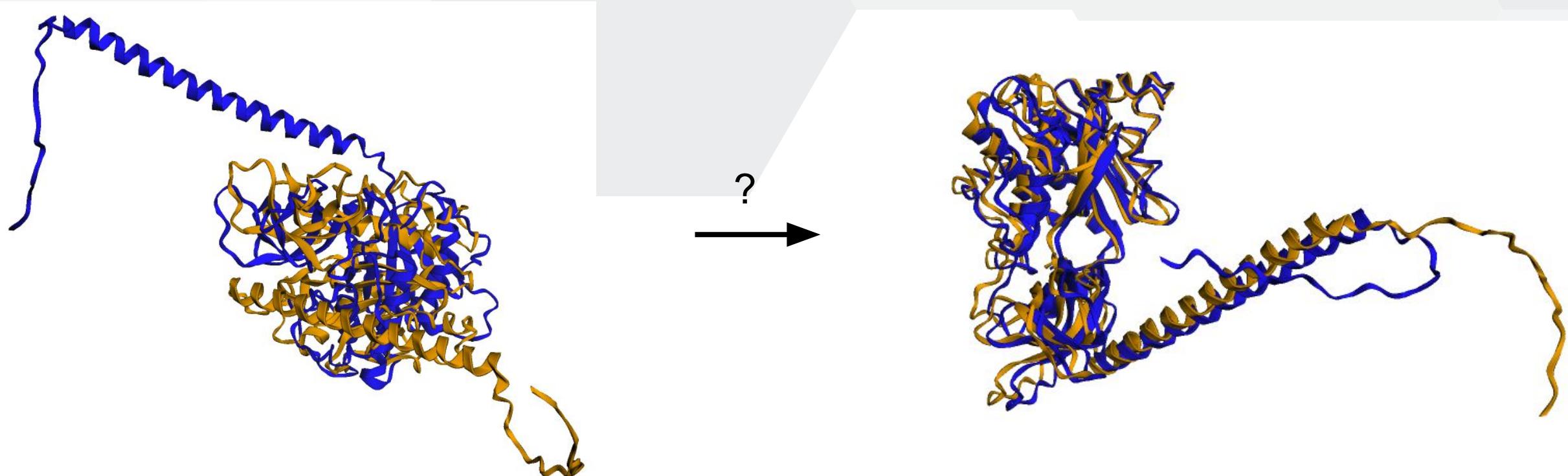
- OmegaFold [repo](#) [PyTorch](#) [DOI](#) [10.1101/2022.07.21.500999](https://doi.org/10.1101/2022.07.21.500999)

- Features: monomer
 - Other: [Unofficial Colab Notebook](#), [\[tweet\]](#) Martin comparing structures, [\[tweet\]](#) Sergey's positional encoding observation

- RGN2 [Github](#)

- Uses AlphaFold's structure module (AF2Rank's template protocol) for final structure refinement.

How can we compare two 3D structures?



RMSD (root-mean-square deviation)

== measure of the average distance between the atoms (usually the backbone atoms) of superimposed proteins

$$RMSD = \sqrt{\frac{1}{N} \sum_{i=1}^N \delta_i^2}$$

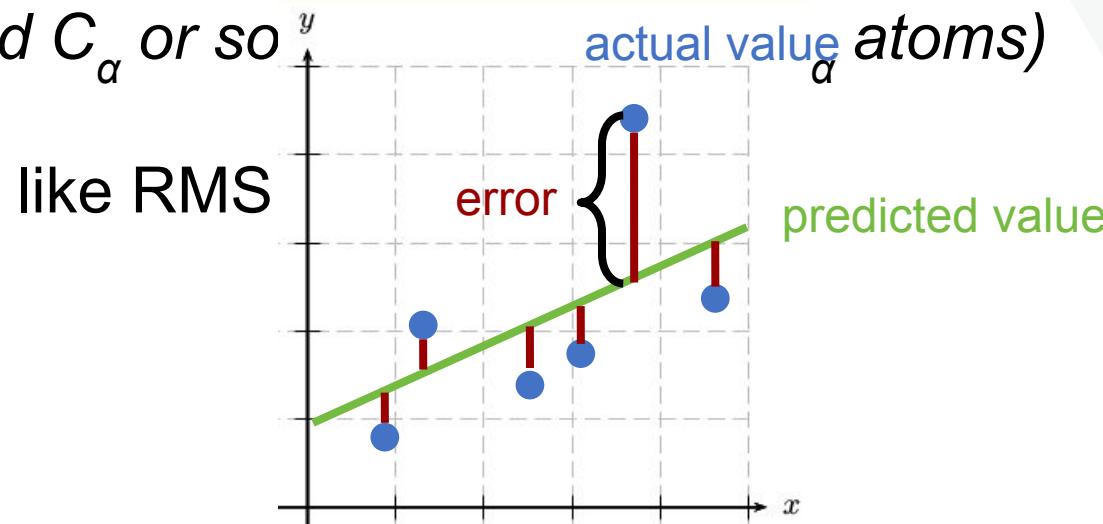
where δ_i is the distance between atom i and either a reference structure or the mean position of the N equivalent atoms (*often calculated for the backbone heavy atoms C, N, O, and C_α or sometimes just the C_α atoms*)

RMSD (root-mean-square deviation)

== measure of the average distance between the atoms (usually the backbone atoms) of superimposed proteins

$$RMSD = \sqrt{\frac{1}{N} \sum_{i=1}^N \delta_i^2}$$

where δ_i is the distance between atom i and either a reference structure or the mean position of the N equivalent atoms (*often calculated for the backbone heavy atoms C, N, O, and C_α or so*)

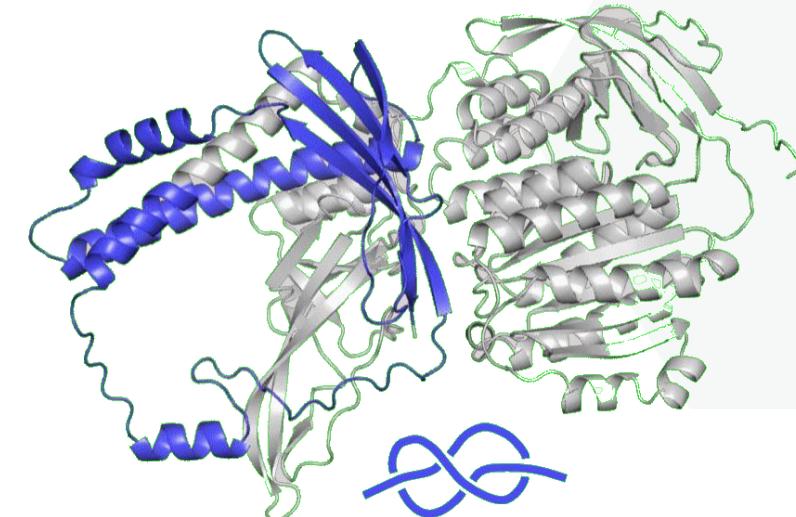
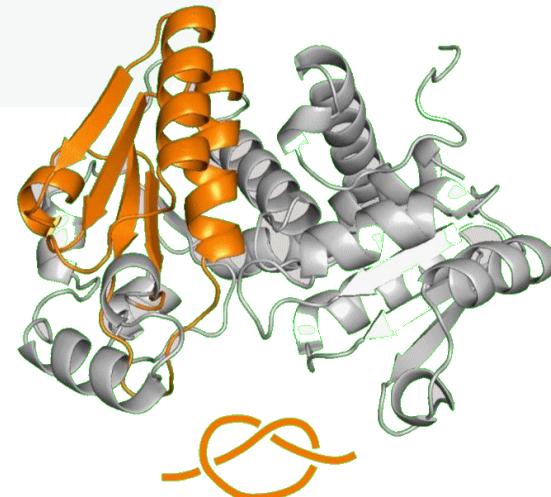


Exercises

- AlphaFold2** - maltaomics_ex0a_alphafold.ipynb
- OmegaFold** - maltaomics_ex0b_omegafold.ipynb
- ESM** - maltaomics_ex0c_esmfold.ipynb

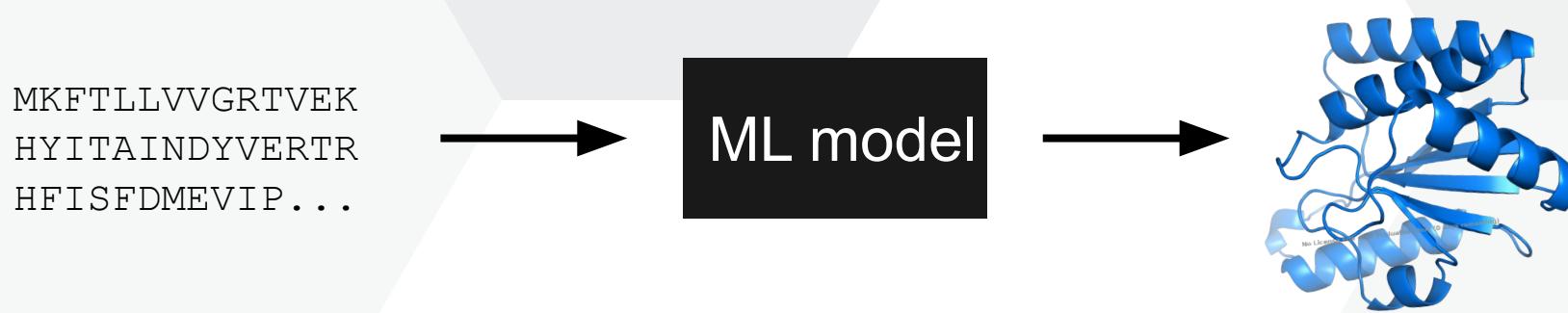
Other protein structure problems

- structure property
 - predict from amino acid sequence whether it's folded structure possesses the property
 - interpret what motif is responsible for the property occurrence
- ideally in future: flip the problem and design new proteins -> get amino acid sequence from a structure that has some wanted property



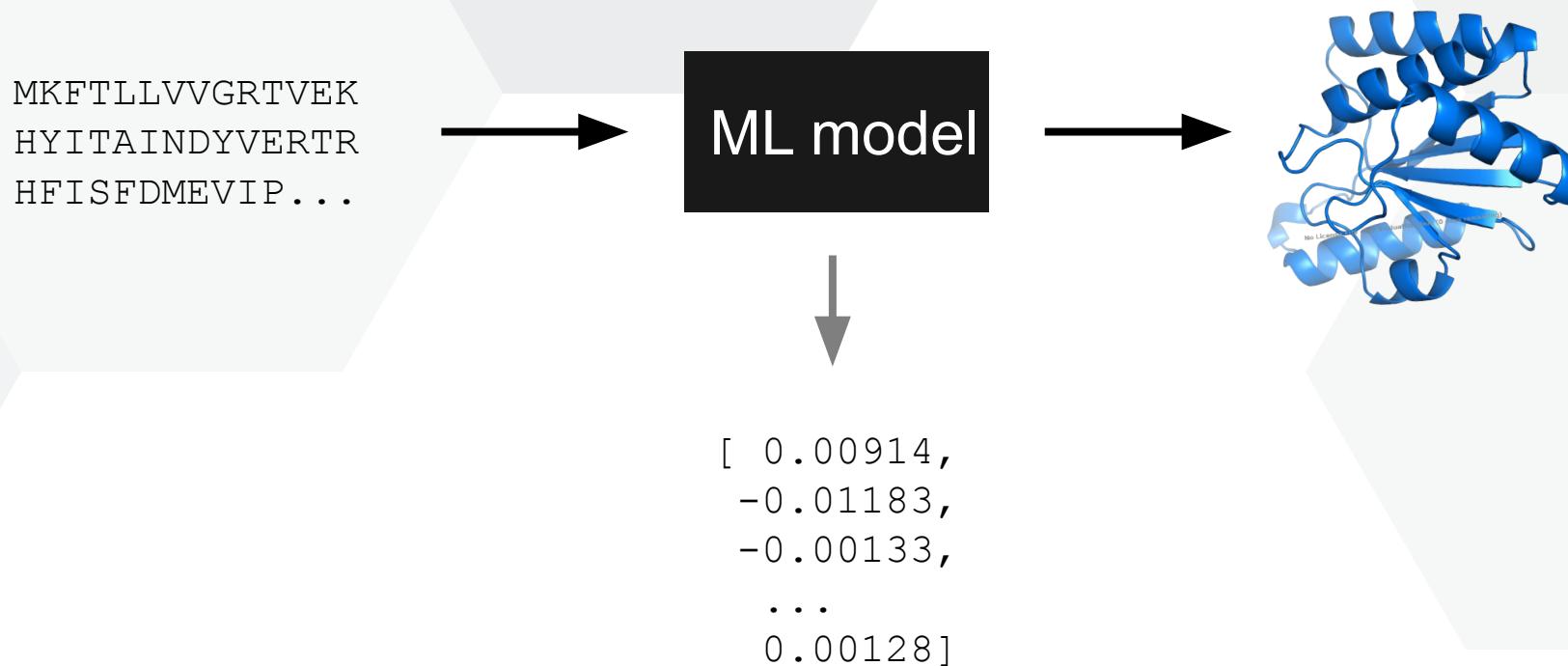
Do we have to work with the 3D structures?

- since protein is determined by its amino acid sequence => it should be enough to work directly with the sequence



Do we have to work with the 3D structures?

- since protein is determined by its amino acid sequence => it should be enough to work directly with the sequence

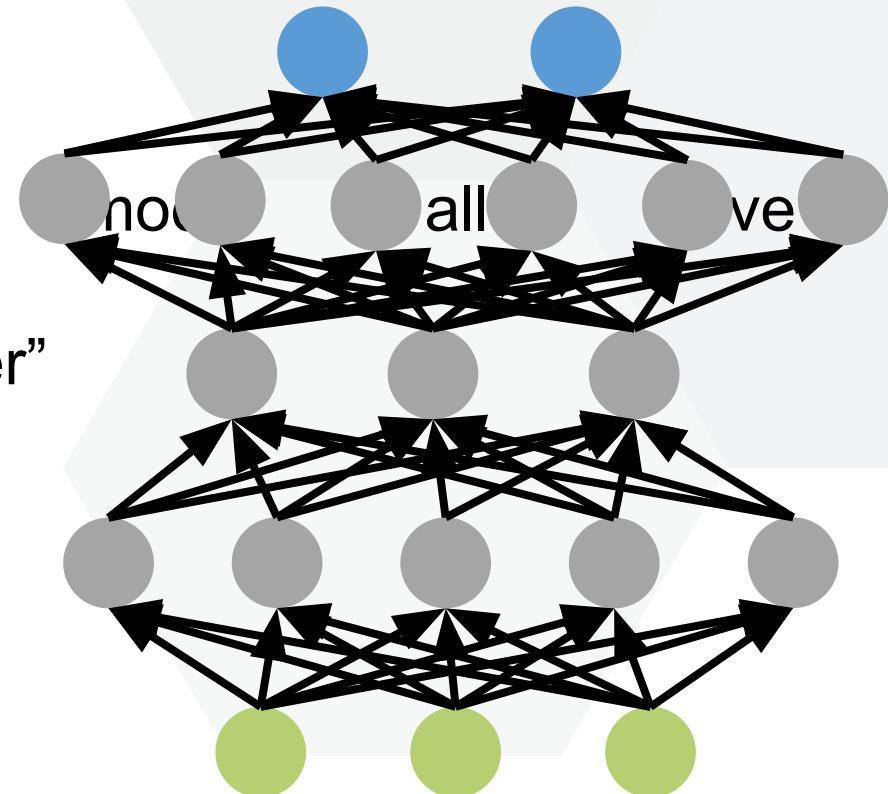


What is an embedding and why you should care

- a relatively low-dimensional numerical representation of some input expressed as a vector

Advantages:

- embeddings from some trained the same size (unlike the original inputs)
- embeddings of similar inputs will be “close together”

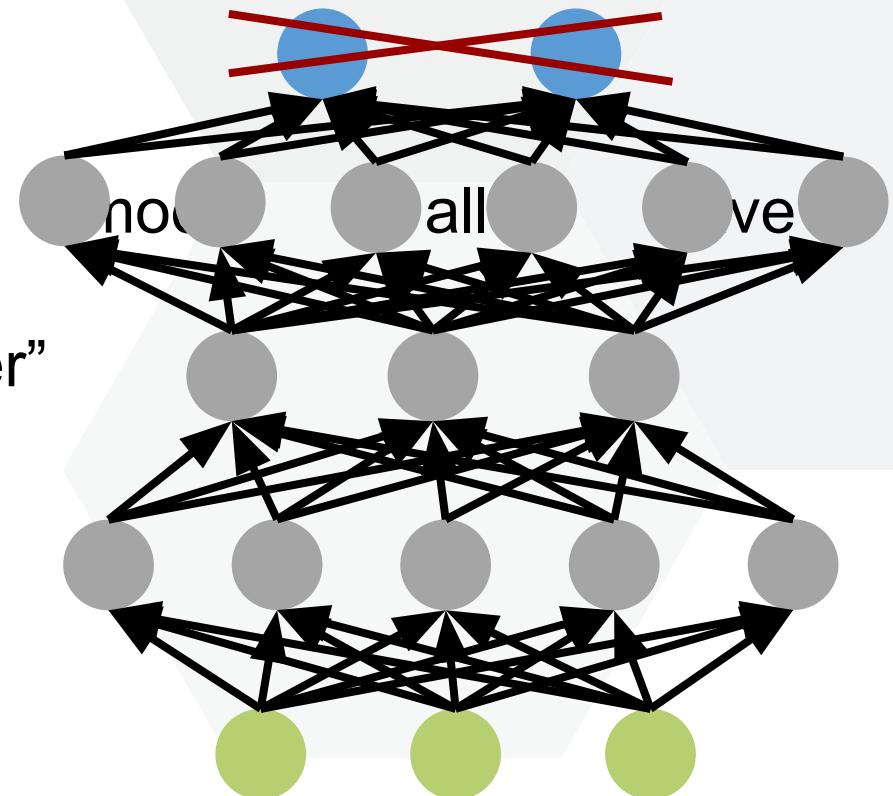


What is an embedding and why you should care

- a relatively low-dimensional numerical representation of some input expressed as a vector

Advantages:

- embeddings from some trained the same size (unlike the original inputs)
- embeddings of similar inputs will be “close together”

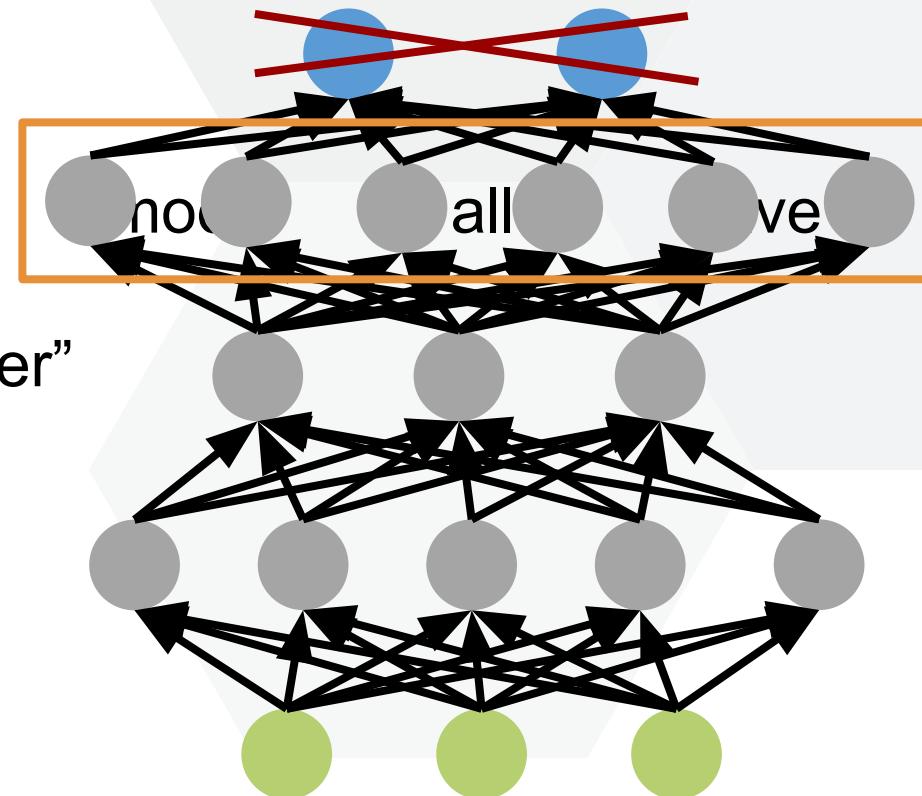


What is an embedding and why you should care

- a relatively low-dimensional numerical representation of some input expressed as a vector

Advantages:

- embeddings from some trained the same size (unlike the original inputs)
- embeddings of similar inputs will be “close together”

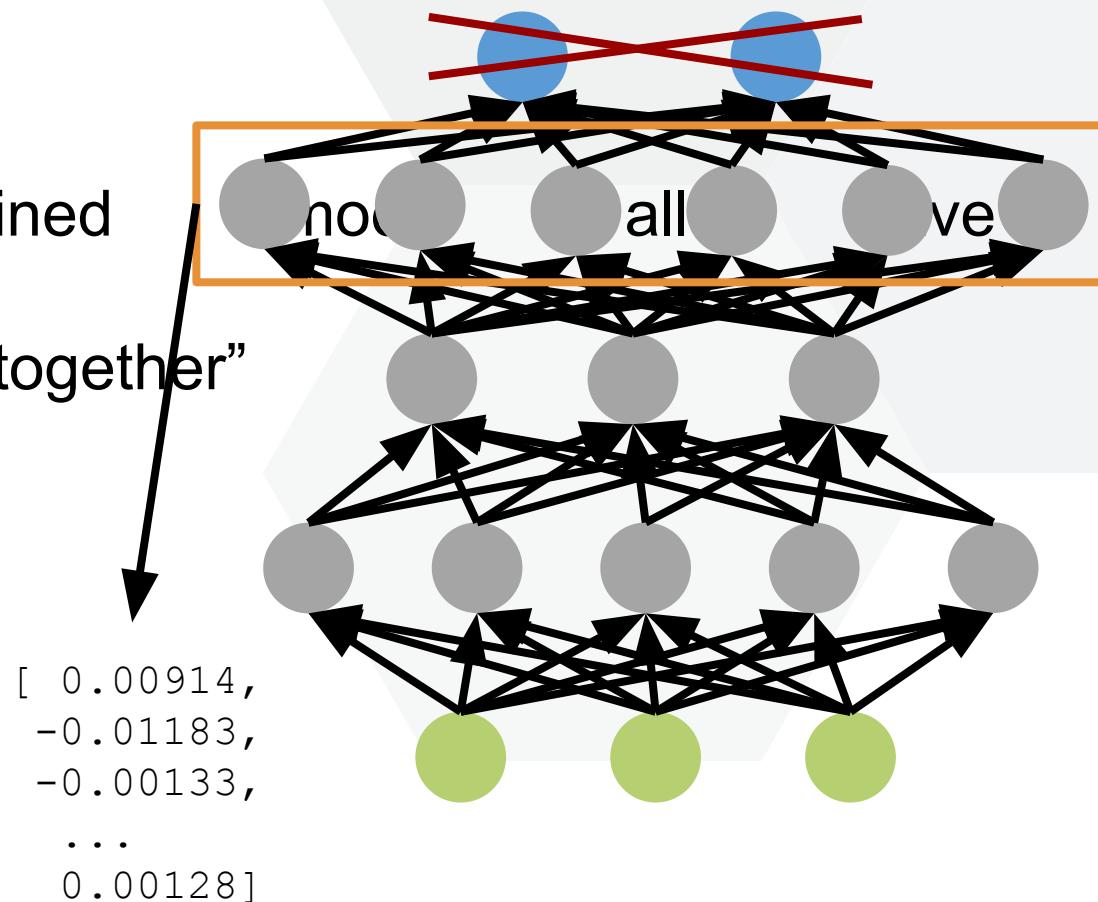


What is an embedding and why you should care

- a relatively low-dimensional numerical representation of some input expressed as a vector

Advantages:

- embeddings from some trained
- the same size (unlike the original inputs)
- embeddings of similar inputs will be “close together”



What is an embedding and why you should care

- a relatively low-dimensional numerical representation of some input expressed as a vector

Advantages:

- embeddings from some trained the same size (unlike the original inputs)
- embeddings of similar inputs will be “close together”

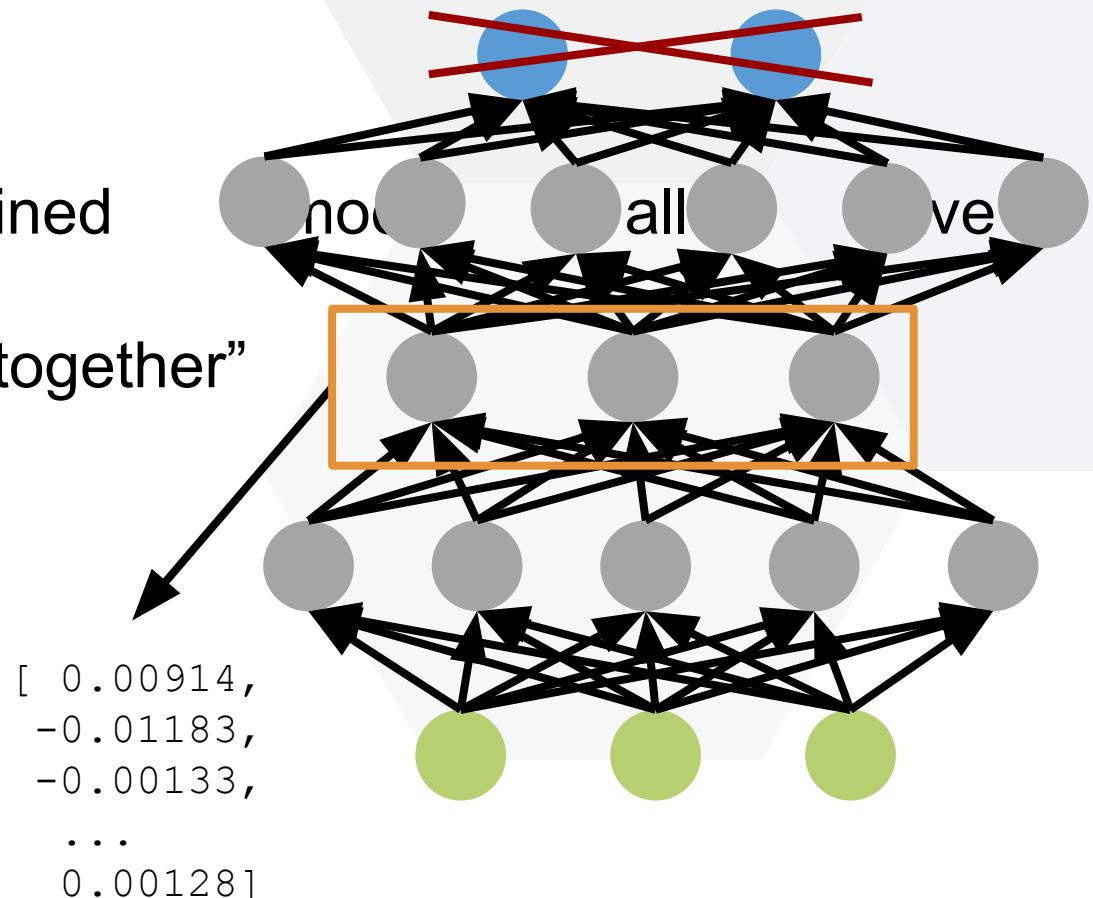
The Nucleotide Transformer: Building and Evaluating Robust Foundation Models for Human Genomics

Hugo Dalla-Torre¹, Liam Gonzalez¹, Javier Mendoza Revilla¹, Nicolas Lopez Carranza¹, Adam Henryk Grzywaczewski², Francesco Oteri¹, Christian Dallago^{2 3}, Evan Trop¹, Hassan Sirelkhatim², Guillaume Richard¹, Marcin Skwark¹, Karim Beguir¹, Marie Lopez^{*† 1}, Thomas Pierrot^{*† 1}

¹InstaDeep ²Nvidia ³TUM

Abstract

Closing the gap between measurable genetic information and observable traits is a longstanding challenge in genomics. Yet, the prediction of molecular phenotypes from DNA sequences alone remains limited and inaccurate, often driven by the scarcity of annotated data and the inability to transfer learnings between prediction tasks. Here, we present an extensive study of foundation models pre-trained on DNA sequences, named the Nucleotide Transformer, integrating information from 3,202 diverse human genomes, as well as 850 genomes from a wide range of species, including



What is an embedding

- a relatively low-dimensional numerical representation of some input expressed as a vector

Advantages:

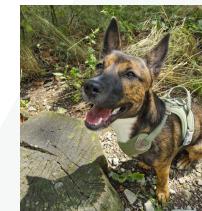
- embeddings from some trained model all have the same size (unlike the original inputs)
- embeddings of similar inputs will be “close together”



[0.00914,
-0.01183,
-0.00133,
...
0.00128]



[0.00926,
-0.01097,
-0.00301,
...
0.00129]



[0.04111,
-0.00873,
-0.0986,
...
0.42666]

What is an embedding

- a relatively low-dimensional numerical representation of some input expressed as a vector

Advantages:

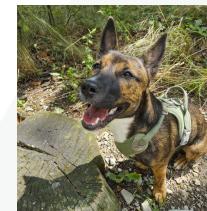
- embeddings from some trained model all have the same size (unlike the original inputs)
- embeddings of similar inputs will be “close together”



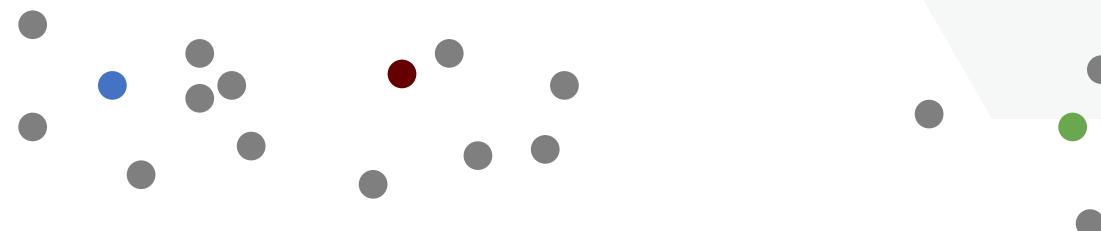
[0.00914,
-0.01183,
-0.00133,
...
0.00128]



[0.00926,
-0.01097,
-0.00301,
...
0.00129]



[0.04111,
-0.00873,
-0.0986,
...
0.42666]



(projection of the embeddings to 2D)

What is an embedding

- a relatively low-dimensional numerical representation of some input expressed as a vector

Advantages:

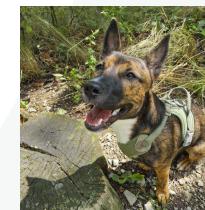
- embeddings from some trained model all have the same size (unlike the original inputs)
- embeddings of similar inputs will be “close together”



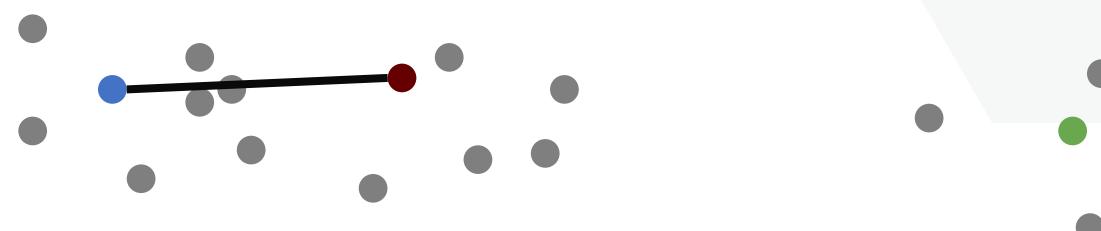
[0.00914,
-0.01183,
-0.00133,
...
0.00128]



[0.00926,
-0.01097,
-0.00301,
...
0.00129]



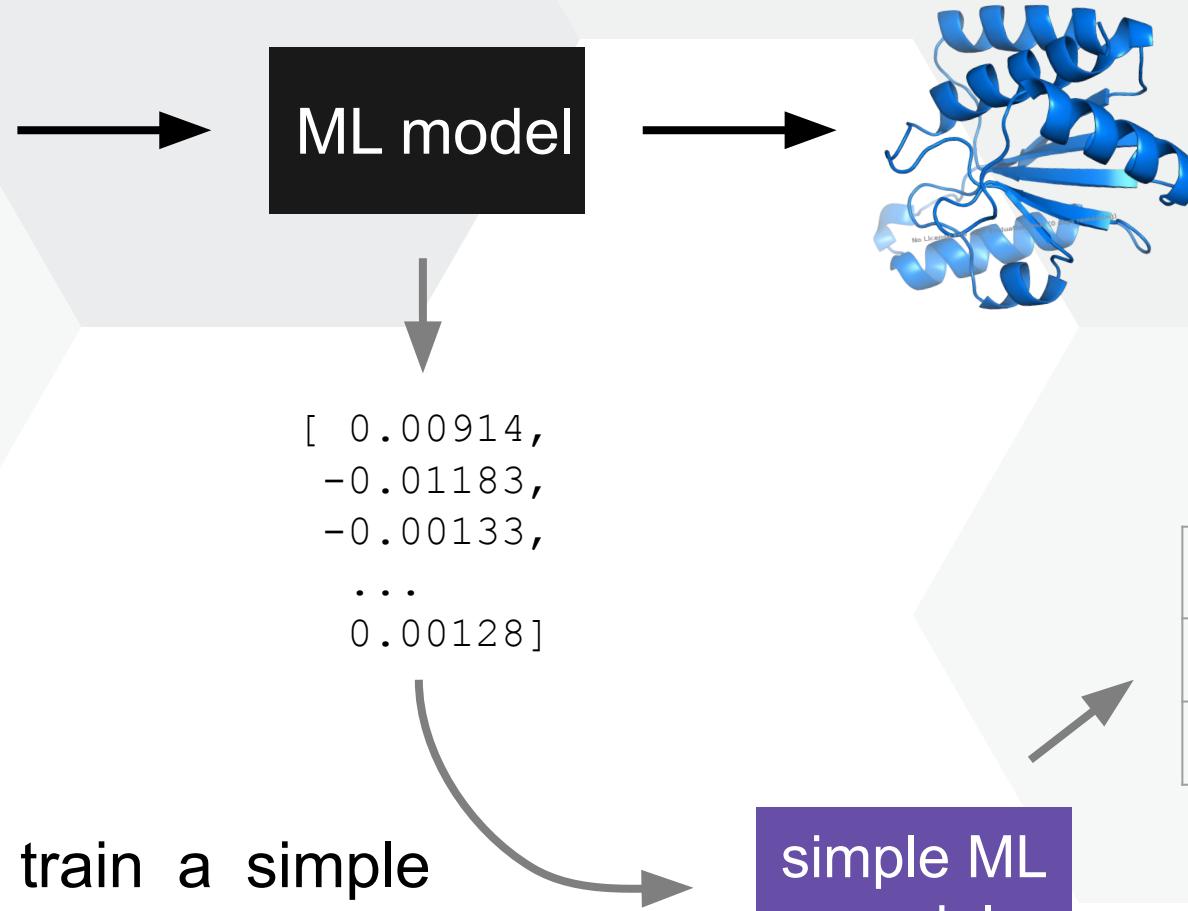
[0.04111,
0.00873,
-0.0986,
...
-0.42666]



Euclidean distance: $d(p,q) = \sqrt{(p-q)^2}$

Extracting embeddings

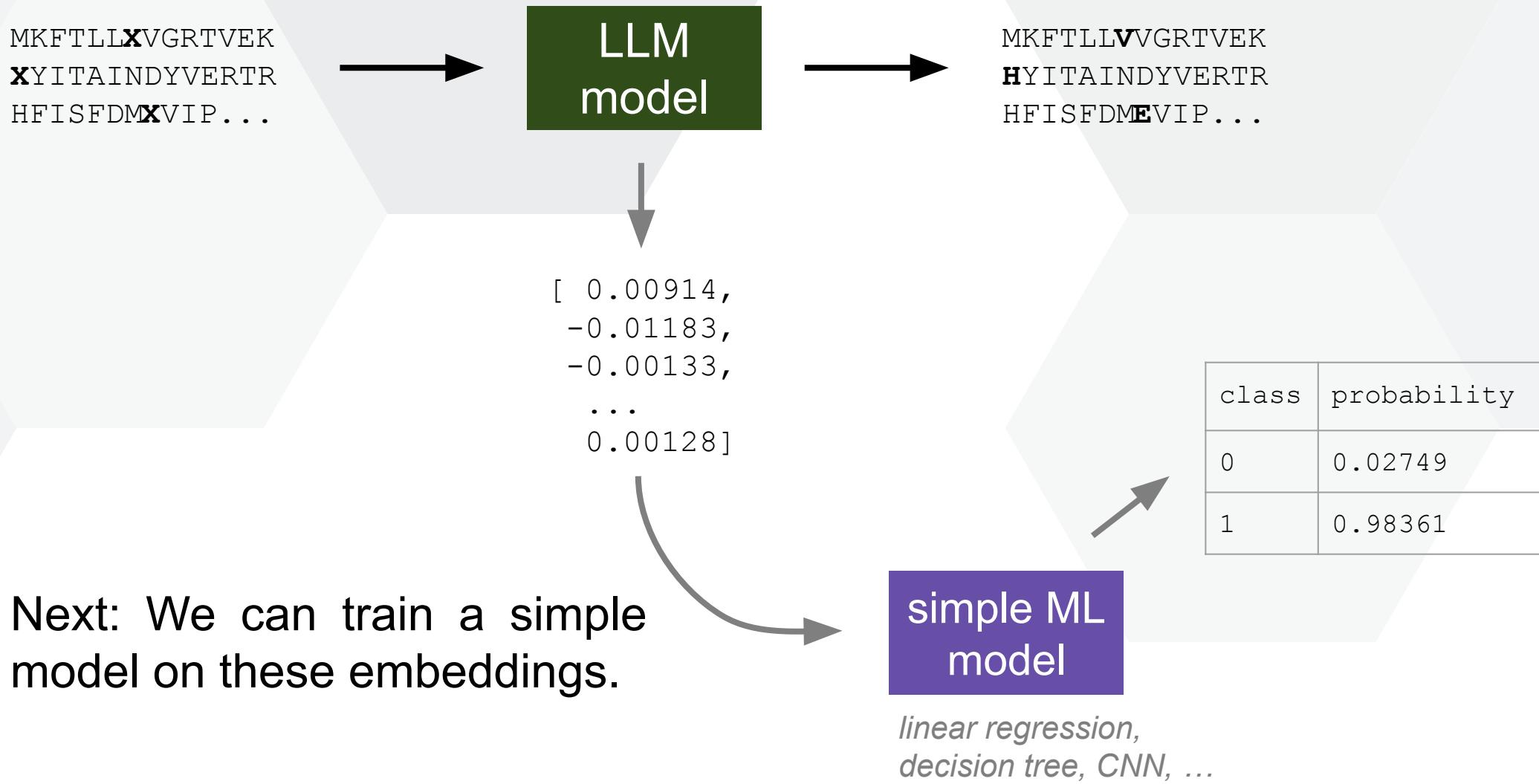
MKF~~T~~LLVVGR~~T~~VEK
HYITAI~~N~~DYVERTR
HFISFDMEVIP...



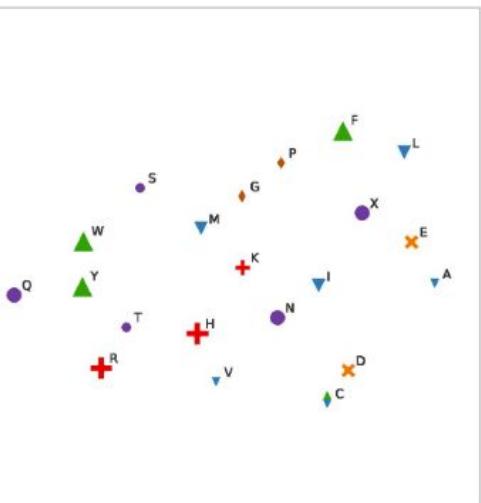
class	probability
0	0.02749
1	0.98361

Next: We can train a simple model on these embeddings.

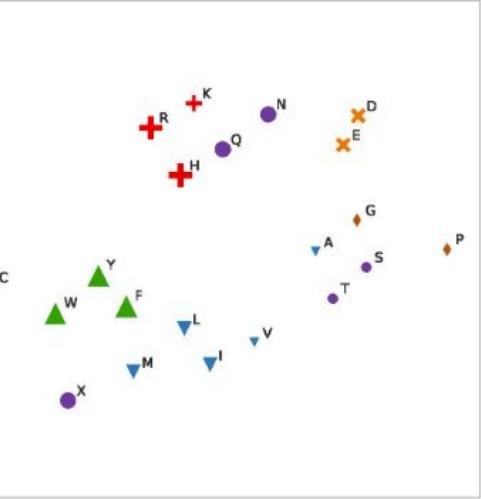
Extracting embeddings



Random

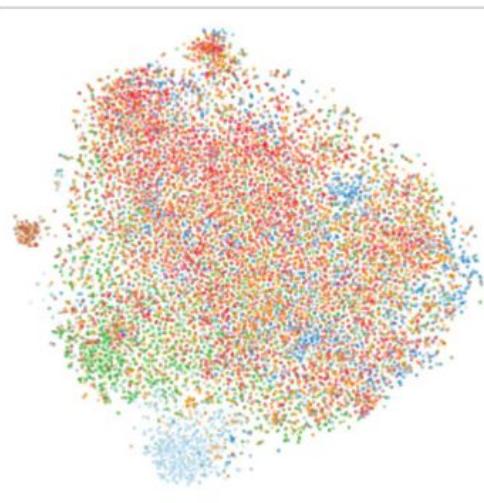


Pre-trained

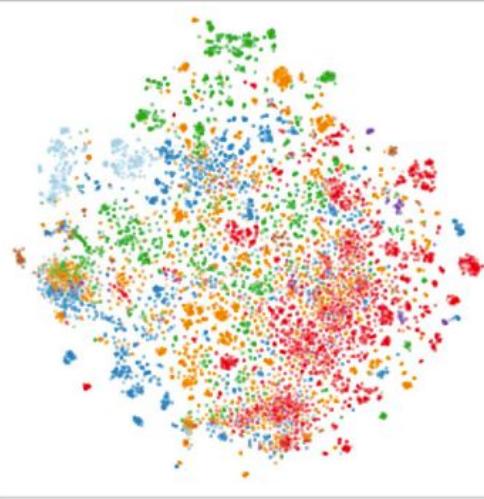


- ▲ Hydrophobic (aromatic)
- ▼ Hydrophobic (aliphatic)
- + Positive
- Negative
- Polar neutral
- ◆ Special cases
- Small (<130 Dalton)
- Medium
- Big (>150 Dalton)

A Amino acids

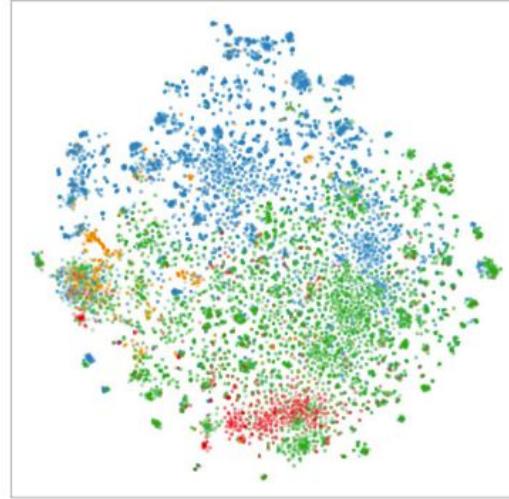
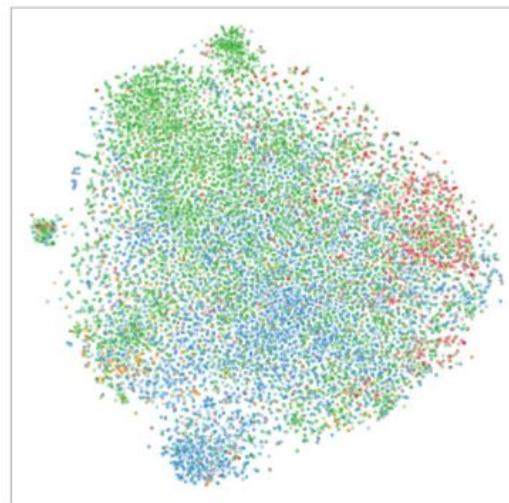


B Structure: SCOPe



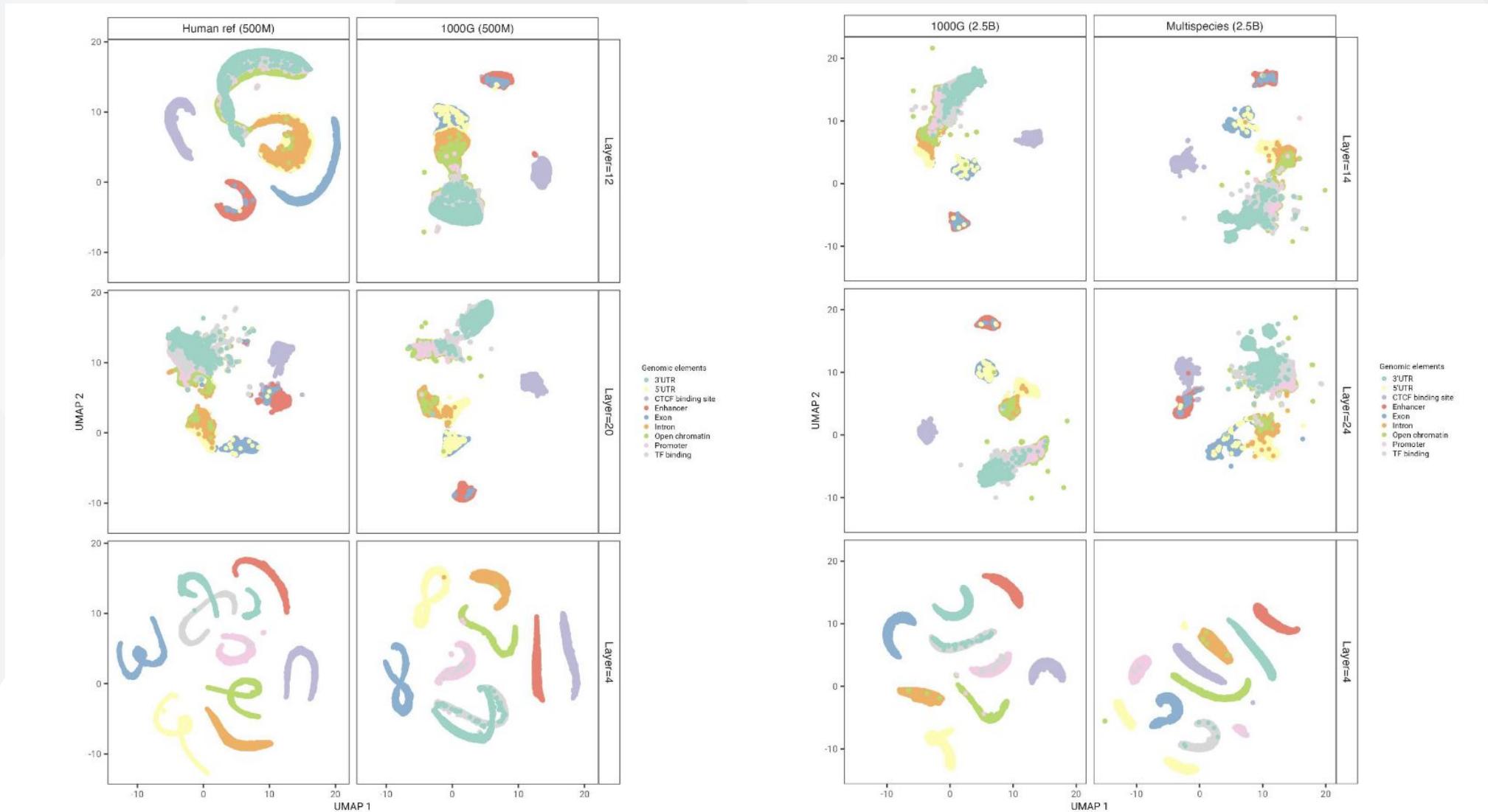
- All alpha
- All beta
- Alpha & beta (a|b)
- Alpha & beta (a+b)
- Small (<130 Dalton)
- Medium
- Big (>150 Dalton)
- Multi-domain
- Membrane, cell surface
- Small proteins

C Lineage: Kingdoms



<https://arxiv.org/ftp/arxiv/papers/2007/2007.06225.pdf>

Do the embeddings encode representations of the different genomic sequence features? [6]



Supplementary Figure 14: U-MAP projections of embeddings of nine regulatory elements from layers 4, 14 and 24 for the two Nucleotide Transformer 500M parameters models.

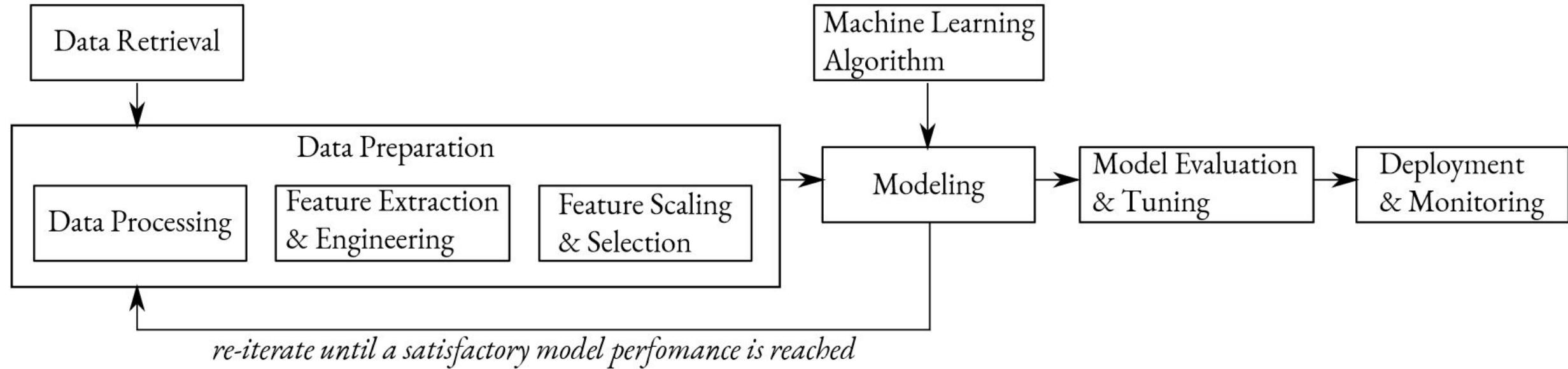
Supplementary Figure 13: U-MAP projections of embeddings of nine regulatory elements from layers 4, 14 and 24 for the two Nucleotide Transformer 2.5B parameters models.

-> confirms layer specialization

30 minutes coffee break



Machine Learning pipeline



How can we preprocess sequence data?
What are the properties of ideal dataset?

Tokenizer

- divides text into smaller units (“tokens”)

1. Tokenization in NLP:

Tokenizer

- divides text into smaller units (“tokens”)
1. Tokenization in NLP:
 - word-level: “This sentence means nothing.” -> “This”, “sentence”, “means”, “nothing”

Tokenizer

- divides text into smaller units (“tokens”)
1. Tokenization in NLP:
 - word-level: “This sentence means nothing.” -> “This”, “sentence”, “means”, “nothing”
 - character-level: -> “T”, “h”, “i”, “s”, “s”, “e”, “n”, “t”, “e”, “n”, “c”, “e”, “m”, “e”, ...

Tokenizer

- divides text into smaller units (“tokens”)
1. Tokenization in NLP:
 - word-level: “This sentence means nothing.” -> “This”, “sentence”, “means”, “nothing”
 - character-level: -> “T”, “h”, “i”, “s”, “s”, “e”, “n”, “t”, “e”, “n”, “c”, “e”, “m”, “e”, ...
 - subword-level: -> “This”, “sentence”, “mean”, “s”, “nothing”

Tokenizer

- divides text into smaller units (“tokens”)
1. Tokenization in NLP:
 - word-level: “This sentence means nothing.” -> “This”, “sentence”, “means”, “nothing”
 - character-level: -> “T”, “h”, “i”, “s”, “s”, “e”, “n”, “t”, “e”, “n”, “c”, “e”, “m”, “e”, ...
 - subword-level: -> “This”, “sentence”, “mean”, “s”, “nothing”
 2. Mapping tokens to integers:
 - dictionary (unique value for each token)

Tokenizer

- divides text into smaller units (“tokens”)
1. Tokenization in NLP:
 - word-level: “This sentence means nothing.” -> “This”, “sentence”, “means”, “nothing”
 - character-level: -> “T”, “h”, “i”, “s”, “s”, “e”, “n”, “t”, “e”, “n”, “c”, “e”, “m”, “e”, ...
 - subword-level: -> “This”, “sentence”, “mean”, “s”, “nothing”
 2. Mapping tokens to integers:
 - dictionary (unique value for each token)
 3. Dealing with different input lengths:
 - padding: adding special [PAD] token to the end of short sentences
 - truncation: cutting off ends of too long sentences

Tokenizer

- divides text into smaller units (“tokens”)

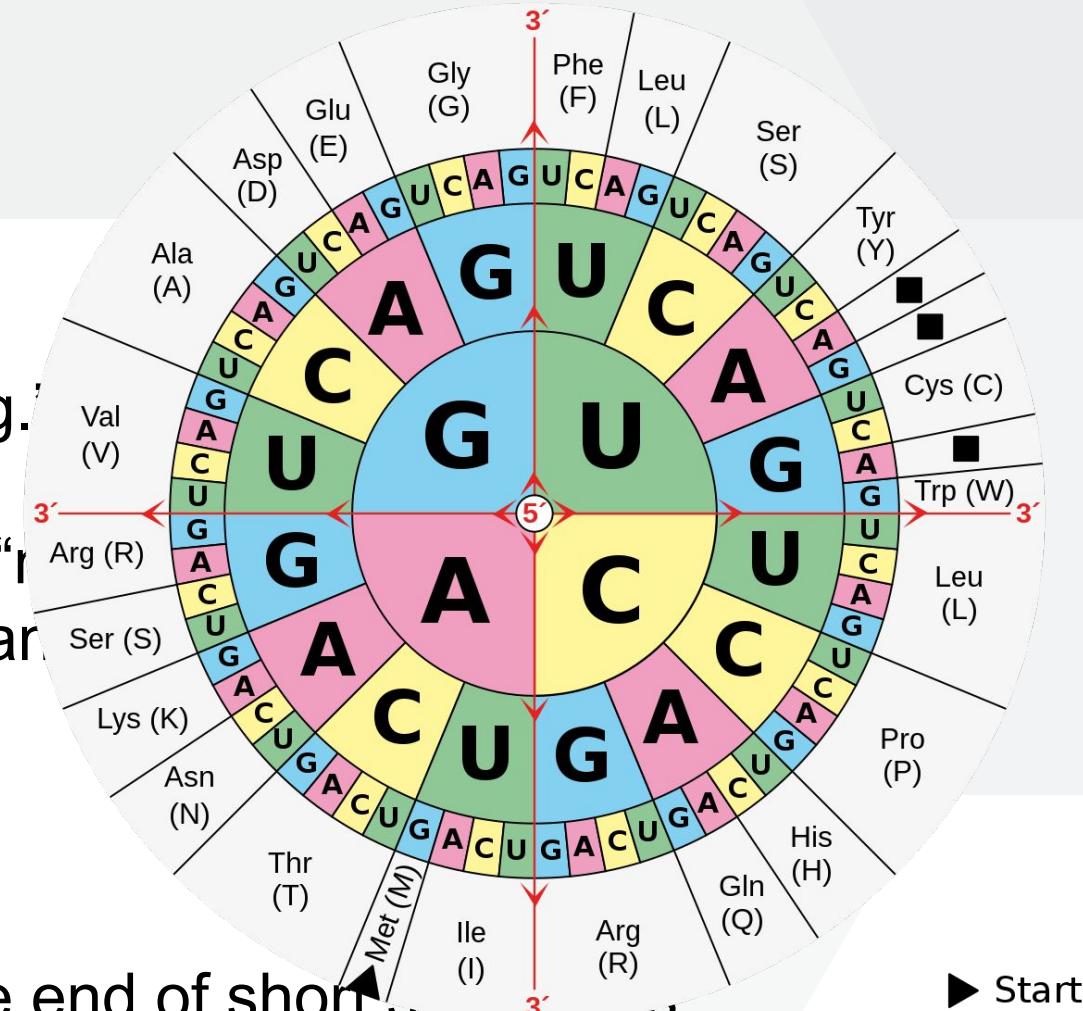
1. Tokenization in NLP:

2. Mapping tokens to integers:

- dictionary (unique value for each token)

3. Dealing with different input lengths:

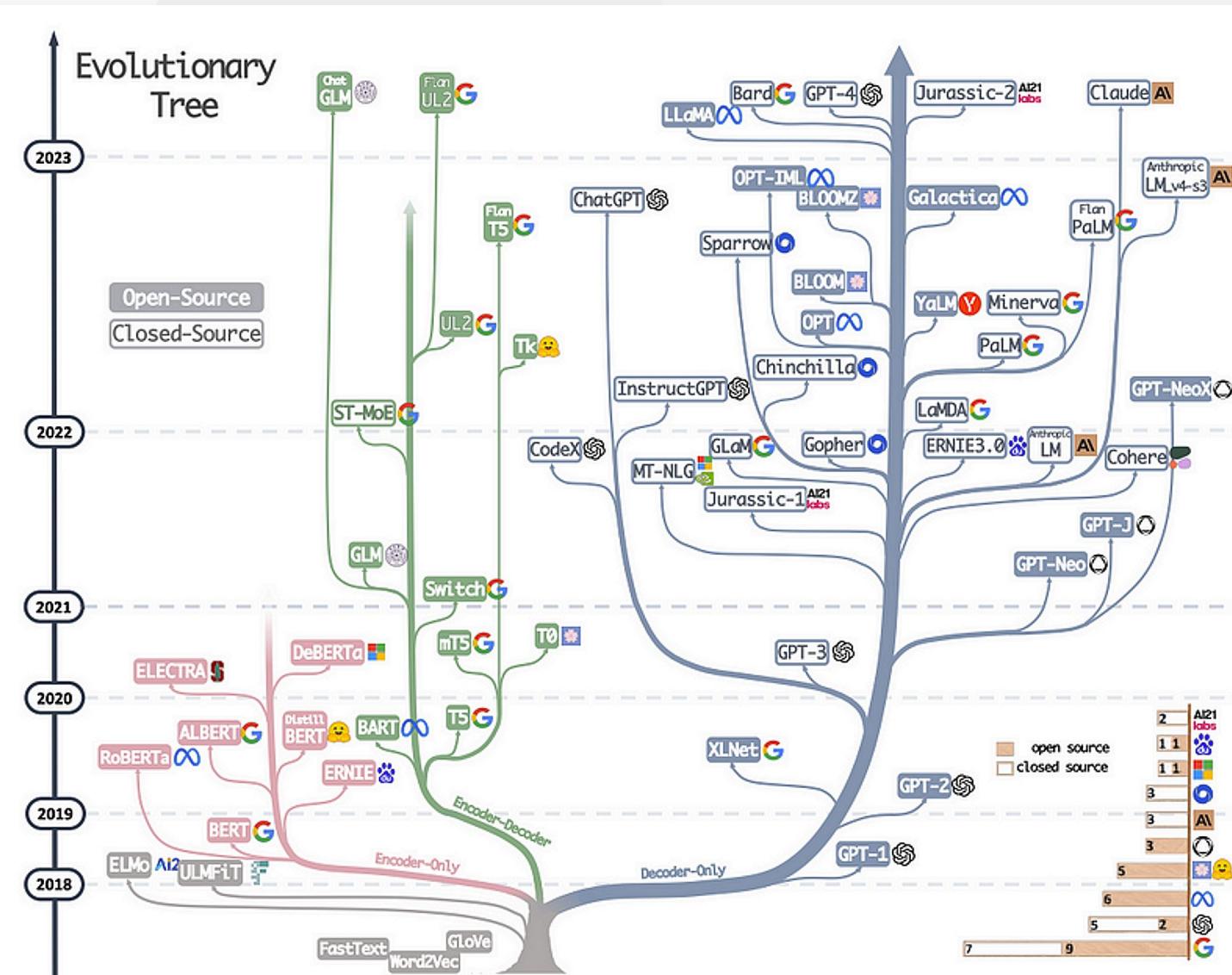
- padding: adding special [PAD] token to the end of short sentences
 - truncation: cutting off ends of too long sentences



 Start
 Stop

Large Language Models (LLMs)

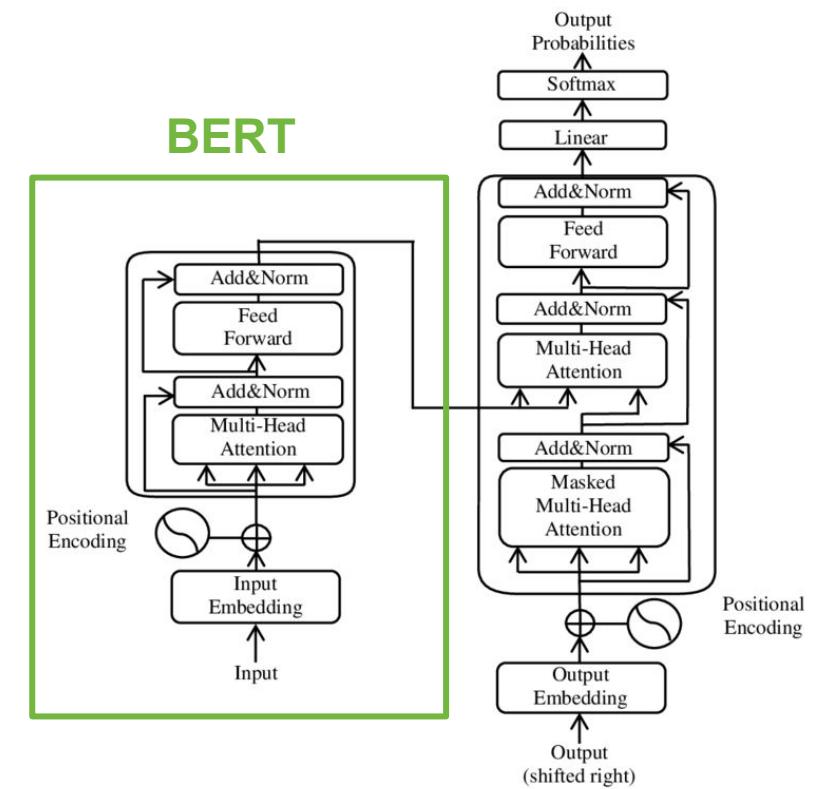
Large Language Models



Transformer architecture [7]

- originally used in NLP (goal: understand not only single words, but also the context of those words)
- NLP tasks: translation, sentence classification, word classification, text generation, text summarization, extracting answers from text, ...
- Encoder-Decoder architecture:
 - Encoder: converts input to some numerical representation
 - Decoder: optimized to generating outputs (*originally designed for translation*)

The most important mechanism: Attention == how important is each word in the input sequence.



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

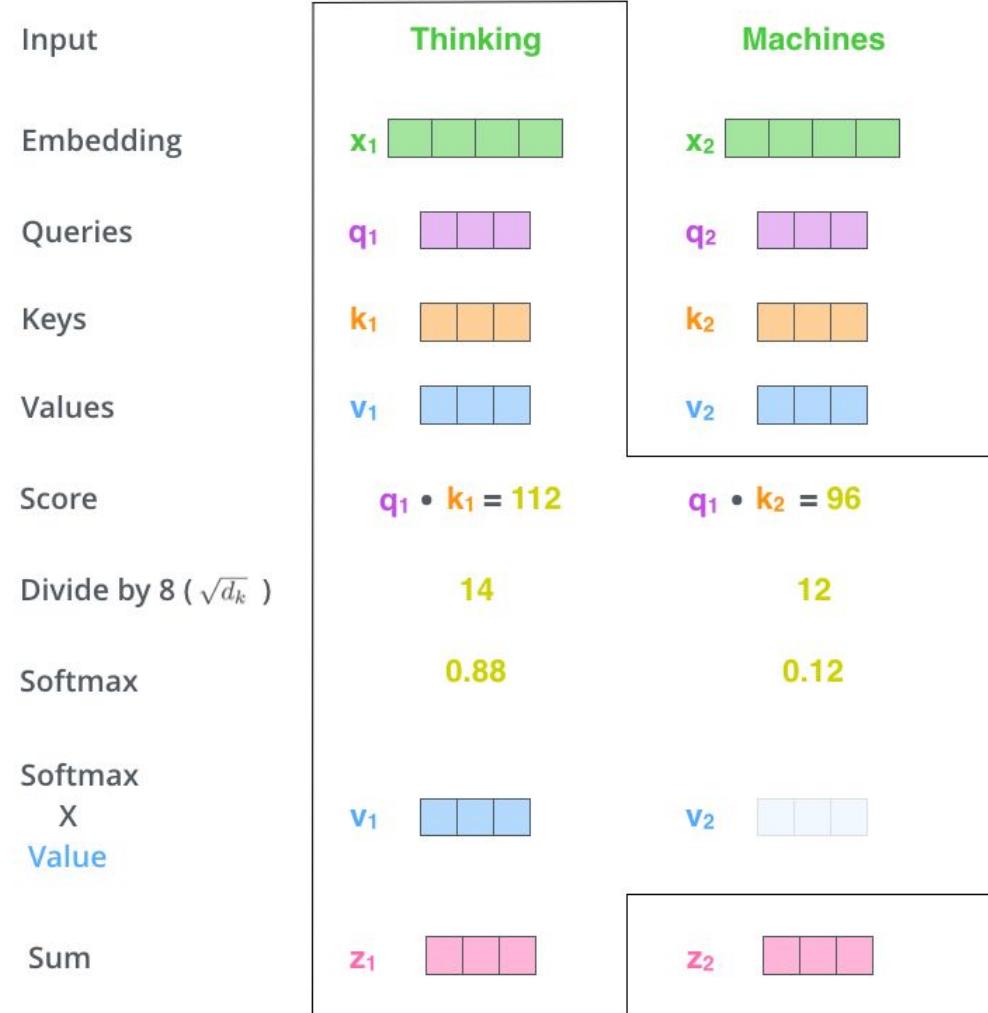
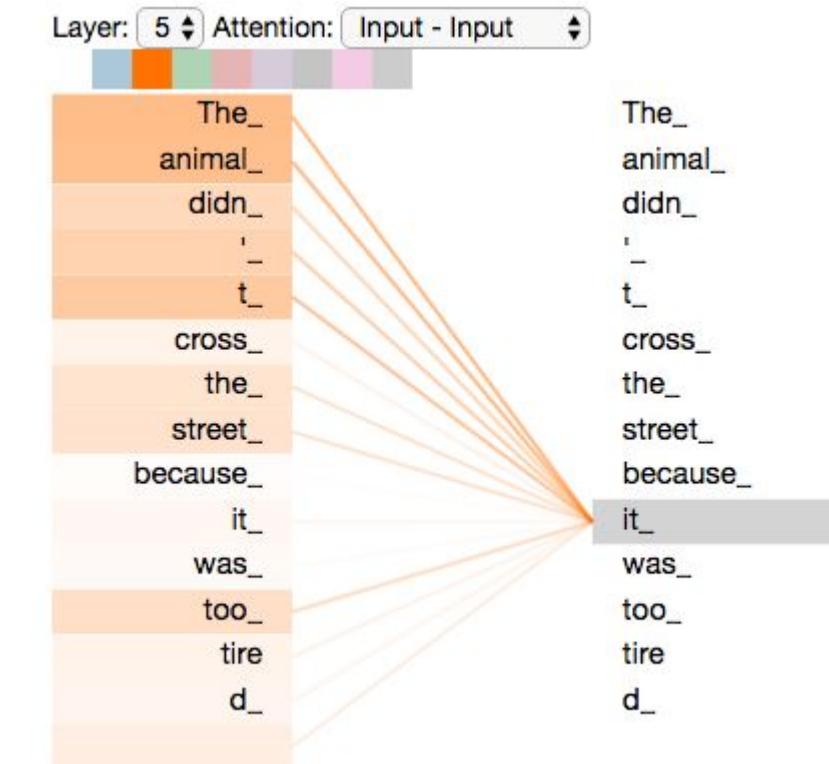
Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

Lukasz Kaiser*
Google Brain
lukaszkaiser@google.com

Illia Polosukhin* ‡
illia.polosukhin@gmail.com

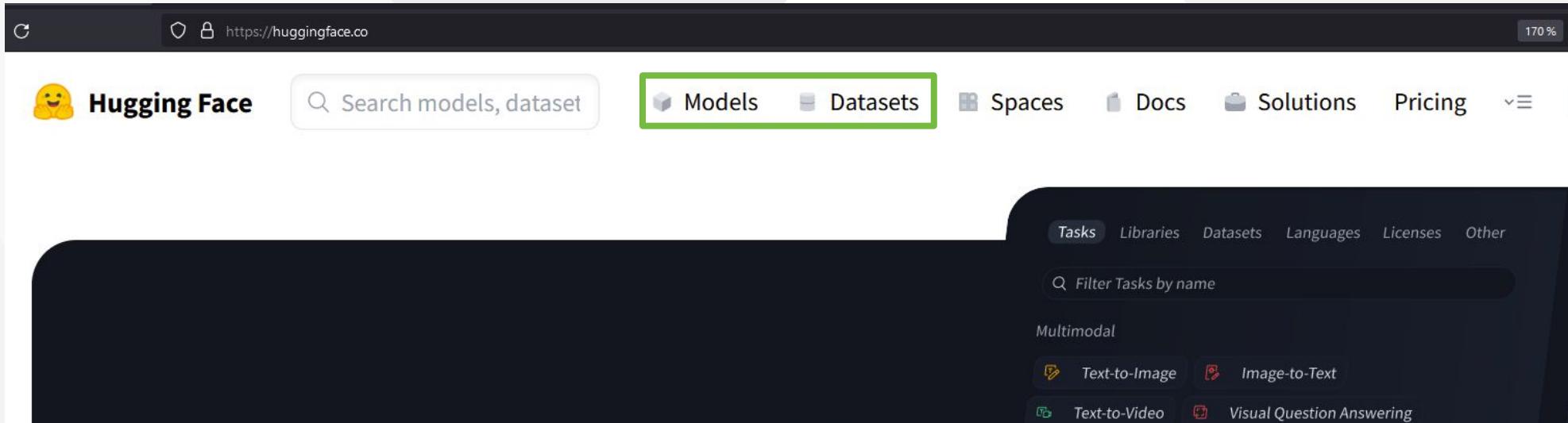
Transformers: Attention (<http://jalammar.github.io/illustrated-transformer/>)

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



HuggingFace 😊 (<https://huggingface.co/>)

- data science platform that provides tools to build, train and deploy ML models
- repository of models and datasets



HuggingFace 😊 (<https://huggingface.co/>)

- data science platform that provides tools to build, train and deploy ML models
- repository of **models** and datasets

The screenshot shows the HuggingFace website interface. At the top, there's a navigation bar with a logo, a search bar, and links for Models, Datasets, Spaces, Docs, Solutions, Pricing, Log In, and Sign Up. Below the navigation is a sidebar with sections for Tasks, Libraries, Datasets, Languages, Licenses, Other, and various sub-sections like Multimodal, Computer Vision, and Natural Language Processing, each with a list of tasks. The main content area is titled "Models 324,824" and features a list of trending models with their names, authors, tasks, last update, and metrics like 11.7k and 394.

Model	Author	Task	Last Update	Metric
tiiuae/falcon-180B	tiiuae	Text Generation	Updated 3 days ago	11.7k
llyasviel/sd_control_collection	llyasviel	Text-to-Image	Updated about 7 hours ago	214
tiiuae/falcon-180B-chat	tiiuae	Text Generation	Updated 3 days ago	3.59k
stabilityai/stable-diffusion-xl-base-1.0	stabilityai	Text-to-Image	Updated 5 days ago	1M
WizardLM/WizardCoder-Python-34B-V1.0	WizardLM	Text Generation	Updated about 4 hours ago	29.8k
meta-llama/Llama-2-7b	meta-llama	Text Generation	Updated Jul 19	2.37k
Phind/Phind-CodeLlama-34B-v2	Phind	Text Generation	Updated 12 days ago	5.42k

HuggingFace 😊 (<https://huggingface.co/>)

- data science platform that provides tools to build, train and deploy ML models
- repository of **models** and datasets

The screenshot shows the HuggingFace website interface. On the left, a sidebar lists various categories: Tasks (Libraries, Datasets, Languages, Licenses, Other), Multimodal (Feature Extraction, Text-to-Image, Image-to-Text, Text-to-Video, Visual Question Answering, Document Question Answering, Graph Machine Learning), Computer Vision (Depth Estimation, Image Classification, Object Detection, Image Segmentation, Image-to-Image, Unconditional Image Generation, Video Classification, Zero-Shot Image Classification), and Natural Language Processing (Text Classification, Token Classification, Table Question Answering, Question Answering, Zero-Shot Classification, Translation, Summarization, Conversational, Text Generation). The main content area displays a list of 27 models under the 'Models' tab. Each model entry includes the owner's name, model name, type, last update, size, and a star count. A search bar at the top allows users to search for models and datasets.

Model	Type	Last Update	Size	Stars
qilowq/AbLang_heavy	Fill-Mask	Updated May 1	~6k	1
Rostlab/prot_bert	Fill-Mask	Updated Dec 11, 2020	~30.2k	50
Rostlab/prot_t5_xl_bfd	Text2Text Generation	Updated Dec 11, 2020	~2.23k	10
rampasek/prot_bert_bfd_rosetta20aa	Text Classification	Updated Mar 29, 2022		
lightonai/RITA_s	Text Generation	Updated May 19, 2022	~1.02k	2
lightonai/RITA_l	Text Generation	Updated May 19, 2022	~5	
jonathang/Protein_Family_Models		Updated Feb 17		
evankomp/learn2therm	Text Classification	Updated Jun 27		
zjunlp/llama-molinst-protein-7b	Text Generation	Updated Jul 27	~9	
AmelieSchreiber/esm2_t6_8M_UR500_rna_b1	Token Classification	Updated Aug 6		

HuggingFace 😊 (<https://huggingface.co/>)

- data science platform that provides tools to build, train and deploy ML models
- repository of Models and **Datasets**

The image shows two screenshots of the HuggingFace website. The left screenshot displays the 'Tasks' section, which is a navigation menu for building, training, and deploying ML models. It includes categories like Multimodal (Feature Extraction, Text-to-Image, Image-to-Text, Text-to-Video, Visual Question Answering, Graph Machine Learning), Computer Vision (Depth Estimation, Image Classification, Object Detection, Image Segmentation, Image-to-Image, Unconditional Image Generation, Video Classification, Zero-Shot Image Classification), and Natural Language Processing (Text Classification, Token Classification, Table Question Answering, Question Answering, Zero-Shot Classification, Translation). The right screenshot shows the 'Datasets' section, which lists 59,663 datasets. The datasets are presented in a grid format, each with a thumbnail, name, owner, last updated date, size, and popularity metrics (3.15k, 3.29k for the first dataset).

Dataset	Owner	Last Updated	Size	Popularity
fka/awesome-chatgpt-prompts	Viewer	Mar 7	3.15k	3.29k
dell-research-harvard/AmericanStories		About 16 hours ago	834	57
tiiuae/falcon-refinedweb		Jun 20	1.3k	498
garage-bAInd/Open-Platypus		25 days ago	10.9k	190
liwu/MNBVC		14 days ago	1.04k	225
b-mc2/sql-create-context		Apr 21	2.32k	123

Reproducibility (<https://paperswithcode.com/>)

The screenshot shows a web browser displaying the [ProteinBERT](https://paperswithcode.com/paper/proteinbert-a-universal-deep-learning-model) page on the [PapersWithCode](https://paperswithcode.com) platform. The page title is "ProteinBERT: a universal deep-learning model of protein sequence and function". It is categorized under "Bioinformatics, Volume 38, Issue 8 2022" and authored by Nadav Brandes, Dan Ofer, Yam Peleg, Nadav Rappoport, Michal Linial. The abstract discusses the development of ProteinBERT, a self-supervised deep language model designed for proteins, which combines language modeling with Gene Ontology annotation prediction. The architecture includes local and global representations for processing long sequences. ProteinBERT outperforms existing methods on various benchmarks, including protein structure, post-translational modifications, and biophysical attributes, using a smaller and faster model. The page also features links to PDF and abstract download options, and sections for code and tasks.

ProteinBERT: a universal deep-learning model of protein sequence and function

Bioinformatics, Volume 38, Issue 8 2022 · Nadav Brandes, Dan Ofer, Yam Peleg, Nadav Rappoport, Michal Linial · [Edit social preview](#)

Self-supervised deep language modeling has shown unprecedented success across natural language tasks, and has recently been repurposed to biological sequences. However, existing models and pretraining methods are designed and optimized for text analysis. We introduce ProteinBERT, a deep language model specifically designed for proteins. Our pretraining scheme combines language modeling with a novel task of Gene Ontology (GO) annotation prediction. We introduce novel architectural elements that make the model highly efficient and flexible to long sequences. The architecture of ProteinBERT consists of both local and global representations, allowing end-to-end processing of these types of inputs and outputs. ProteinBERT obtains near state-of-the-art performance, and sometimes exceeds it, on multiple benchmarks covering diverse protein properties (including protein structure, post-translational modifications and biophysical attributes), despite using a far smaller and faster model than competing deep-learning methods. Overall, ProteinBERT provides an efficient framework for rapidly training protein predictors, even with limited labeled data.

[PDF](#) [Abstract](#)

Code

Owner	Name	Stars	Language
	nadavbra/protein_bert	363	TensorFlow
	Aedelon/ProteinBERT-PyTorch-Replica...	3	PyTorch

Tasks

Task Type
Language Modelling
Protein Secondary Structure Prediction
Protein Structure Prediction

HuggingFace 😊 (<https://huggingface.co/>)

Loading a Dataset:

```
from datasets import Dataset, load_dataset  
  
dss = load_dataset('<USER_NAME>/<DATASET_NAME>')  
dss
```

```
DatasetDict({  
    test: Dataset({  
        features: ['identifier', 'aa_sequence', 'length', 'label', 'family_name'],  
        num_rows: 39412  
    })  
    train: Dataset({  
        features: ['identifier', 'aa_sequence', 'length', 'label', 'family_name'],  
        num_rows: 157644  
    })  
})
```

code cell

code cell output

HuggingFace 😊 (<https://huggingface.co>)

Pipelines

- objects that abstract complex code from the library

Parameters

- **task** (str) — The task defining which pipeline will be returned. Currently accepted tasks are:
 - "audio-classification": will return a [AudioClassificationPipeline](#).
 - "automatic-speech-recognition": will return a [AutomaticSpeechRecognitionPipeline](#).
 - "conversational": will return a [ConversationalPipeline](#).
 - "depth-estimation": will return a [DepthEstimationPipeline](#).
 - "document-question-answering": will return a [DocumentQuestionAnsweringPipeline](#).
 - "feature-extraction": will return a [FeatureExtractionPipeline](#).
 - "fill-mask": will return a [FillMaskPipeline](#).
 - "image-classification": will return a [ImageClassificationPipeline](#).
 - "image-segmentation": will return a [ImageSegmentationPipeline](#).
 - "image-to-text": will return a [ImageToTextPipeline](#).
 - "mask-generation": will return a [MaskGenerationPipeline](#).
 - "object-detection": will return a [ObjectDetectionPipeline](#).
 - "question-answering": will return a [QuestionAnsweringPipeline](#).
 - "summarization": will return a [SummarizationPipeline](#).
 - "table-question-answering": will return a [TableQuestionAnsweringPipeline](#).
 - "text2text-generation": will return a [Text2TextGenerationPipeline](#).
 - "text-classification" (alias "sentiment-analysis" available): will return a [TextClassificationPipeline](#).
 - "text-generation": will return a [TextGenerationPipeline](#).
 - "text-to-audio" (alias "text-to-speech" available): will return a [TextToAudioPipeline](#).
 - "token-classification" (alias "ner" available): will return a [TokenClassificationPipeline](#).
 - "translation": will return a [TranslationPipeline](#).
 - "translation_xx_to_yy": will return a [TranslationPipeline](#).
 - "video-classification": will return a [VideoClassificationPipeline](#).
 - "visual-question-answering": will return a [VisualQuestionAnsweringPipeline](#).
 - "zero-shot-classification": will return a [ZeroShotClassificationPipeline](#).
 - "zero-shot-image-classification": will return a [ZeroShotImageClassificationPipeline](#).
 - "zero-shot-audio-classification": will return a [ZeroShotAudioClassificationPipeline](#).
 - "zero-shot-object-detection": will return a [ZeroShotObjectDetectionPipeline](#).
- **model** (str or [PreTrainedModel](#) or [TFPreTrainedModel](#), optional) — The model that will be used by the pipeline to make predictions. This can be a model identifier or an actual instance of a pretrained model inheriting from [PreTrainedModel](#) (for PyTorch) or [TFPreTrainedModel](#) (for TensorFlow).

HuggingFace 😊 (<https://huggingface.co/>)

Using a Model (+ Tokenizer in a Pipeline) for prediction:

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification, TextClassificationPipeline

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, do_lower_case=False)
model = AutoModel.from_pretrained(MODEL_NAME, num_labels=2)
pipe = TextClassificationPipeline(model=model, tokenizer=tokenizer, return_all_scores=True)
```

On 1 sequence:

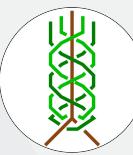
```
pipe('DENCADENCA')

[[{'label': 'LABEL_0', 'score': 0.5323733687400818},
 {'label': 'LABEL_1', 'score': 0.4676266014575958}]]
```

Or on the whole Dataset:

```
def tokenize_function(examples):
    return tokenizer(' '.join(list(examples['seq'])))
tokenized_dss = dss.map(tokenize_function, num_proc=4)
results = pipe(tokenized_dss['test']['seq'])
```

Case study: Protein knots



(<https://topoly.cent.uw.edu.pl/>)

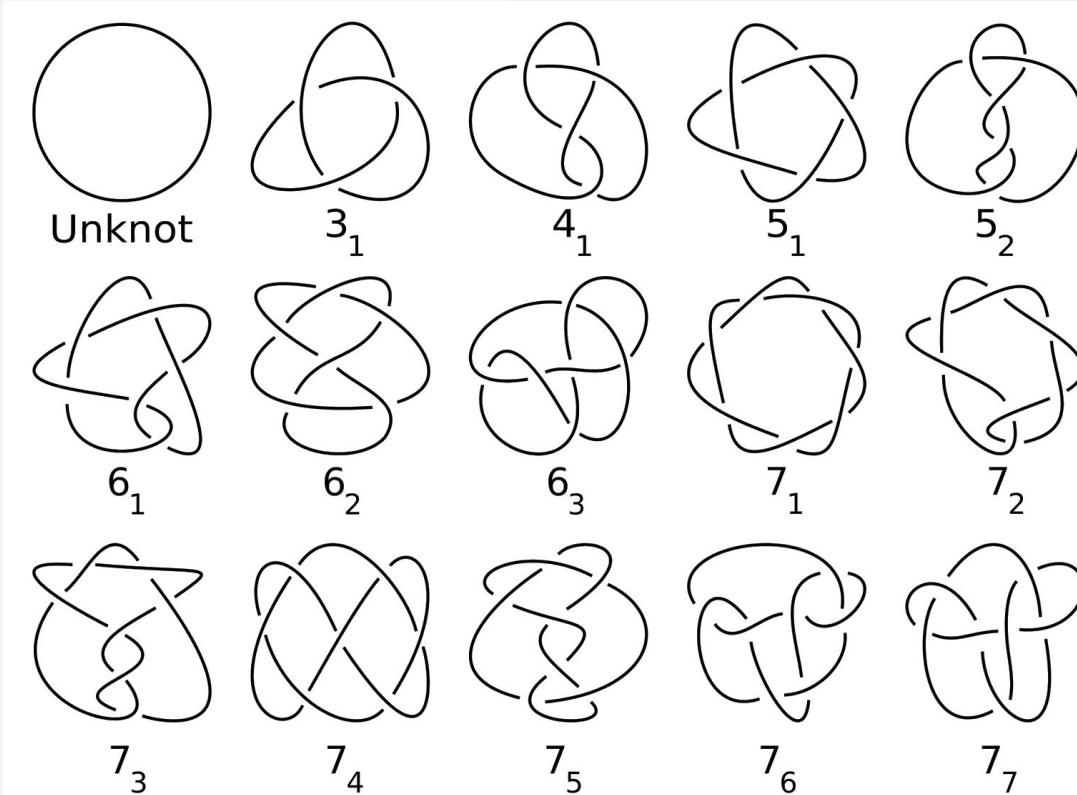
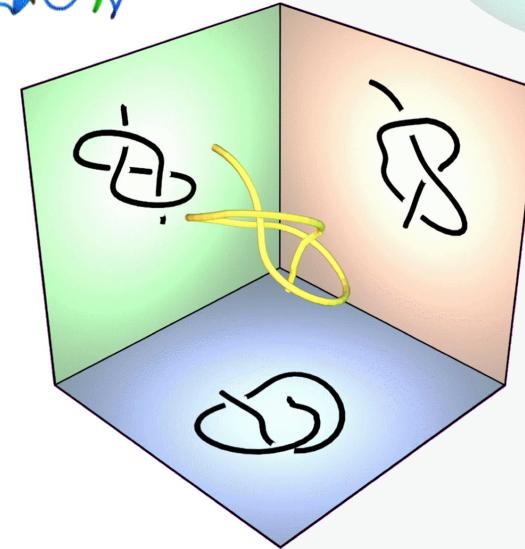
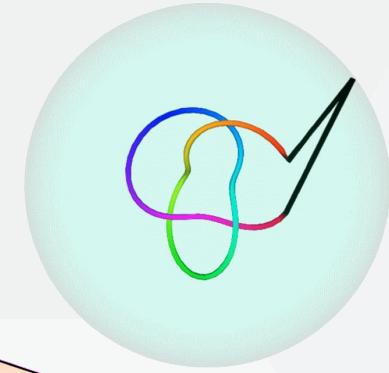
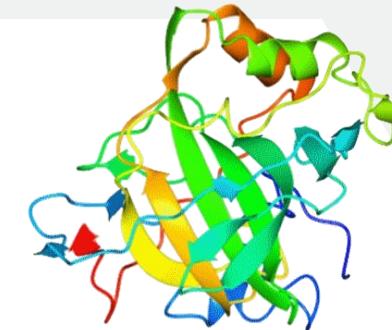


Figure "Prime Knots with seven or less crossings" source:
https://en.wikipedia.org/wiki/File:Knot_table.svg (accessed:
12.12.2022)

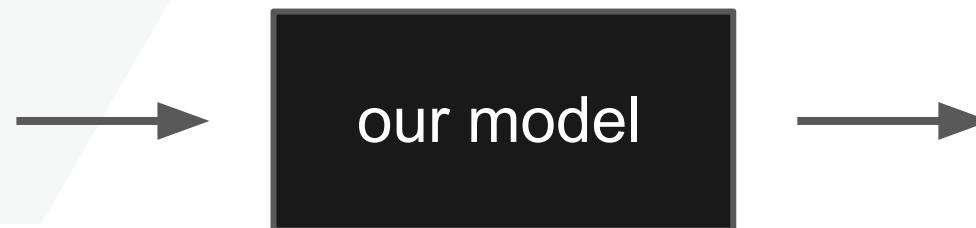


<https://www.nature.com/articles/srep42300>

Our Approach

The simplified problem we are trying to solve: **for a given protein sequence*** (string of amino acids) **decide whether it is knotted or unknotted** ($\text{== binary classification}$).

MSSPEARAGVKRPRPTQSL
PPPALPQLVAEQHTAIPP
TDKDSQRLLIVVLSNASLE
TYKASHGGTGRNGVQREE
KYSLLNSDEHIGVMR...



Probability of class 0: 0.9871515
Probability of class 1: 0.0128484

*no additional information (MSA, tertiary or quaternary structure) is provided to the model!

Our Approach

The simplified problem we are trying to solve: **for a given protein sequence*** (string of amino acids) **is it knotted or not?** **Knotted** (== *binary classification*).

MSSPEARAGVKRPRQSL
PPPALPQLVAEQHTAIPP
TDKDSQRLLIVVLSNASLE
TYKASHGGTGRNGVQREE
KYSLLNSDEHIGVMR...

Knot or Not? Sequence-Based Identification of Knotted Proteins With Machine Learning

Denisa Šrámková*, †,‡ Maciej Sikora*, ¶,§ Dawid Uchal, ¶,|| Eva Klimentová, †,‡

Agata P. Perlinska, ¶ Mai Lan Nguyen, ¶ Marta Korpacz, ¶,§ Roksana

Malinowska, ¶,§ Paweł Rubach, ¶,⊥ Petr Šimeček, ‡ and Joanna I. Sulkowska*, ¶

†*National Centre for Biomolecular Research, Faculty of Science, Masaryk University,
Kamenice 5, 625 00 Brno, Czech Republic*

‡*Central European Institute of Technology, Masaryk University, Brno, 62500, Czech
Republic*

*no additional information (MSA, tertiary or quaternary structure) is provided to the model!

of class 0: 0.9871515
of class 1: 0.0128484

Exercise

HuggingFace basics

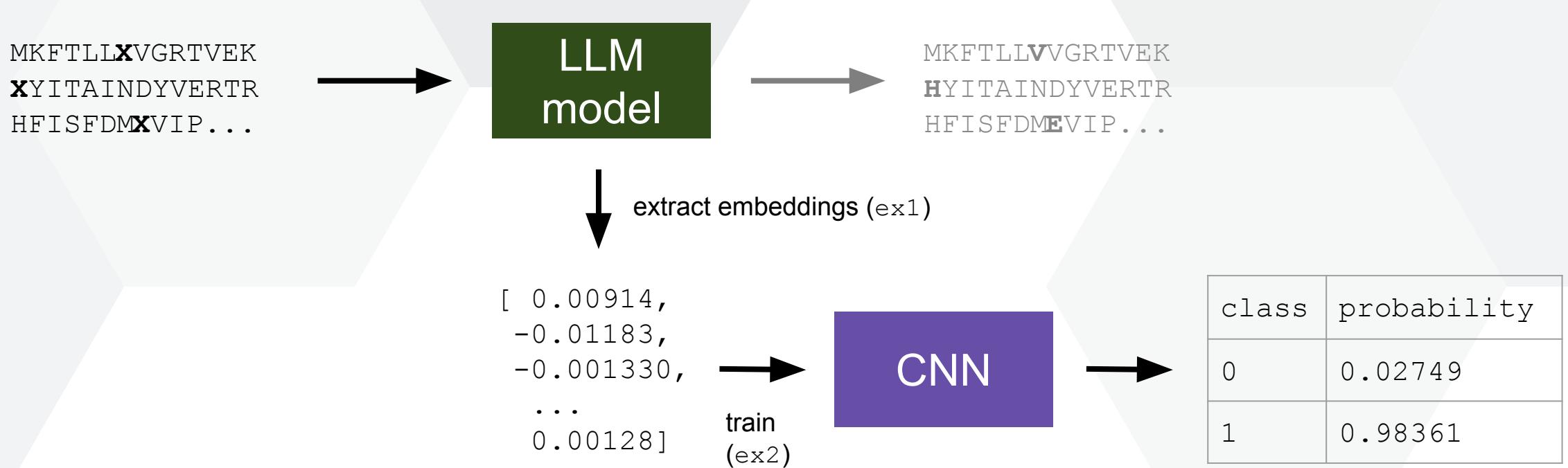
- maltaomics_ex0_huggingface.ipynb

Exercises

Extracting ProtBert-BFD embeddings

Training a CNN classifier on top of the embeddings

- maltaomics_ex1_embedding_extraction.ipynb
- maltaomics_ex2_embedding_training.ipynb



Drawbacks of using embeddings

```
[ 0.00914,  
-0.01183,  
-0.001330,  
...  
0.00128]
```

simple ML
model

class	probability
0	0.02749
1	0.98361

Drawbacks of using embeddings

[0.00914,
-0.01183,
-0.001330,
...
0.00128]

simple ML
model

class	probability
0	0.02749
1	0.98361

very good

Drawbacks of using embeddings

[0.00914,
-0.01183,
-0.001330,
...
0.00128]

simple ML
model

class	probability
0	0.02749
1	0.98361

very good

But why?

Drawbacks of using embeddings

[0.00914,
-0.01183,
-0.001330,
...
0.00128]

simple ML
model

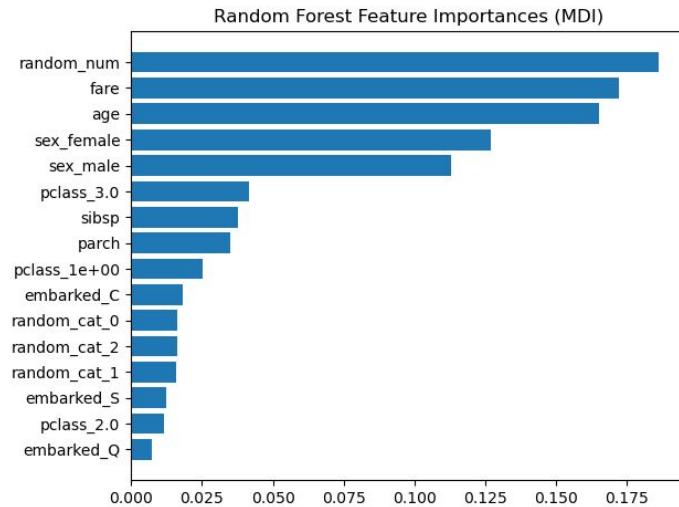
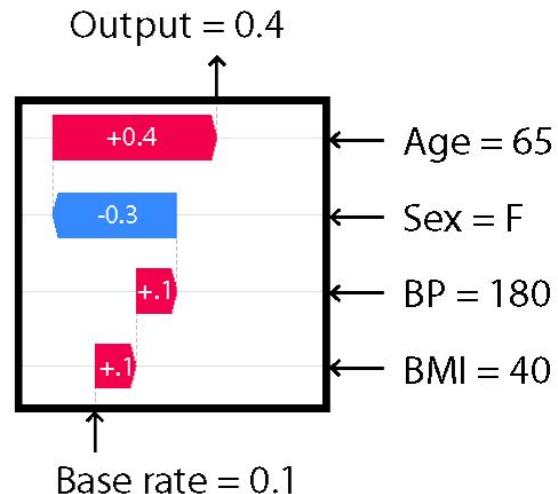
class	probability
0	0.02749
1	0.98361

very good

But why?



SHAP



Drawbacks of using embeddings

[0.00914,
-0.01183,
-0.001330,
...
0.00128]

simple ML
model

class	probability
0	0.02749
1	0.98361

very good

But why?



Output = 0.4

+0.4

Aae = 65

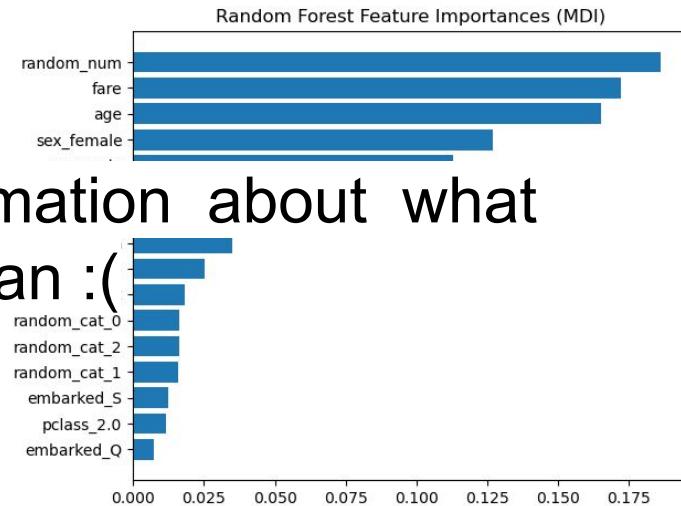
SHAP

Base rate = 0.1

+.1

BMI = 40

We don't have much information about what
do the embedding values mean :(



HuggingFace 😊 (<https://huggingface.co/>)

Finetuning a Model:

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification, Trainer,  
TrainingArguments  
  
tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, do_lower_case=False)  
model = AutoModel.from_pretrained(MODEL_NAME)  
  
trainer = Trainer(  
    model,  
    training_args,  
    train_dataset=tokenized_dss['train'],  
    eval_dataset=tokenized_dss['test'],  
    tokenizer=tokenizer,  
    compute_metrics=compute_metrics  
)  
trainer.train()
```

HuggingFace 😊 (<https://huggingface.co/>)

Finetuning a Model:

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification, Trainer,  
TrainingArguments  
  
tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)  
model = AutoModel.from_pretrained(MODEL_NAME)  
  
trainer = Trainer(  
    model,  
    training_args,  
    train_dataset=tokenized_dss['train'],  
    eval_dataset=tokenized_dss['test'],  
    tokenizer=tokenizer,  
    compute_metrics=compute_metrics  
)  
trainer.train()  
  
training_args = TrainingArguments(  
    output_dir='<OUTPUT_DIR_PATH>',  
    learning_rate=1e-5,  
    num_train_epochs=1,  
    weight_decay=0.01,  
    optim='adafactor'  
    ...)
```

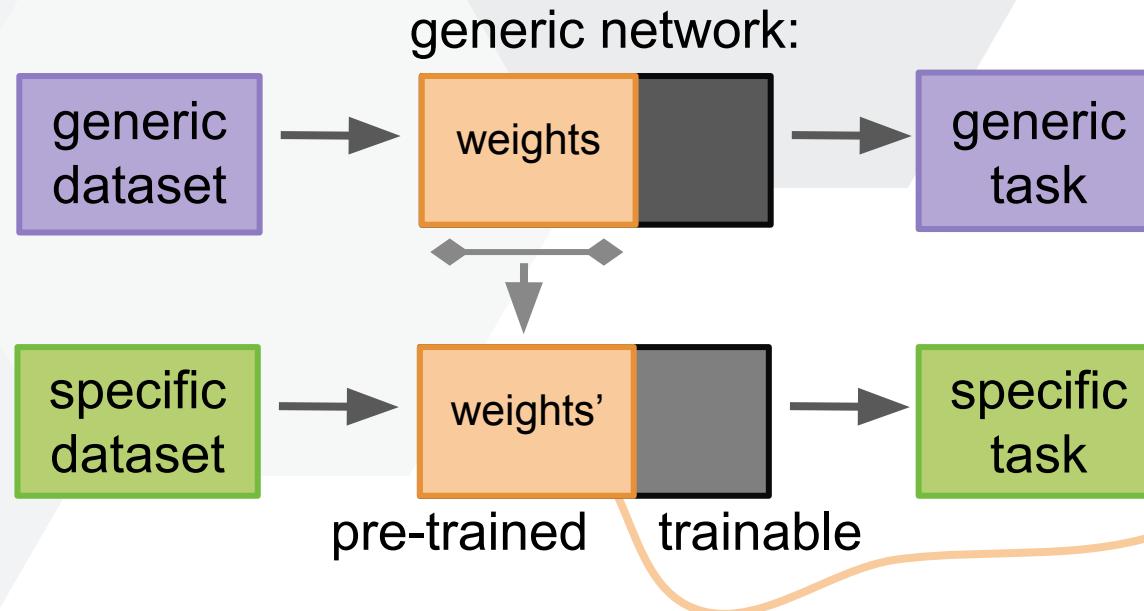
HuggingFace 😊 (<https://huggingface.co/>)

Finetuning a Model:

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification, Trainer,  
TrainingArguments  
  
tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, do_lower_case=False)  
model = AutoModel.from_pretrained(MODEL_NAME)  
  
trainer = Trainer(  
    model,  
    training_args,  
    train_dataset=tokenized_dss['train'],  
    eval_dataset=tokenized_dss['test'],  
    tokenizer=tokenizer,  
    compute_metrics=compute_metrics  
)  
trainer.train()  
  
from sklearn.metrics import accuracy_score  
  
def compute_metrics(eval_preds):  
    metric = evaluate.load('precision')  
    logits, labels = eval_preds  
    predictions = np.argmax(logits, axis=-1)  
    return {'accuracy': accuracy_score(labels, pred))}
```

Exercise

Finetuning a ProtBert-BFD model - maltaomics_ex3_finetuning.ipynb



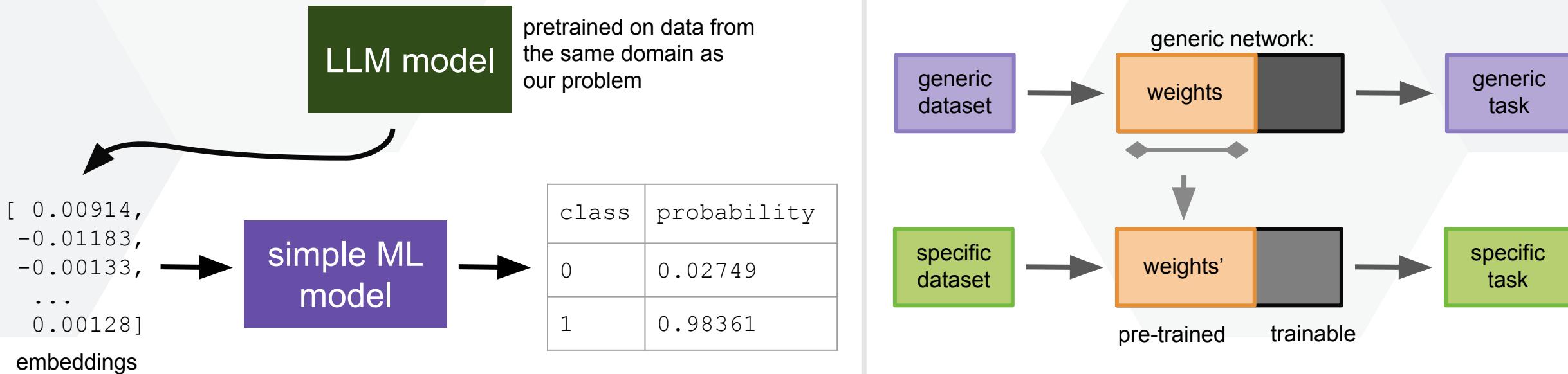
DistilProtBERT originally trained for distinguishing between real proteins and their randomly shuffled counterparts.
ProtBert-BFD on masked language modelling

Finetuning == approach to transfer learning in which weights of pre-trained model are trained on new data

- can be done on entire neural network, or on subset of its layers (other layers are “frozen”)

Summary

- we should get as much from already existing solutions as possible
- when we are solving protein structure problems it is possible to work only with amino acid sequences
- **2 approaches:**



References

- [1] Jumper et al. (2021). Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873), 583–589. <https://doi.org/10.1038/s41586-021-03819-2>
 - [2] Wu et al. (2022). High-resolution de novo structure prediction from primary sequence. *bioRxiv (Cold Spring Harbor Laboratory)*. <https://doi.org/10.1101/2022.07.21.500999>
 - [3] Lin et al. (2023). Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637), 1123–1130. <https://doi.org/10.1126/science.adc2574>
 - [4] *Benchmarking machine learning methods for protein folding: A comparative study of esmfold, Omegafold and alphafold* (no date) *End 2 End AI Molecule Design*. Available at: <https://310.ai/2023/05/17/benchmarking-machine-learning-methods-for-protein-folding-a-comparative-study-of-esmfold-omegafold-and-alphafold/> (Accessed: 01 September 2023).
 - [5] Elnaggar et al. (2022). ProtTrans: Toward understanding the language of life through self-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10), 7112–7127. <https://doi.org/10.1109/tpami.2021.3095381>
 - [6] Vaswani et al. (2017). Attention is All you Need. *arXiv (Cornell University)*, 30, 5998–6008. <https://arxiv.org/pdf/1706.03762v5>
 - [7] Dalla-Torre, Hugo, et al. The Nucleotide Transformer: Building and Evaluating Robust Foundation Models for Human Genomics. 15 Jan. 2023, <https://doi.org/10.1101/2023.01.11.523679>
- Pictures without direct citation were created using [BioRender](#).