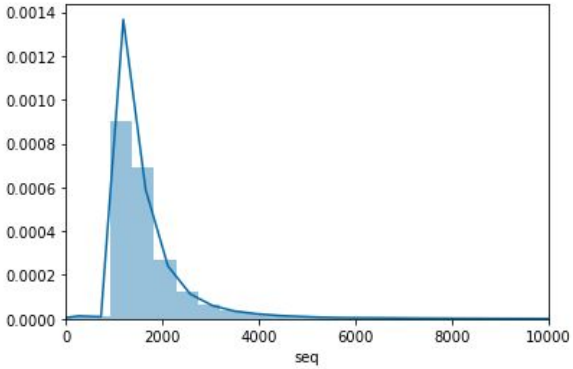
Overview

- Getting more background knowledge on encoder-decoder with attention models
- Building bi-directional **LSTM**. **Seq2Seq with attention** and **transformer encoder** model to train
- Compiling protein sequence datasets and annotations
- Visualizing embedding results

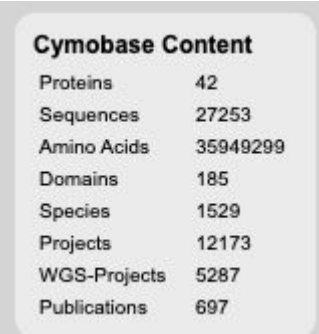
Datasets

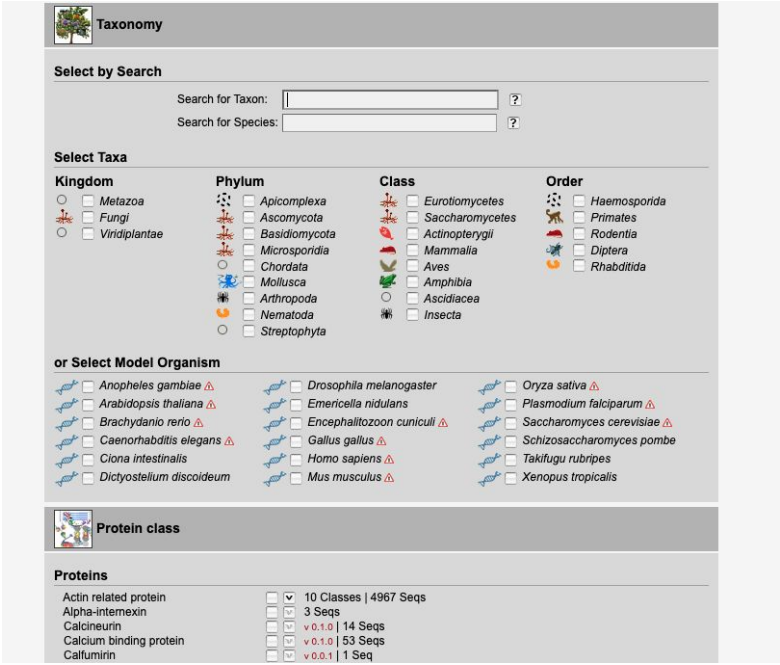
- **UniRef50**
  - UniRef100 contains all UniProt Knowledgebase records plus selected UniParc records (see below). It is generated by clustering identical sequences and subfragments.
  - UniRef50 is built by clustering UniRef90 seed sequences that have at least 50% sequence identity to and 80% overlap with the longest sequence in the cluster
  - **UniRef50_01:** 1/10 of UniRef50 data for faster training/developing, 846396 rows

	uniref50_ident	seq
0	UniRef50_A0A5A9P0L4	MEEITQIKKRLSQTVRLEGKEDLLSKKDSITNLKTEEHVSVKKMVI...
1	UniRef50_A0A410P257	MKAIAWLIVLTFLPEQVAWAVDYNLRGALHGAVAPLVSAATVATDG...
2	UniRef50_Q8WZ42	MTTQAPTFTQPLQSVVVLEGSTATFEAHISGFPVPEVSWFRDGQVI...
3	UniRef50_A0A672ZWI7	MVIHQRHTSDESFSSSPVEIRITAATPIPELAEERSAEKPPAVTET...
4	UniRef50_A0A4U5TZD8	MSTQAPTFTQPLQSVVALEGSAATFEAQVSGSPVPEVSWFRDGQVL...



- **Cymobase**
  - https://www.cymobase.org/cymobase [Cymobase: A database for cytoskeletal and motor protein sequence information]
  - Human curation and some level of annotations
  - Downloaded all the fasta files for each protein class, very tedious





- **Motor_toolkit**
- [https://link.springer.com/referenceworkentry/10.1007%2F978-3-642-16712-6_762] [https://www.sciencedirect.com/science/article/pii/S0092867403001119]
-
    - Sequences compiled for text searching uniprot swissprot on (basically getting all kinesin, dynein, and myosin_V)
        - 1. Conventional Kinesin: KIF5B, KHC, NKin, KLP1, KinA, DdK5
        - 2. Kinesin II: 1) heterodimer 2) homodimer: Osm3/KIF17 three Kinesin-2 subfamilies exist: the KIF3A, KIF3B/C, and KIF17 subfamilies
        - 3. Kinesin III: KIF1A, KIF1B, KIF13A, UNC104, DdUnc104
        - 4. Cytoplasmic dynein
        - 5. myosin V
    - **Kinesin_labeled:** for Kinesin, further annotate using a compiled name corresponds to different classes of kinesin, but most remain unlabeled

```
: motor_toolkit.groupby("type").count().iloc[:,0]
```

```
: type
dynein       665
kinesin     1055
myosin_v    1535
Name: Entry, dtype: int64
```

label	Entry
kinesin_1	15
kinesin_10	10
kinesin_11	5
kinesin_12	9
kinesin_13	7
kinesin_14a	6
kinesin_14b	9
kinesin_2	9
kinesin_3	16
kinesin_4	8
kinesin_5	13
kinesin_6	7
kinesin_7	3
kinesin_8	6
kinesin_9	6
unlabeled	926

- Df_dev
  - Entire dataset on the kaggle challenge for predicting pfam family
  - With the hope that dataset is random and balanced
  - The length of pfam sequence is shorter and more consistent
  - **1212912 entries in total, used as the training data on every model for now**
  - Df_dev_001: randomly 1/100 of datasets for t-SNE plotting as a general “background”

- PfamA_motors
  - Motor-related proteins belong to 4 superfamilies: actin-like, p loop gtpase, tubulin_c, tubulin binding
  - The dataset is the compilation of families from the above 4 clans from entire pfamA.fasta
  - PfamA_target:
  - PfamA_balanced: 4500 entries from each clan, but family-wise may not be balanced, use to inspect both how 4 clans mapped on to a general background and could serve as a background when inspecting embedding of each pfam family within the clan

```
1. actin-like:
    - PF00349    Hexokinase_1    2704    actin_like
    - PF00022    Actin    7689    actin_like
    - PF03727    Hexokinase_2    2859    actin_like
    - PF06723    MreB_Mbl    3621    actin_like
    - PF14450    FtsA    4161    actin_like
2. tubulin_c
    - PF03953    Tubulin_C    1999    tubulin_c
    - PF12327    FtsZ_C    2756    tubulin_c
3. tubulin_binding
    - PF00091    Tubulin    5951    tubulin_binding
    - PF10644    Misat_Tub_SegII    628    tubulin_binding
    - PF13809    Tubulin_2    396    tubulin_binding
    - PF14881    Tubulin_3    660    tubulin_binding
4. p_loop_gtpase
    - PF00063    Myosin_head    8304    p_loop_gtpase
    - PF00225    Kinesin    15099    p_loop_gtpase
    - PF03028    Dynein_heavy    2322    p_loop_gtpase
```

```
pfamtarget.groupby("pfamA_name").count().iloc[:,0]
```

```
pfamA_name
Actin                7689
Dynein_heavy         2322
FtsA                 4161
FtsZ_C               2756
Hexokinase_1         2704
Hexokinase_2         2859
Kinesin             15099
Misat_Tub_SegII       628
MreB_Mbl             3621
Myosin_head          8304
Tubulin              5951
Tubulin_2             396
Tubulin_3             660
Tubulin_C            1999
Name: Unnamed: 0, dtype: int64
```

```
pfamA_motors.groupby("clan_x").cou
```

```
clan_x
actin_like          175087
p_loop_gtpase      1727354
tubulin_binding       7635
tubulin_c             4755
Name: Unnamed: 0, dtype: int64
```

```
pfamtarget.groupby("clan_x").co
```

```
clan_x
actin_like          21034
p_loop_gtpase       25725
tubulin_binding      7635
tubulin_c            4755
Name: Unnamed: 0, dtype: int64
```

Models

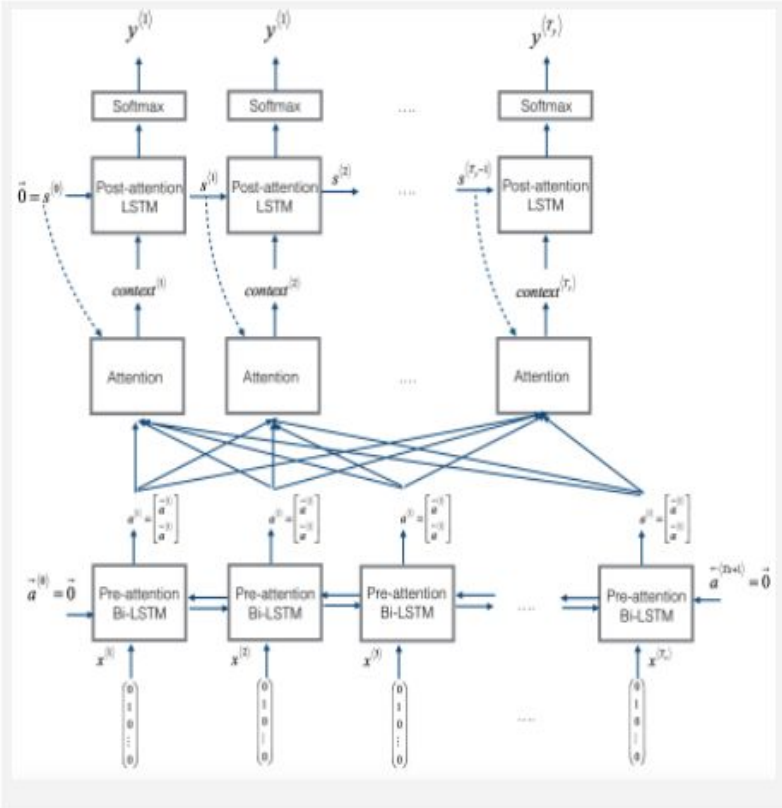
LSTM2: 64*2 hidden states, training to predict the clan, trained on pfamA_balanced, reached a test accuracy of 94%

LSTM5: 64*2 hidden states, next token prediction, trained on df_dev

Seq2seq_with_attention:

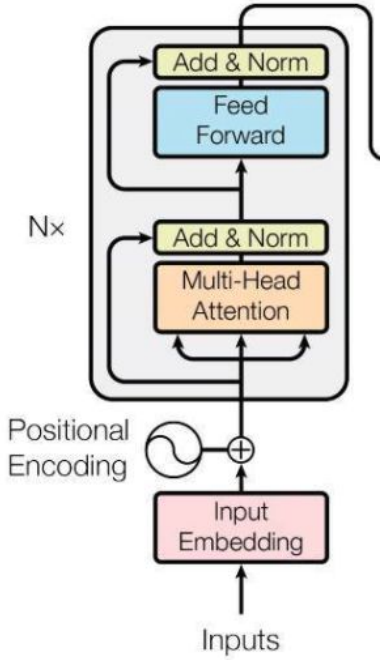
- Basic architecture adopted from https://arxiv.org/abs/1506.03134
- Attention layer adopted from https://arxiv.org/pdf/1409.0473.pdf
- Nice tutorial at: https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/
- Trained much slow as instead of using the last hidden state, it uses a weighted sum of all hidden states (length of sequence dependent)
- Stopped training when after passing 230k sequence in df_dev
- **Widely used in neural machine translation but confused when if comes to our case (it is good for studying 2 pair of languages with internal alignment in between)**

$$p(\mathbf{y}) = \prod_{t=1}^{T} p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c),$$



Transformer_encoder

- To start with a try, adopt the minimized roberta hyperparameters
- Encoder connected to a fully connected layer to output softmax of next token prediction
- 21-->768 embedding
- 6 layers of nn.TransformerEncoderLayer
- 12 heads for attention learning
- nhid = 200 # the dimension of the feedforward network model
- Looking back, the embedding dimension might be too big and useless in my training



Training

Training Data: all on df_dev, without any evo-tune, except for LSTM2

Time to train: varies but for now all within 24 hours for 1 epoch

Loss: does not converge, storing loss vector may lead to memory error

Training >=3 models on hpc may lead to cuda memory issue

Dimensionality Reduction

Why TSNE for now: try to preserve local neighborhood by minimizing KL divergence

Why first PCA: TSNE is computationally heavy and sklearn suggests to first perform PCA to reduce dimensionality on large data

Several datasets are very unbalanced in that a pfam family or a clan dominates, and the learned coordinates might not be ideal for clustering

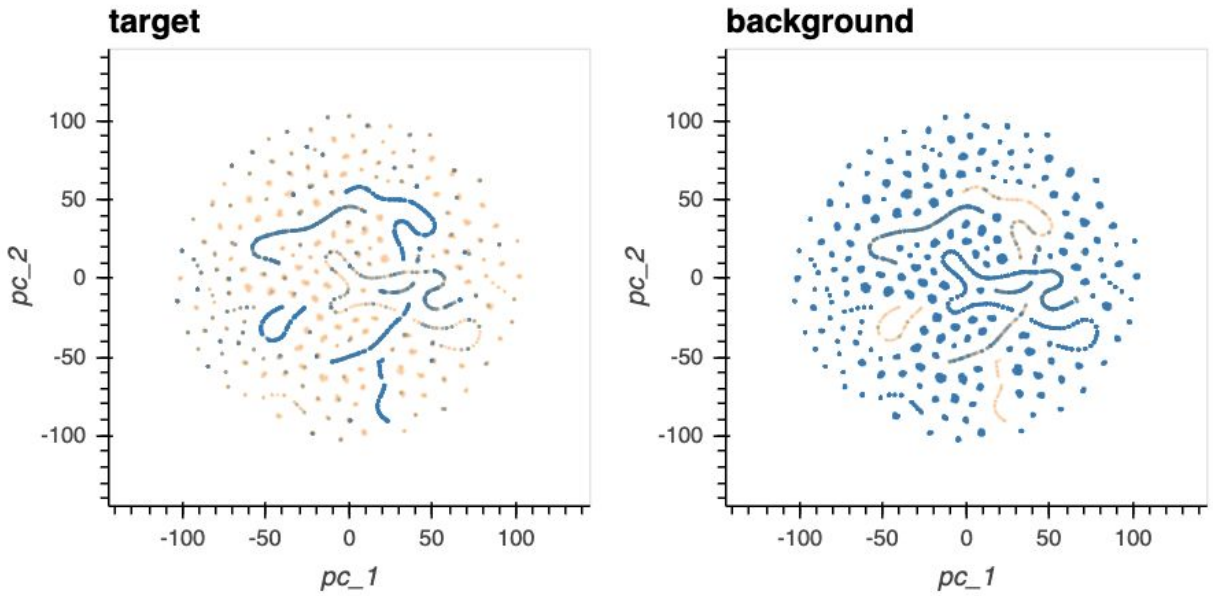
Plot t-SNE result

TO-DO

- More balanced training dataset
- Evo-tuning on motor proteins
- Directly train on motor proteins
- Supervised learning (on kinesin type, for example)
- Fine tune training hyperparameters
- More balanced dataset to be visualized
- Other dimensionality reduction methods
- Testing on proteins with phylogenetic relation/protein at different stability level

0. Inspect the PCs variance for transformer model
1. More balanced training dataset
2. Evo-tuning on motor proteins
3. Directly train on motor proteins
4. Supervised learning (on kinesin type, for example)
5. Fine tune training hyperparameters (for transformer especially)
6. More balanced dataset to be visualized
7. Other dimensionality reduction methods
8. Testing on proteins with phylogenetic relation/protein at different stability level

As of last presentation: Transformer TSNE



Transformer SVD Analysis

Transformer Encoder

```
hn_motortoolkit = np.load("../data/hn_transformerencoder_motortoolkit.npy")
hn_pfammotors= np.load("../data/hn_transformerencoder_pfammotors.npy")
hn_dfdev = np.load("../data/hn_transformerencoder_dfdev.npy")
print(hn_motortoolkit.shape)
print(hn_pfammotors.shape)
print(hn_dfdev.shape)
```

```
(3235, 768)
(1914831, 768)
(1212912, 768)
```

```
u, s, v = np.linalg.svd(hn_motortoolkit)
```

```
s[0:10]
```

```
array([1.6293489e+05, 2.0418205e+00, 1.3351089e+00, 1.0232561e+00,
       8.3068514e-01, 6.5283227e-01, 5.3178918e-01, 4.0863904e-01,
       3.9062384e-01, 3.7214699e-01], dtype=float32)
```

```
u, s, v = np.linalg.svd(hn_pfammotors[1:10000,:])
```

```
s[0:10]
```

```
array([8.9091117e+04, 7.2454149e-01, 3.5801572e-01, 3.0840611e-01,
       2.6150218e-01, 2.3170654e-01, 1.9952329e-01, 1.9444086e-01,
       1.7441043e-01, 1.5912345e-01], dtype=float32)
```

```
u, s, v = np.linalg.svd(hn_dfdev[110000:120000,:])
```

```
s[0:10]
```

```
s[0:10]
```

```
array([4.9420723e+04, 4.5896345e-01, 4.5610082e-01, 2.8848442e-01,
       2.5892946e-01, 2.1507658e-01, 2.0485921e-01, 1.7426261e-01,
       1.5409079e-01, 1.3732859e-01], dtype=float32)
```

Seq2Seq

```
hn_motortoolkit = np.load("../data/hn_s2sencoder_motortoolkit.npy")
hn_pfammotors= np.load("../data/hn_s2sencoder_pfammotors.npy")
hn_dfdev = np.load("../data/hn_s2sencoder_dfdev.npy")
```

```
u, s, v = np.linalg.svd(hn_motortoolkit)
```

```
s[0:10]
```

```
array([257.47748 ,  85.86048 ,  56.964283,  40.470497,  37.40044 ,
        30.117552,  25.10778 ,  23.10032 ,  22.317842,  20.853163],
      dtype=float32)
```

```
u, s, v = np.linalg.svd(hn_pfammotors[1:20000,:])
```

```
s[0:10]
```

```
array([585.71173 , 277.25177 , 176.3308  , 112.73699 , 102.46996 ,
        96.50056 ,  77.918335,  73.30223 ,  59.386765,  55.59358 ],
      dtype=float32)
```

```
u, s, v = np.linalg.svd(hn_dfdev[110000:120000,:])
```

```
s[0:10]
```

```
s[0:10]
```

```
array([400.16098 , 190.04813 , 113.94885 ,  84.33569 ,  70.39741 ,
        68.85091 ,  58.15323 ,  51.65772 ,  43.141678,  41.207542],
      dtype=float32)
```

LSTM5

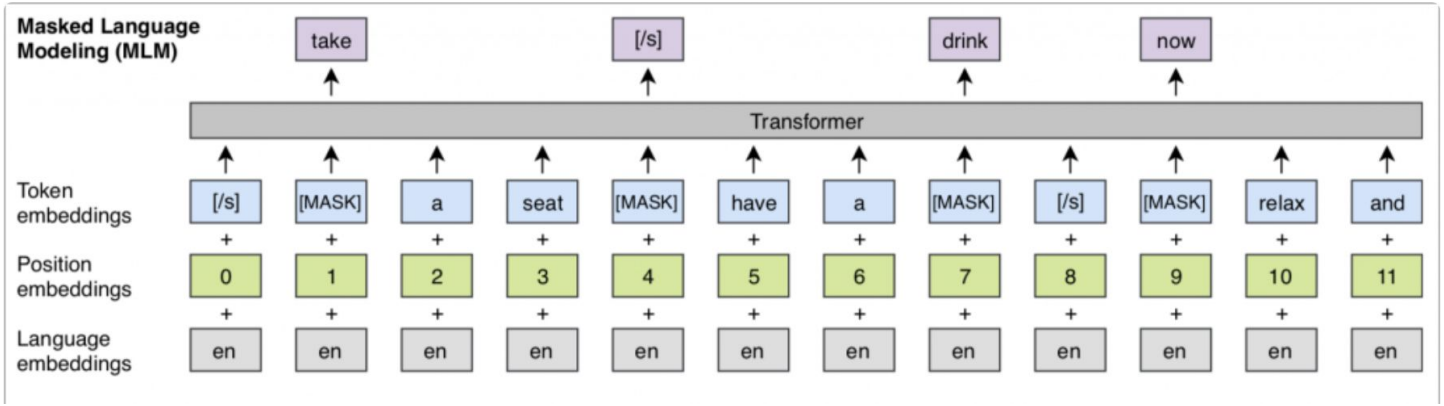
```
hn_motortoolkit = np.load("../data/hn_lstm5_motortoolkit.npy")
hn_pfammotors= np.load("../data/hn_lstm5_pfammotors.npy")
hn_dfdev = np.load("../data/hn_lstm5_dfdev.npy")
print(hn_motortoolkit.shape)
print(hn_pfammotors.shape)
print(hn_dfdev.shape)
```

```
(3255, 256)
(1914831, 256)
(1212912, 256)
```

```
array([526.8835  , 189.43001 , 160.37401 , 155.0355  , 129.91481 ,
       120.06838 , 117.729546, 114.029755, 110.14683 , 102.44181 ],
      dtype=float32)
```

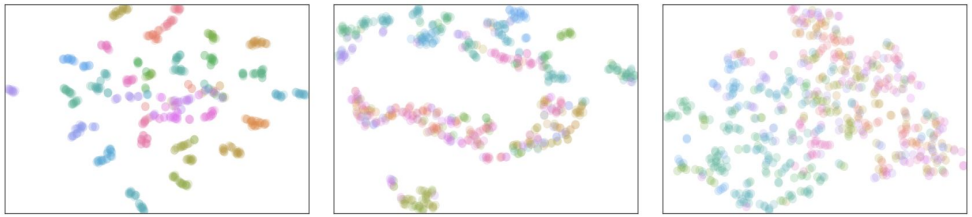
MLM transformer encoder model

- In the previous model, a mask is applied so that at each word, it sees only the word prior to it. (When predicting #2 word, it sees only #1 word, and all rest are masked to -INF), so that the model follows an auto-regressive manner
-
- In the MLM setting, certain proportion of the sentence is randomly masked (15% in BERT), and they are masked throughout the training process. The loss is only on those positions' correctness. Out of the 15% of masked positions, 80% are changed to [MASK] token, 10% is randomized to a AA, and 10% to be remained the same



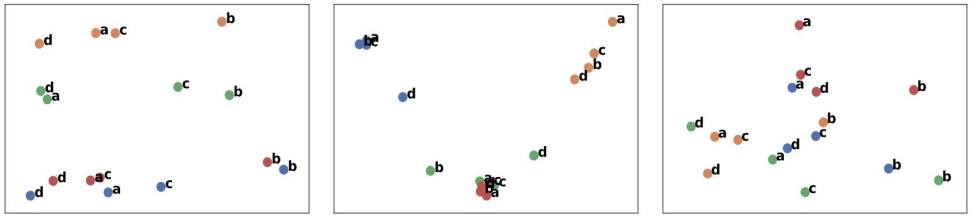
Face

l model



Transformer (trained) Transformer (untrained) Unigram

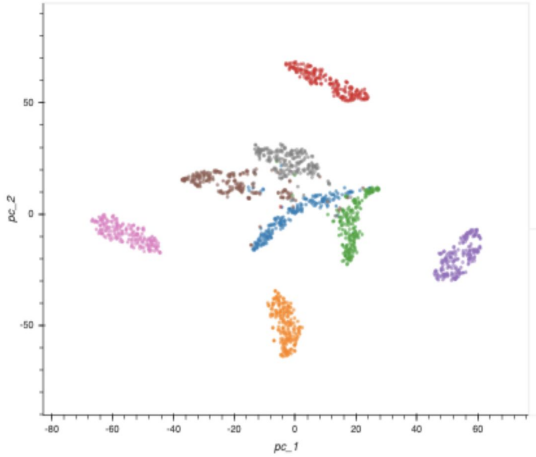
(a) Sequence representations (t-SNE)

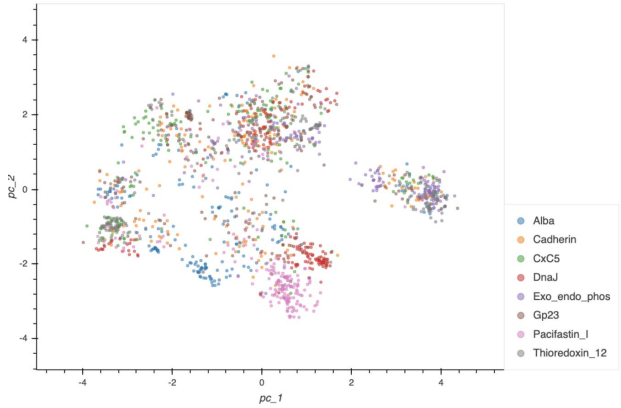


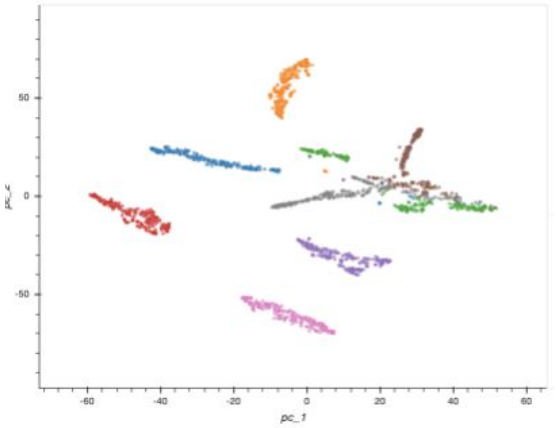
Transformer (trained) Transformer (untrained) Unigram

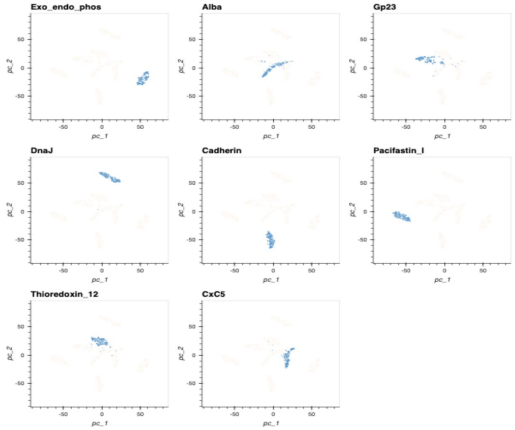
(b) Sequence representations (PCA)

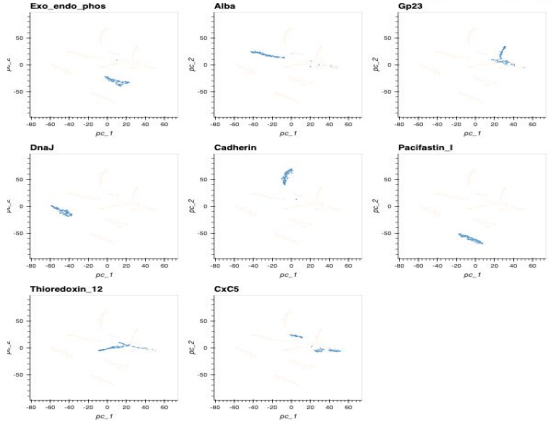
Figure 2. Protein sequence representations encode and organize biological variations. (a) Each point represents a gene, and each gene is colored by the orthologous group it belongs to (dimensionality is reduced by t-SNE). Orthologous groups of genes are densely clustered in the trained representation space. By contrast, the untrained representation space and unigram representations do not reflect strong organization by evolutionary relationships. (b) Genes corresponding to a common biological variation are related linearly in the trained representation space. Genes are colored by their orthologous group, and their species is indicated by a character label. PCA recovers a species axis (horizontal) and orthology axis (vertical) in the trained representation space, but not in the untrained or unigram spaces. Representations are from the 36-layer Transformer model trained on UniParc.


pc_2
pc_1

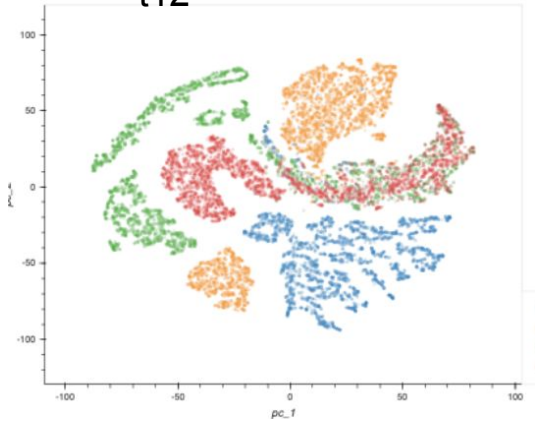

pc_2
pc_1
Alba
Cadherin
CxC5
DnaJ
Exo_endo_phos
Gp23
Pacifastin_I
Thioredoxin_12

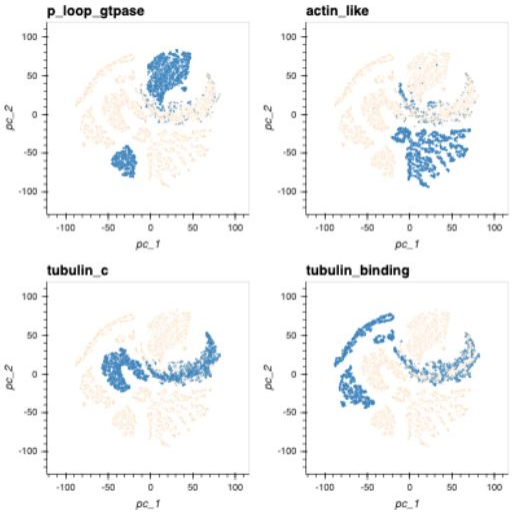

pc_2
pc_1


Exo_endo_phos
Alba
Gp23
DnaJ
Cadherin
Pacifastin_I
Thioredoxin_12
CxC5
pc_1
pc_2

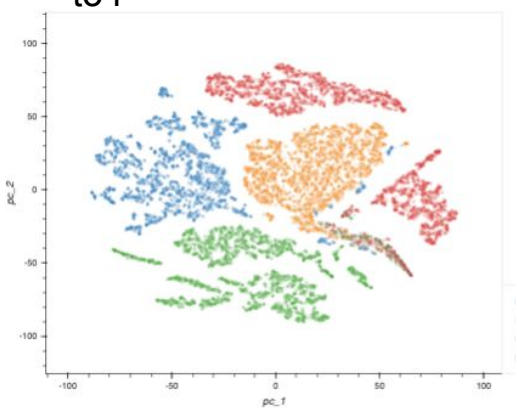

Exo_endo_phos
Alba
Gp23
DnaJ
Cadherin
Pacifastin_I
Thioredoxin_12
CxC5
pc_1
pc_2

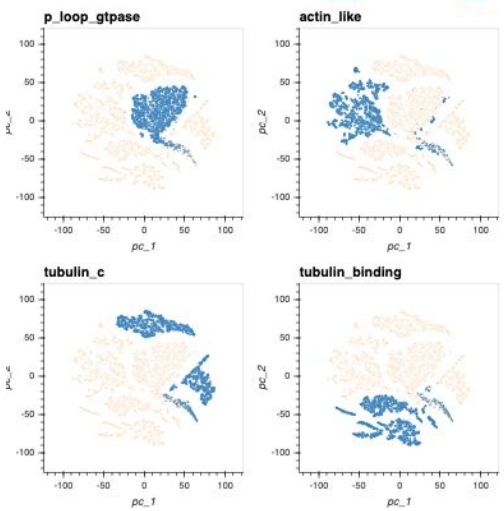
t12



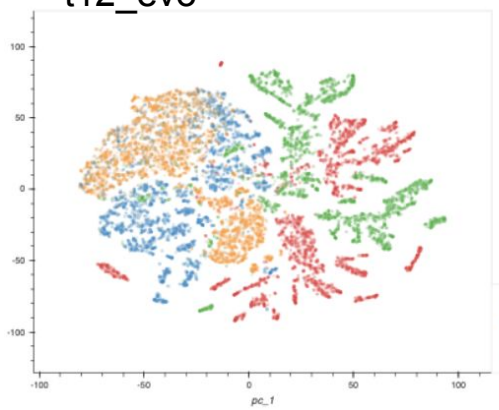


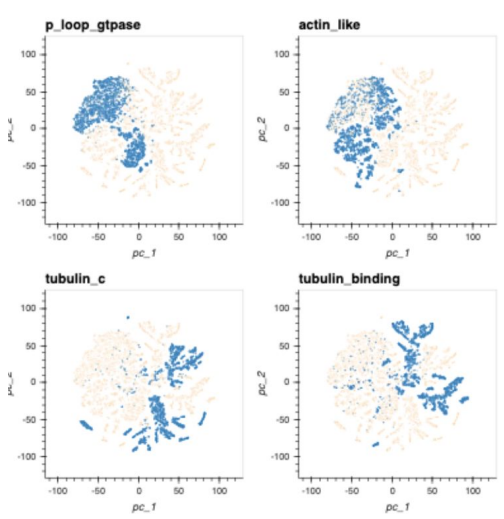
t34

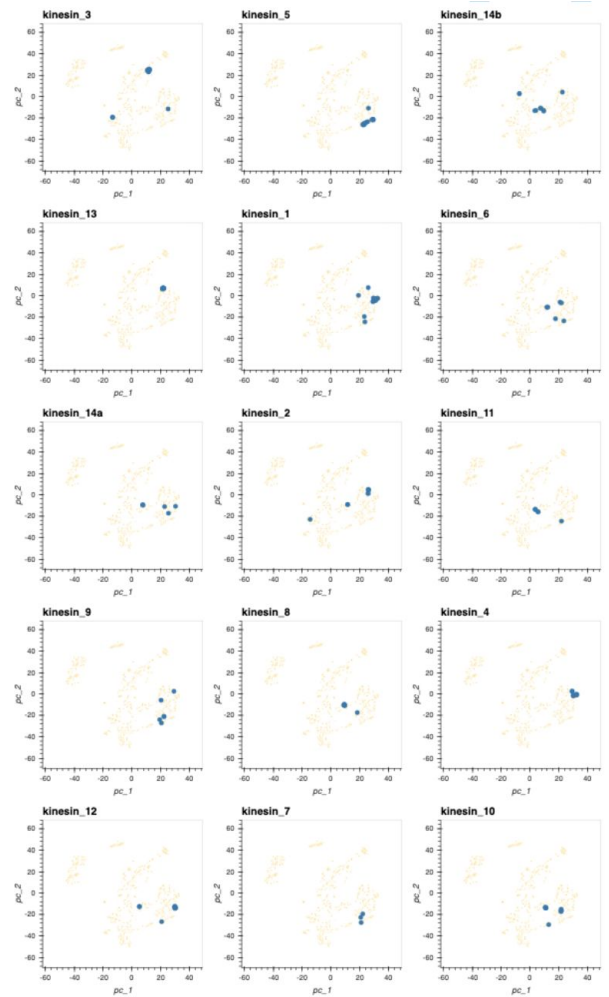




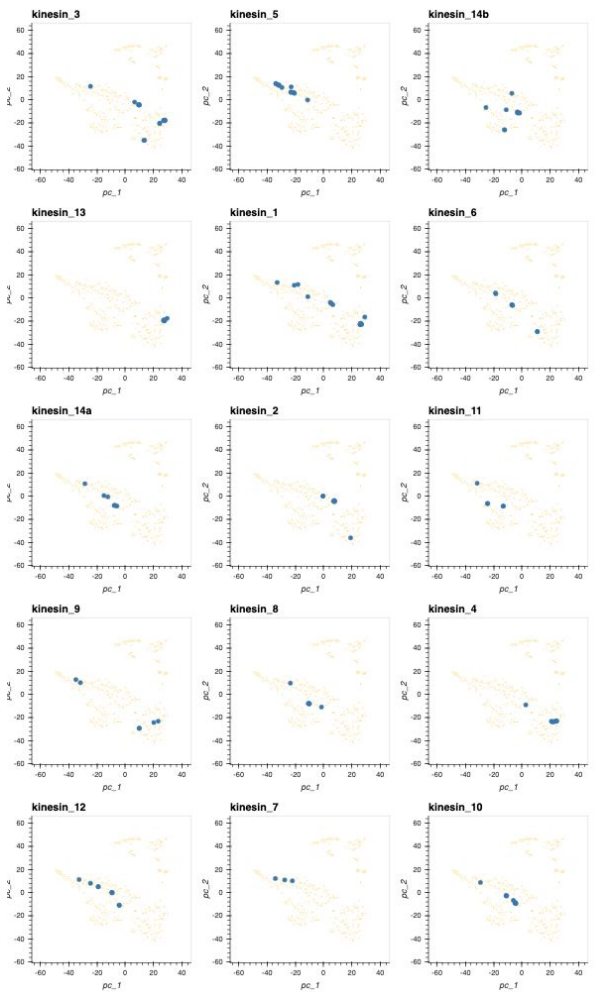
t12_evo



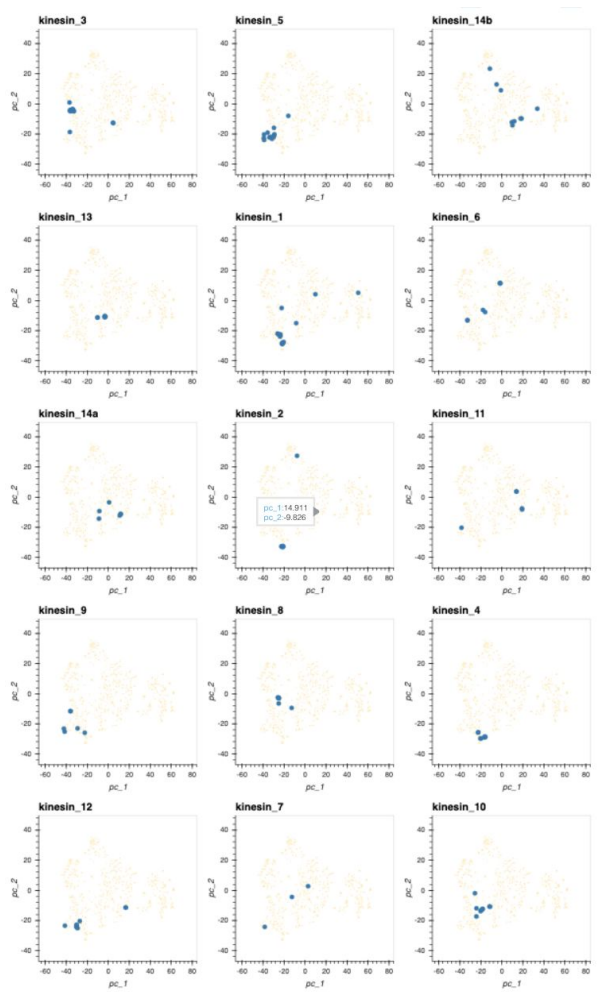




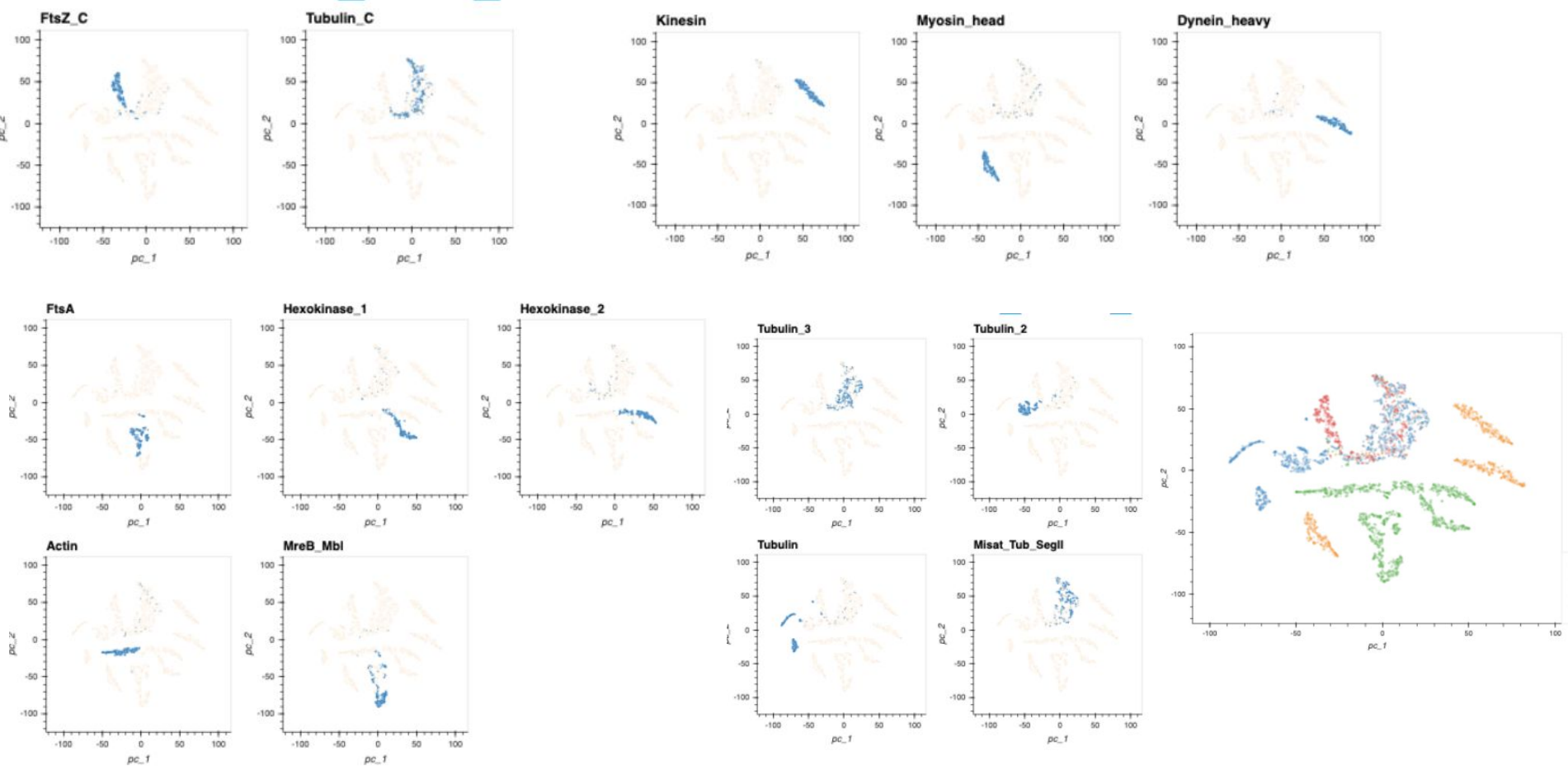
t12



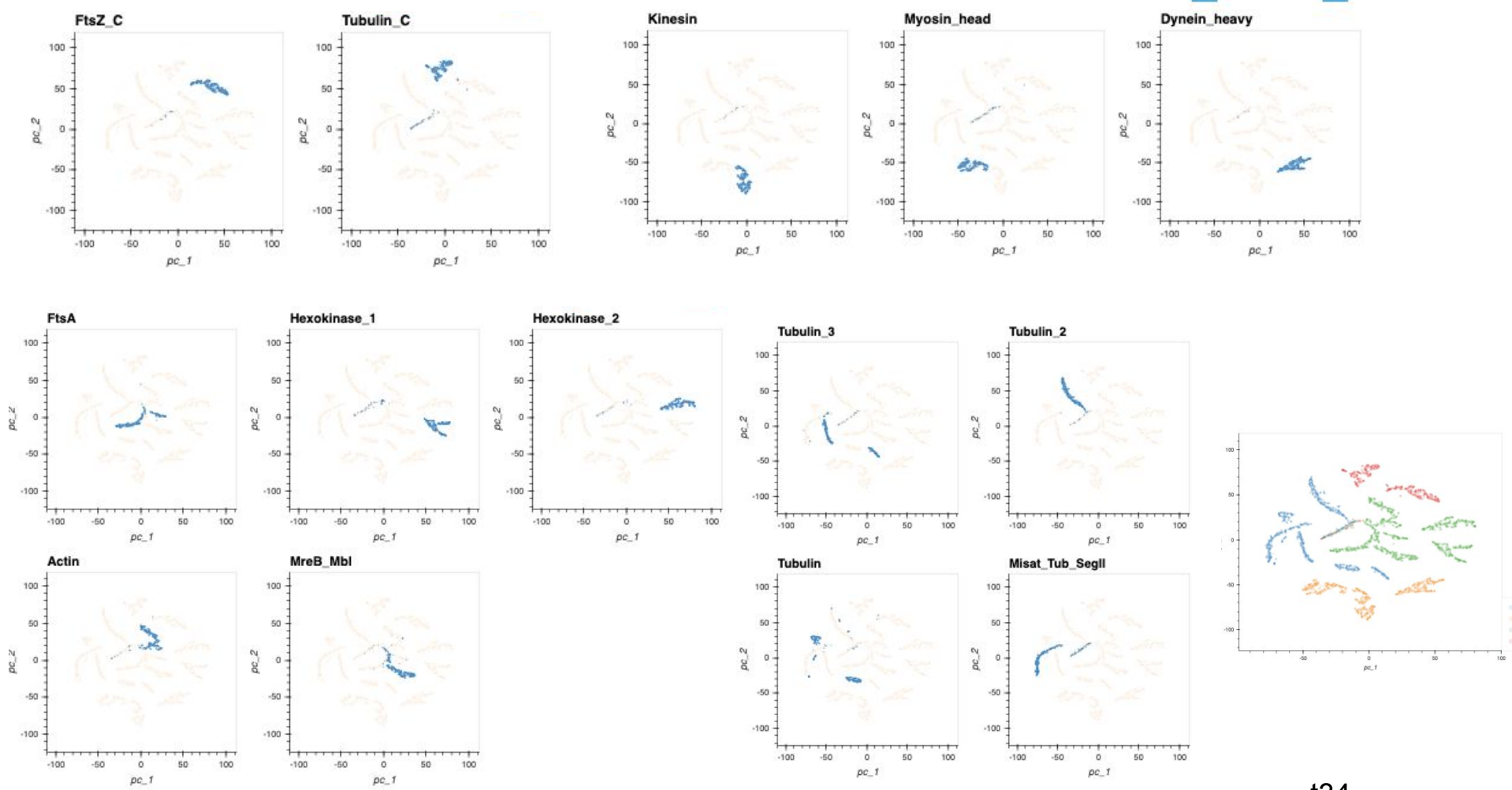
t34



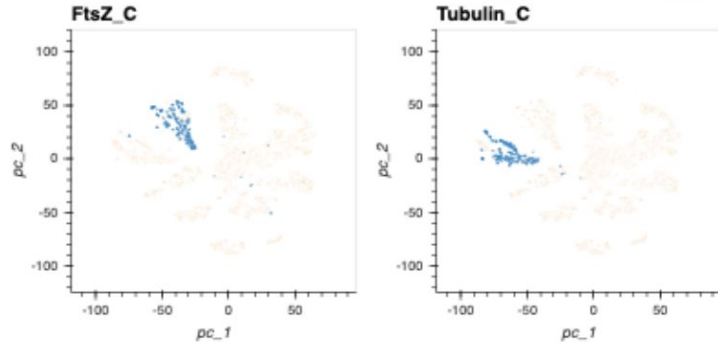
t12_evo

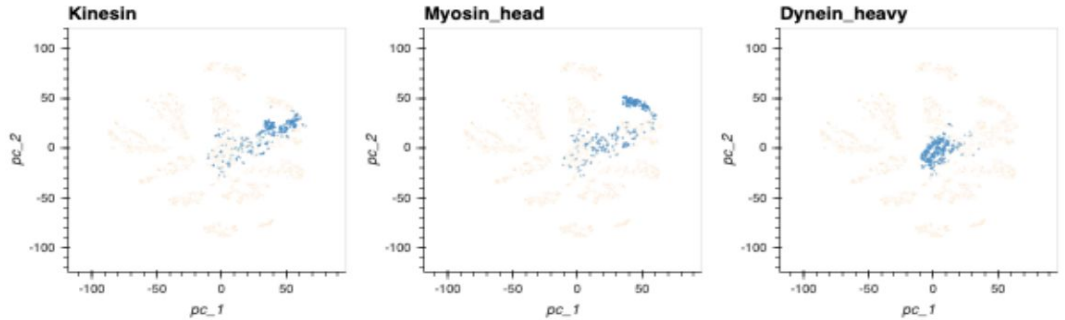

FtsZ_C
Tubulin_C
Kinesin
Myosin_head
Dynein_heavy
FtsA
Hexokinase_1
Hexokinase_2
Tubulin_3
Tubulin_2
Actin
MreB_Mbl
Tubulin
Misat_Tub_SegII
pc_1
pc_2

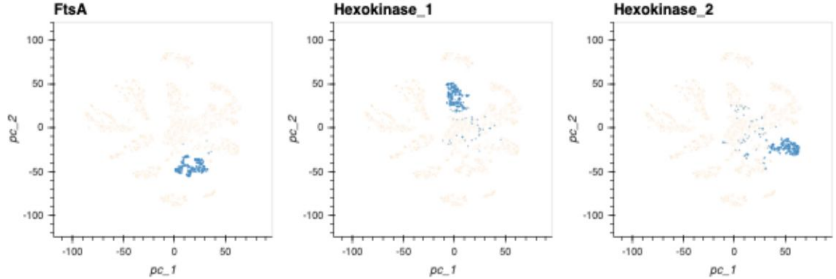
t12

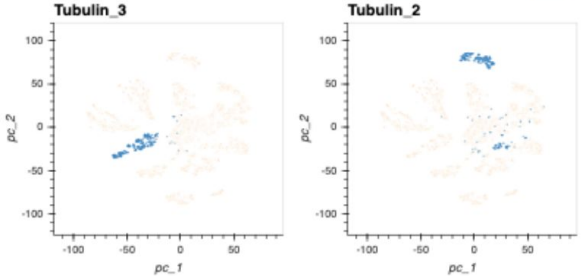

FtsZ_C
Tubulin_C
Kinesin
Myosin_head
Dynein_heavy
FtsA
Hexokinase_1
Hexokinase_2
Tubulin_3
Tubulin_2
Actin
MreB_Mbl
Tubulin
Misat_Tub_SegII
pc_1
pc_2

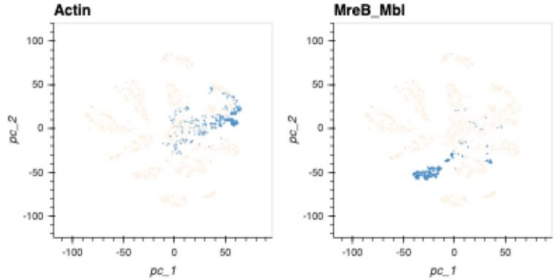
t34

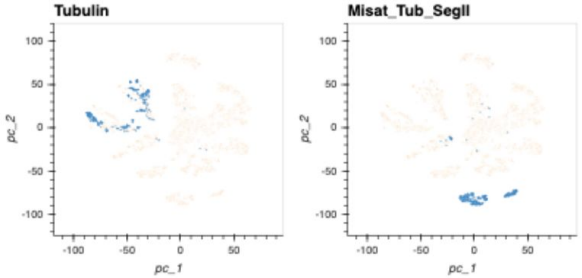

FtsZ_C
Tubulin_C
pc_1
pc_2


Kinesin
Myosin_head
Dynein_heavy
pc_1
pc_2

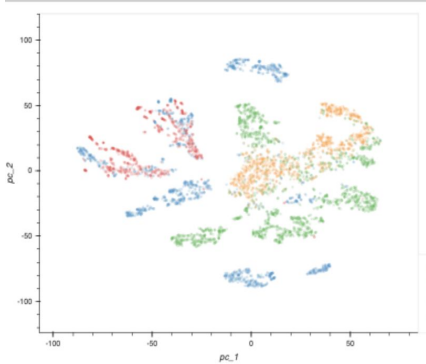

FtsA
Hexokinase_1
Hexokinase_2
pc_1
pc_2


Tubulin_3
Tubulin_2
pc_1
pc_2


Actin
MreB_Mbl
pc_1
pc_2


Tubulin
Misat_Tub_SegII
pc_1
pc_2

t12_evo


pc_1
pc_2

ProGen :Language modelling for protein generation

ProGen: Language Modeling for Protein Generation

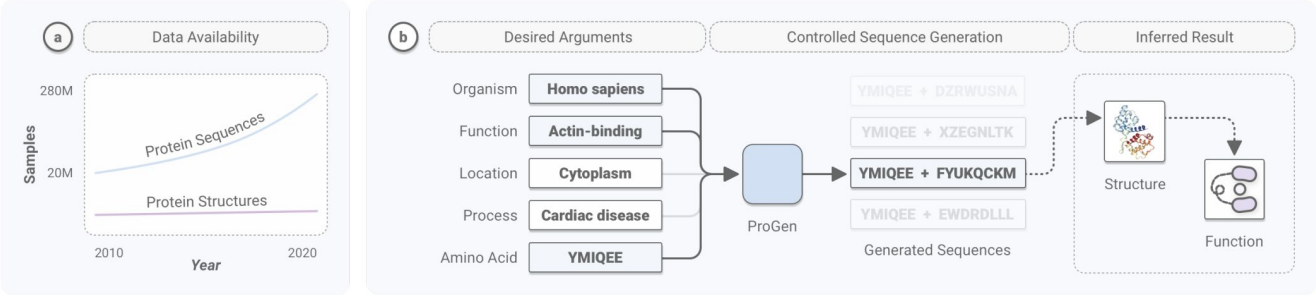


Figure 1. a) Protein sequence data is growing exponentially as compared to structural data. b) We utilize protein sequence data along with taxonomic and keyword tags to develop a conditional language model: ProGen.

- Include sequence and its inverse (invariant to temporal notion of sequence generation)
- Pretend protein conditioning tags from databases (biased toward SWISSPROT)
- Embedding 1028, inner dim 512, 36 layer, 8 attention heads per layer
- Generate protein variants with high fitness score than random mutation
- Lower sequence similarity but better secondary structural accuracy with more than 8 conditioning tags
- Input a short context sequence, then sampled using top-k, repetition penalty

3D deep convolutional neural networks for amino acid environment similarity analysis

Wen Torng & Russ B. Altman

BMC Bioinformatics **18**, Article number: 302 (2017) | Cite this article

Abstract

Background

Central to protein biology is the understanding of how structural elements give rise to observed function. The surfeit of protein structural data enables development of computational methods to systematically derive rules governing structural-functional relationships. However, performance of these methods depends critically on the choice of protein structural representation. Most current methods rely on features that are manually selected based on knowledge about protein structures. These are often general-purpose but not optimized for the specific application of interest.

- With 3D convolutional data on available structures to perform semi-supervision
- More dataset/properties to be mapped and tested (stability)
- !! Standardization before PCA
- CASP/CAGI challenge dataset

Thermophilic/Non-thermophilic prediction

ProtDataTherm:

A database for thermostability analysis and engineering of proteins

- Paper: [https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5788348/#!po=50.0000]
- DBSource: [https://drive.google.com/drive/folders/1oOkUYSJXy8NUP1sQ5U0GQFNvSuwdjU67]
- claims to offer ~600k thermostable (Tm >40°C) proteins categorized by domain families (Pfams).

Generate embeddings

- pdt_motor.csv: The overlap entries between ProtDataTherm and the four pfam superfamilies relating to motor proteins (p loop gtpase, actin like, tubulin c, tubulin binding)
- Generate embedding using transformer model ESM for both 12 layers, 34 layers
- The embeddings are mapped to 1280 and 768 dimensions respectively

Training Classification model

- Try create a balanced training set by sampling the same number of min(thermophilic, non-thermophilic) of a family. For now do no sample from a family is it does not contain one of the classes¶
- The raw embedding, the re-scale the embedding dimensions, also a reduced-dimension version using PCA are used for downstream classification task
- For each of the entry, classify whether the entry is from a thermophilic organism or not, which in this context indicating the thermostability of the organism
- A 80%-20% train-test split were used, test acc were to evaluate the model

clan	is_thermophilic	uniprot_id
actin_like	0	3604
	1	3604
p_loop_gtpase	0	24382
	1	24382
tubulin_binding	0	278
	1	278
tubulin_c	0	2
	1	2

Result

	T12	T34
Soft SVM	0.718	0.760
Logistic Regression	0.763	0.802
kNN classifier	0.818	0.837
NN	0.834	0.857

Visualize binary thermophilic labels in embedded space

Since most of the entries are from p_loop_gtpase, to visualize binary labels from top 3 families within p loop gtpase

- top_fam = df_plot.loc[df_plot["clan"]=="p_loop_gtpase",:].groupby("pfam_id").count().reset_index().sort_values(['uniprot_id'], ascending=False).head(3)["pfam_id"]

PF00005 PF01926 PF01656 ALL

Observation: The thermophilic and non-thermophilic proteins are first grouped by pfam families, but form sub-clusters like populations within each pfam families based on thermophilic labels

Kinesin Superfamily Embedding

- **KIF Home Page from Duke** https://sites.duke.edu/kinesin/
- **KIF Home Page from U Tokyo** http://cb.m.u-tokyo.ac.jp/KIF/
- **Motor toolkit:** Regular Expression Search of kinesin gene names on Uniprot
- U Tokyo dataset provided emsembl accession to each entry, uniprot entries are directly queried by using the provided accession
- Duke dataset does not contain any accession information, but contain amino acid sequences for kinesin domains. BLASTp against all the sequences is very time-ineffective. Instead, the unique information in the dataset is the kinesin direction and speed info. For the 20 entries that have documented mobility information, use BLASTp then uniprot query to retrieve their full sequences
- The two KIF database contains curated kinesin family information, whereas the Motor toolkit dataset's kinesin family information is based on regex matching to the gene names

Compiled Kinesin Dataset

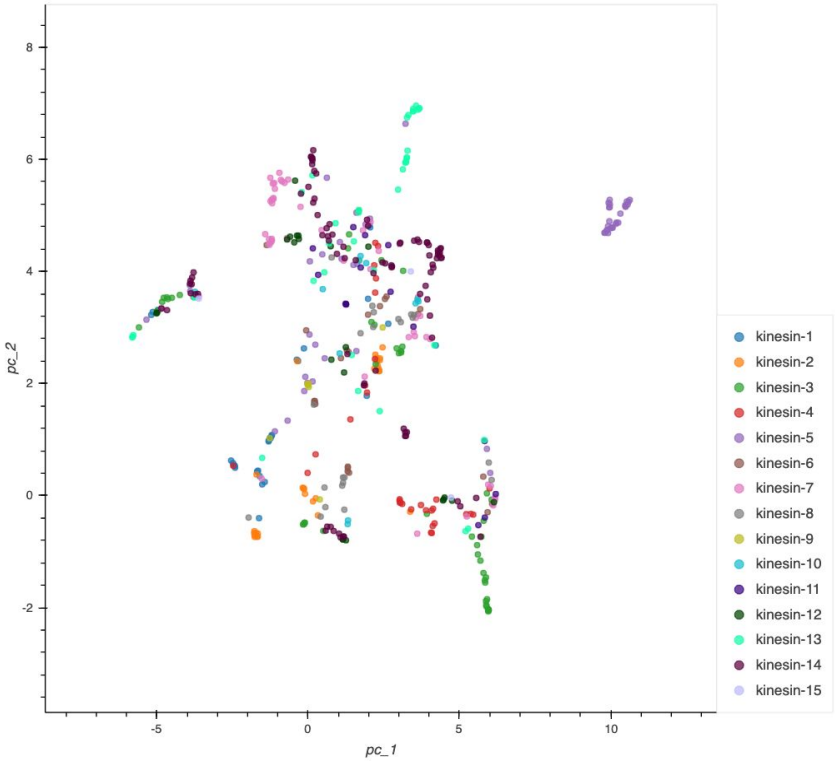
- 0: unlabelled kinesin family
- 1~14: the labelled kinesin families
- 15: Orphan
- 16: Kinesin Light Chain (KLC)

db_name	#Entry
kif_duke	85
kif_jp	456
motor_toolkit	57

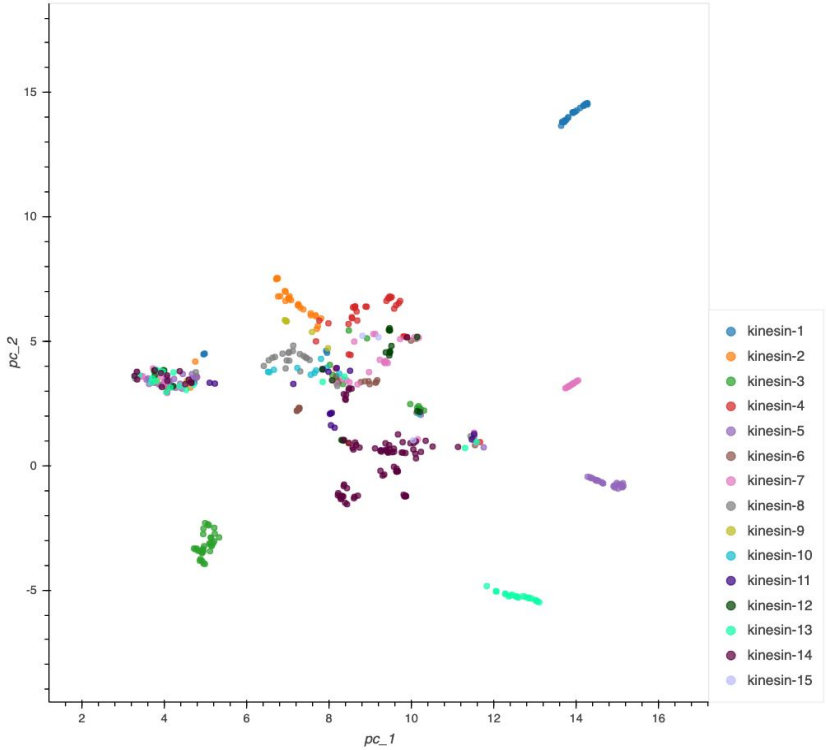
Weight Tuning and 2D dimension reduction

- **Following variations of ESM model are used:**
  - 34-layer raw ESM model (evotune on t34 model is computationally infeasible)
  - 12-layer raw ESM model
  - 12-layer raw ESM model with dynein + KIF sequence
  - 12-layer raw ESM model with KIF sequence only
- **Following normalization are applied:**
  - Scaled: each dimension of protein embedding is standardized
  - Reduced: retrieve the first #N PCs that captured >90% variance
- **Following visualization are generated:**
  - 2D PCA
  - TSNE
  - UMAP

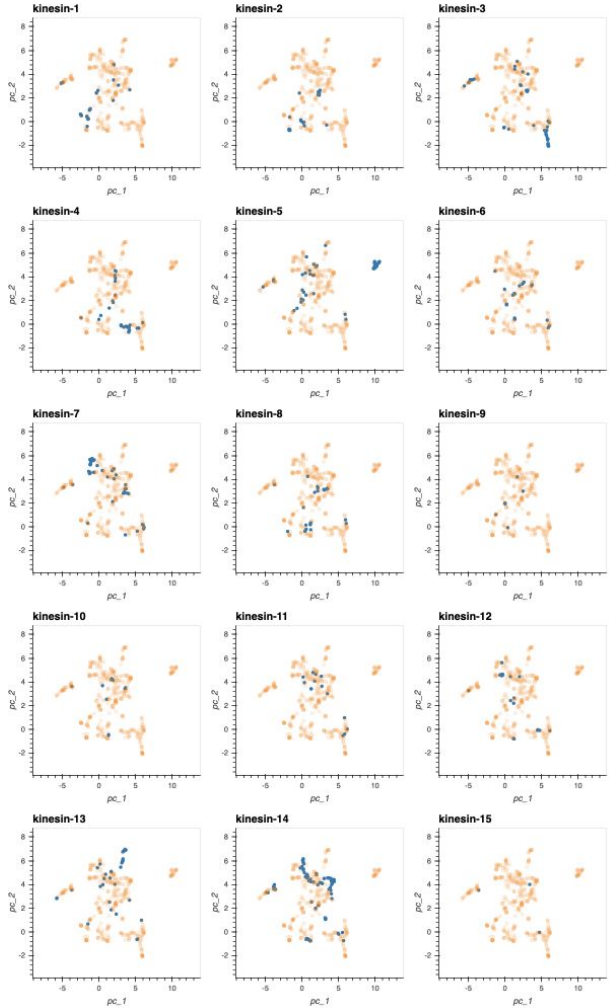
Visualization: Kinesin Family Clustering



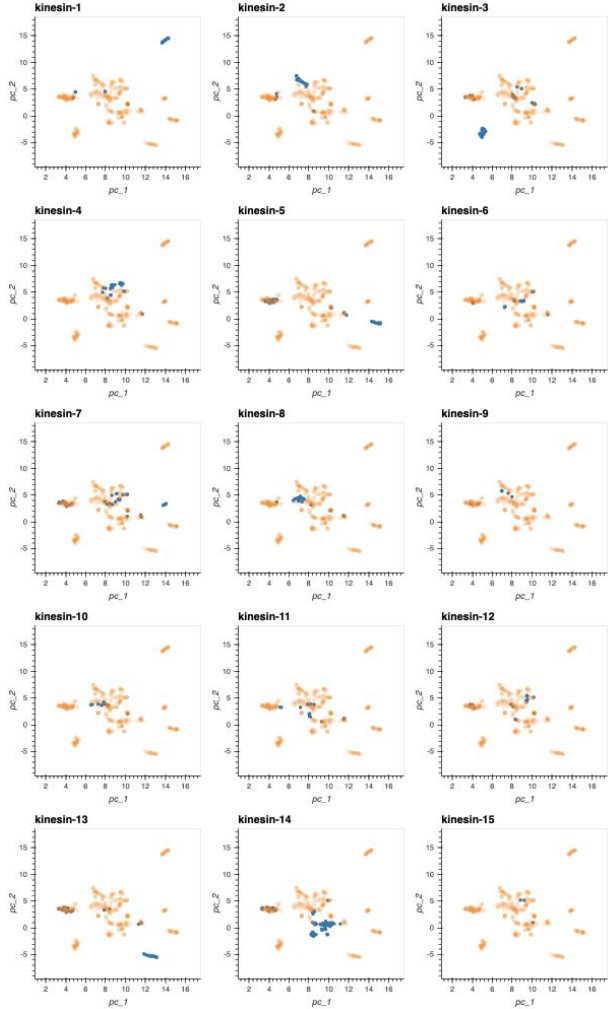
12 Layer ESM transformer



12 Layer ESM transformer + evotune on dynein+kinesin



12 Layer ESM transformer



12 Layer ESM transformer + evotune on dynein+kinesin

Visualization: Kinesin Family Clustering

Visualization: Kinesin Family Clustering

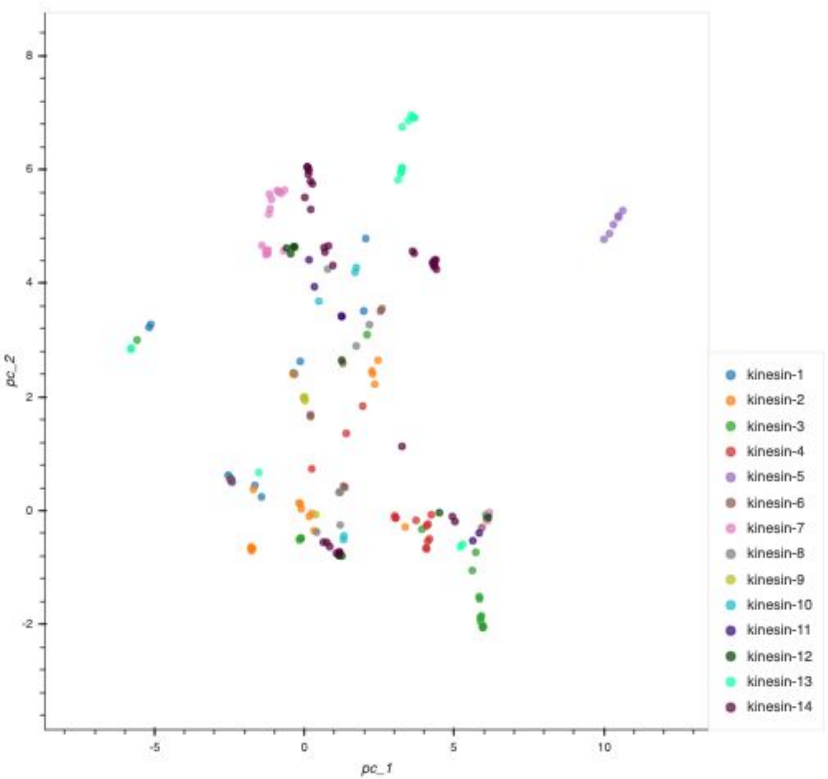
- The evotuned ESM model embedding space characterized the kinesin family clusters better
- With the evotuned model, the clustering is tight and clear for most of the 15 kinesin families

Visualization: Kinesin Family Clustering of Top 4 organisms in the dataset

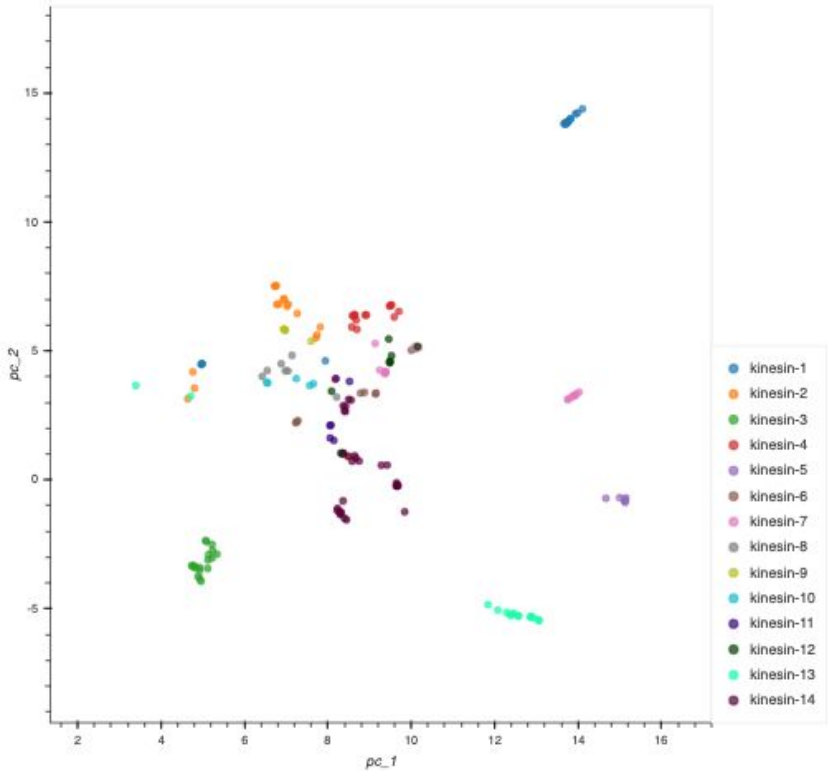
kif_all.loc[kif_all["db_name"] ==
"kif_jp",:].groupby("Organism").count().reset_index().sort_values("seq",ascending = False).iloc[:4,:2]

	Organism	Entry
2	Arabidopsis thaliana (Mouse-ear cress)	59
38	Homo sapiens (Human)	46
49	Mus musculus (Mouse)	42
25	Drosophila melanogaster (Fruit fly)	29

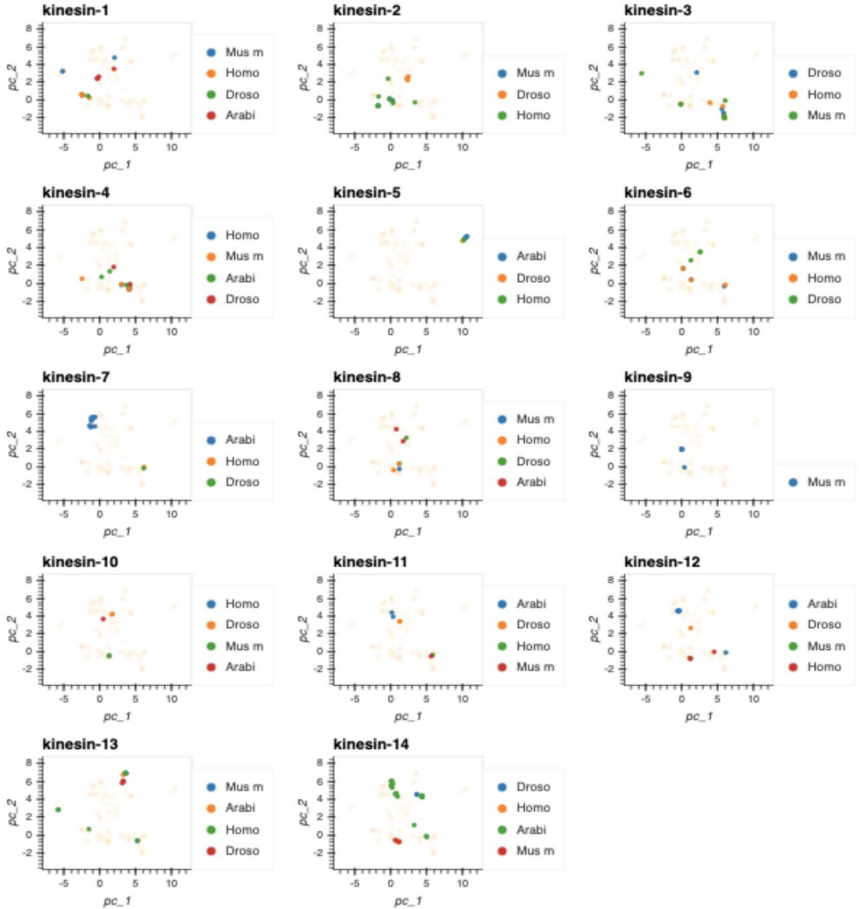
Visualization: Top 4 Organisms color by kinesin family



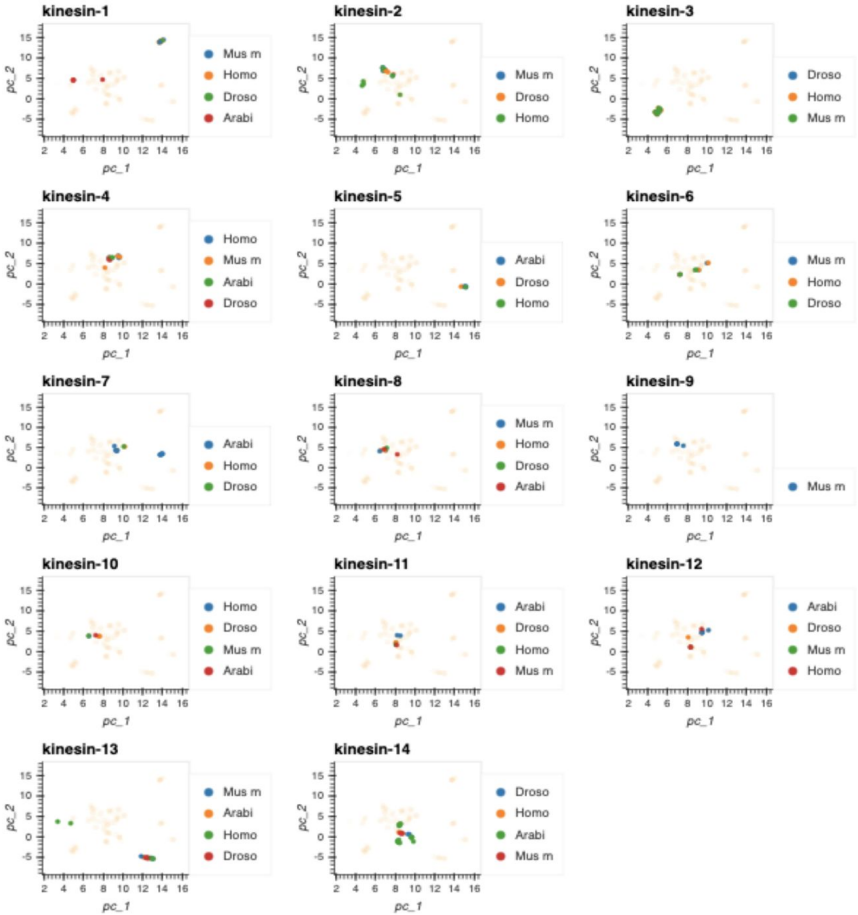
12 Layer ESM transformer



12 Layer ESM transformer + evotune on dynein+kinesin

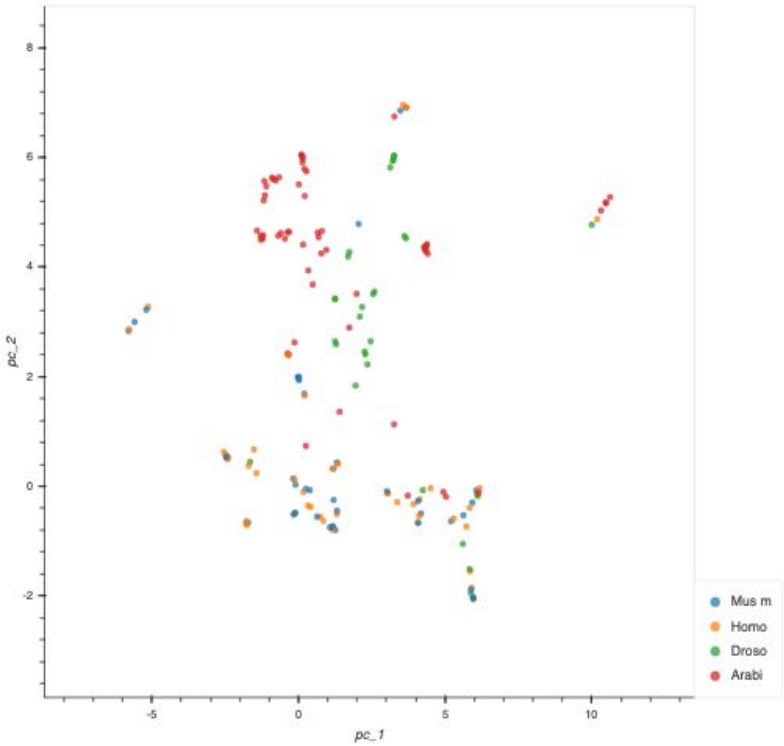


12 Layer ESM transformer

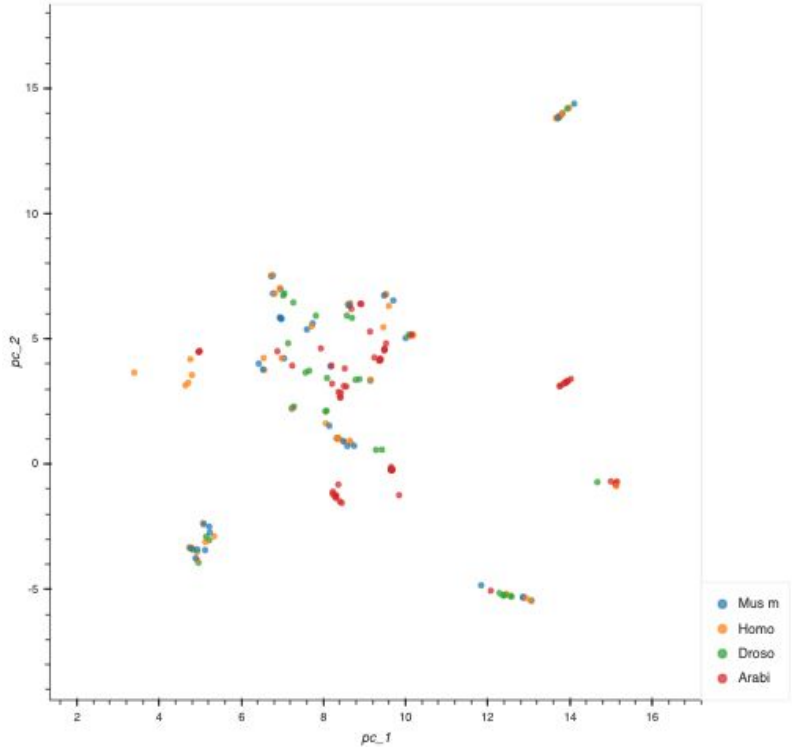


12 Layer ESM transformer + evotune on dynein+kinesin

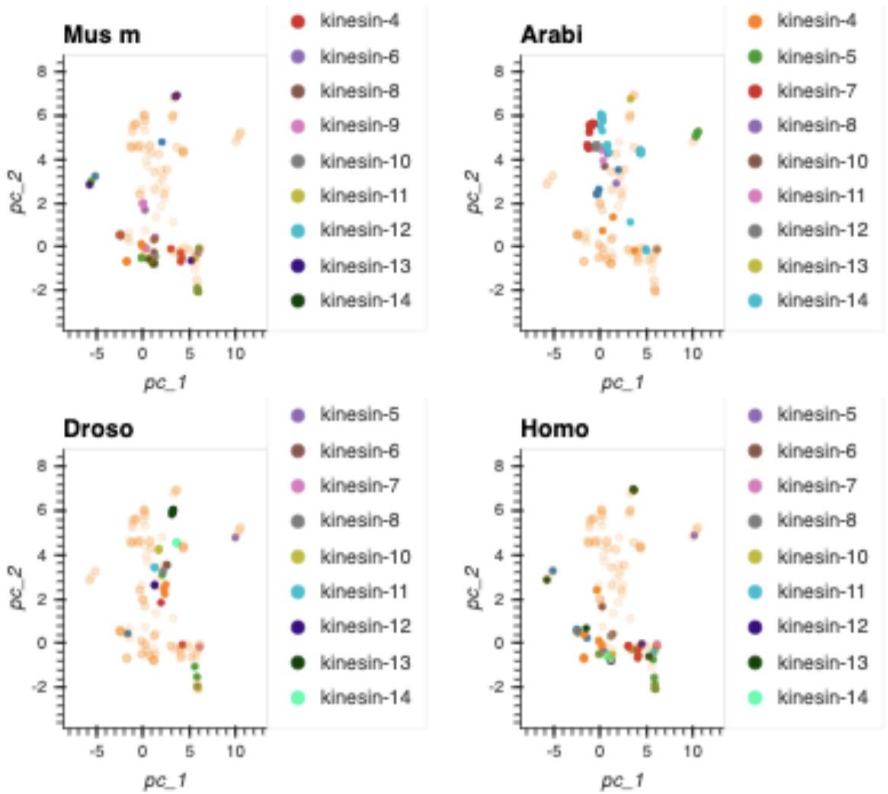
Visualization: Top 4 Organisms color by organism



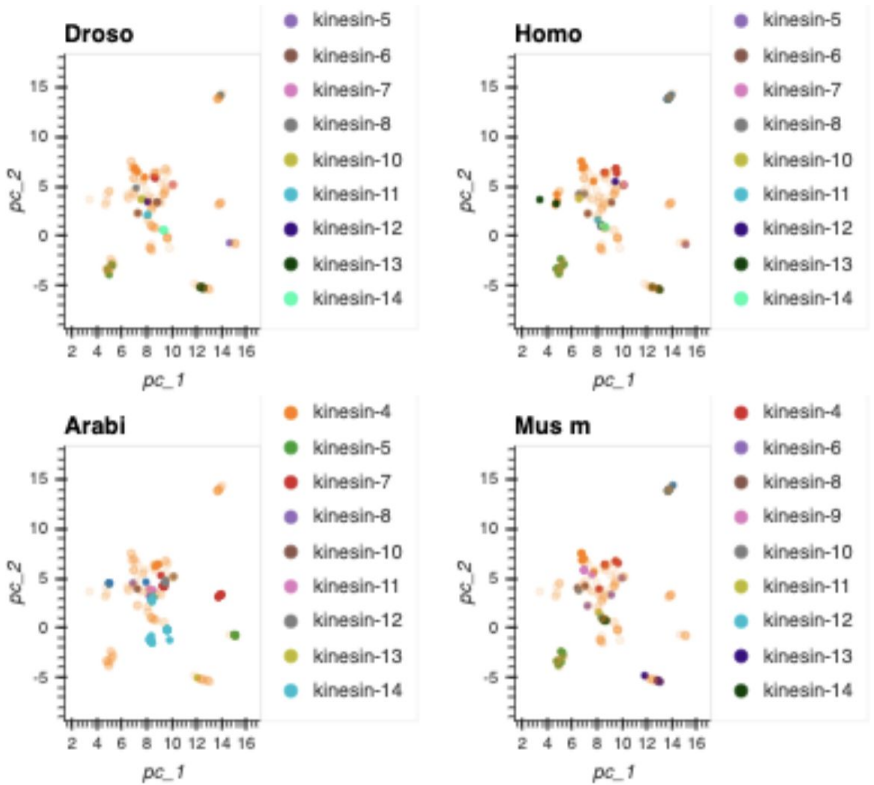
12 Layer ESM transformer



12 Layer ESM transformer + evotune on dynein+kinesin



12 Layer ESM transformer



12 Layer ESM transformer + evotune on dynein+kinesin

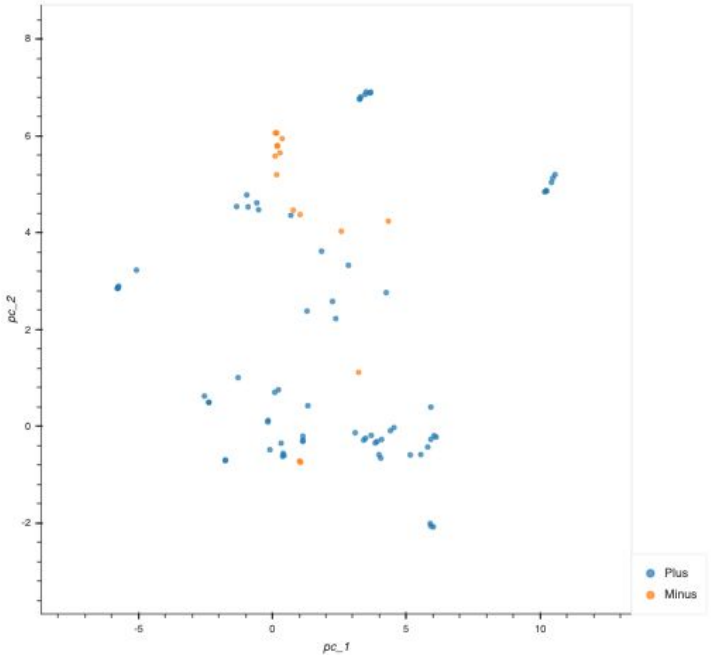
Visualization: Top 4 Organisms color by organism

- From coloring by organism and coloring by kinesin families for the top 4 organisms, we could observe that
- **Before weight updates on dynein+kinesin**, the clustering is grouped more based on the type of organism
- **After the weight updates,** the clustering is grouped more based on the type of kinesin family

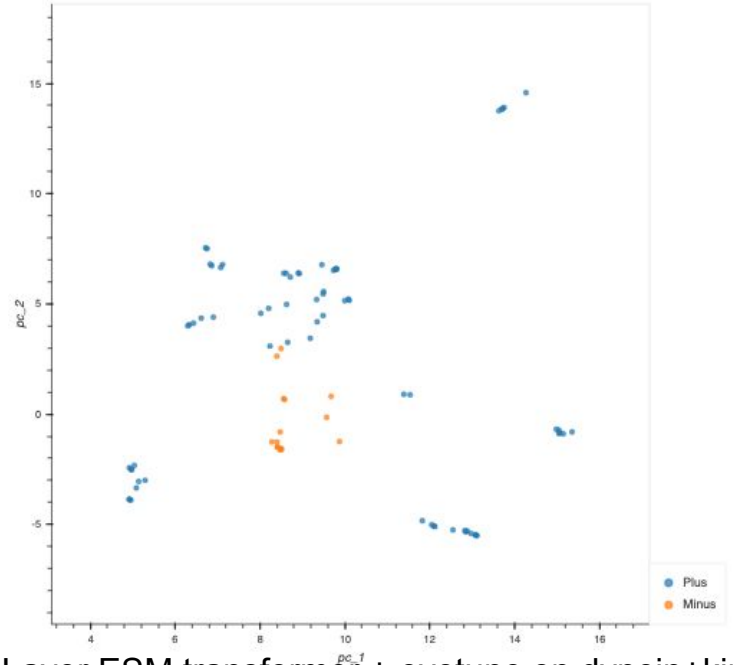
	kif_acc	Molecular mass (kDa)	Motor polarity & velocity	Species/protein	uniprot_acc
0	DmKHC	110	Plus, 54 µm/min	D. melanogaster KHC	P17210;P21613;P35978;P28738;O60282;Q61768;Q2PQ...
1	LpKHC	109	Plus, 30 µm/min	L. pealii KHC	P21613;P17210;O60282;P28738;P33175;Q12840;P359...
2	NcKHC	103	Plus, 120-180 µm/min	N. crassa KHC (NKin)	P48467;Q86Z98;Q86ZC1;O43093;Q9US60;Q54UC9;P331...
3	Dmklp68D	88	Plus, 18 µm/min	D. melanogaster Klp68D	NaN
4	MmKIF3A	80	Plus, 36 µm/min	M. musculus KIF3A	P28741;Q4R628;Q9Y496;Q5R4H3;P46872;P46871;Q617...
5	MmKIF3B	85	Plus, 18 µm/min	M. musculus KIF3B	Q61771;O15066;P46871;Q5R706;O35066;O55165;O147...
6	SpKRP85	79	Plus, 24 µm/min	S. purpuratus KRP85	P46872;Q4R628;Q9Y496;Q5R4H3;P28741;P46871;Q617...
7	SpKRP95	84	Plus, 24 µm/min	S. purpuratus KRP95	P46871;O15066;Q61771;P46872;Q5R706;O14782;O350...
8	MmKIF1A	192	Plus, 72 µm/min	M. musculus KIF1A	P33173;F1M4A4;Q12756;O60333;Q60575;O88658;O438...
9	MmKIF1B	130	Plus, 40 µm/min	M. musculus KIF1B	Q60575;O88658;O60333;Q12756;F1M4A4;P33173;O350...
10	MmKIF4	140	Plus, 2-12 µm/min	M. musculus KIF4	P33174;O95239;Q2VIQ3;Q90640;Q91784;F1M5N7;Q9QX...
11	HsKSP	121	Plus, 2 µm/min	H. sapiens KSP (HsEg5)	P52732;Q6P9P6;B2GU58;P28025;Q91783;P46874;F4II...
12	DmKLP61F	121 (130)	Plus, 2 µm/min	D. melanogaster Klp61F (KRP130)	P46863;P46874;B2GU58;Q91783;Q6P9P6;P52732;P280...
13	XlEg5	119	Plus, 2 µm/min	X. laevis Eg5	P28025;Q91783;B2GU58;Q6P9P6;P52732;P46874;F4II...
14	HsMKLP1	110	Plus, 4 µm/min	H.sapiens MKLP1	Q02241;E9Q5G3;O95235;Q96Q89;Q80WE4;Q7TSP2;Q9V8...
15	MmKIF2	81	Plus, 28 µm/min	M. musculus KIF2	Q5R9Y9;P28740;O00139;Q9WV63;Q2NL05;Q5ZKV8;Q916...
16	DmNcd	78	Minus, 8-15 µm/min	D. melanogaster Ncd	P20480;P28739;Q5XI63;P79955;Q60443;P46875;Q9BP...
17	ScKAR3	84	Minus, 1-2 µm/min	S. cerevisiae Kar3	P17119;Q9US03;P28739;Q92376;Q0J9V3;A3BFT0;P468...
18	CgCHO2	69	Minus, 1-8 µm/min	C. griseus CHO2	Q60443;Q5XI63;Q9QWT9;Q9BW19;P79955;P46875;P468...
19	Xlklp2	159	Plus, 2-4 µm/min	X. laevis Klp2	Q91785;Q498L9;Q9NS87;Q7TSP2;Q6P9L6;Q9GYZ0;B9FU...

Visualization: direction & velocity

- Kinesin-14 motors display a slow velocity towards the microtubule minus-end compared with the plus-end-directed conventional kinesin (42.7±2.19 nm/s)

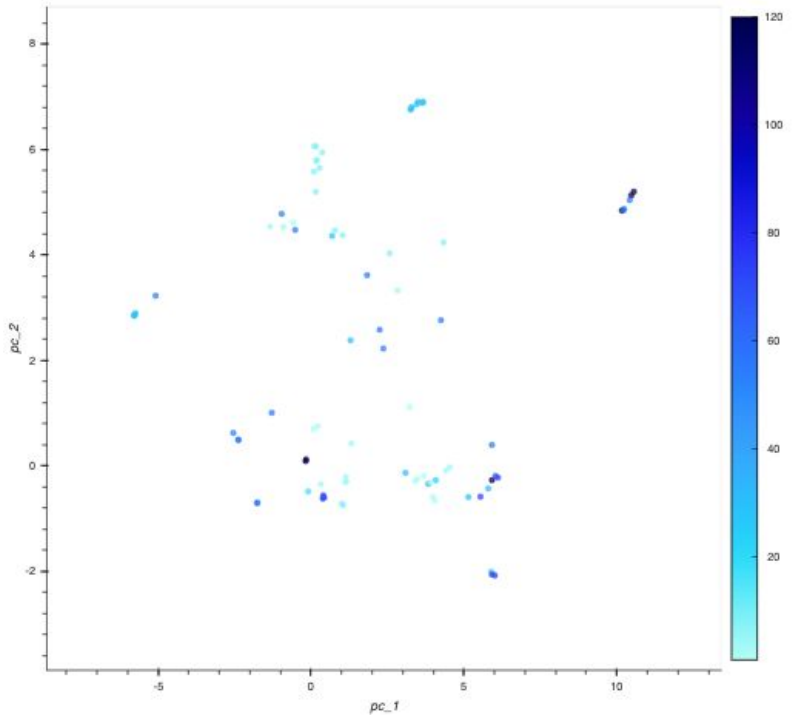


12 Layer ESM transformer

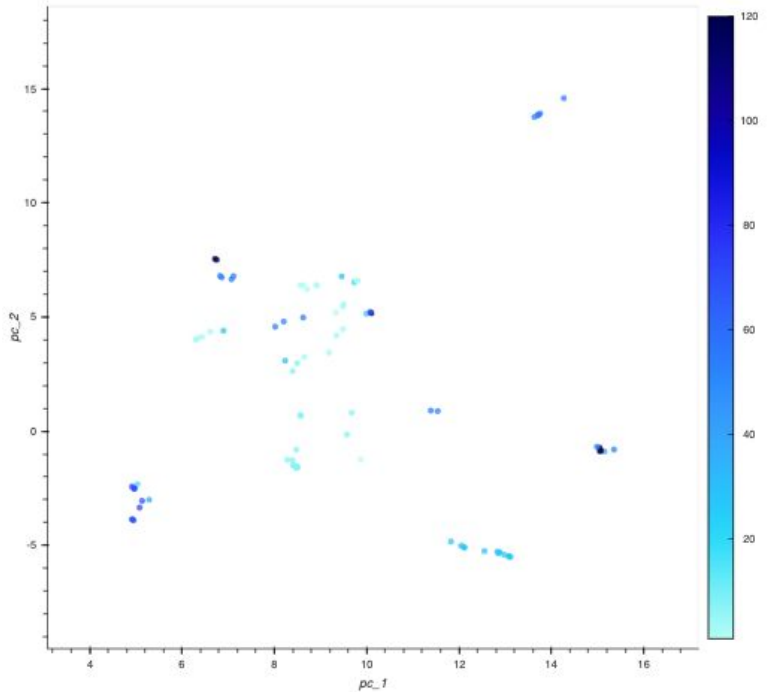


12 Layer ESM transformer + evotune on dynein+kinesin

Visualization: direction & velocity

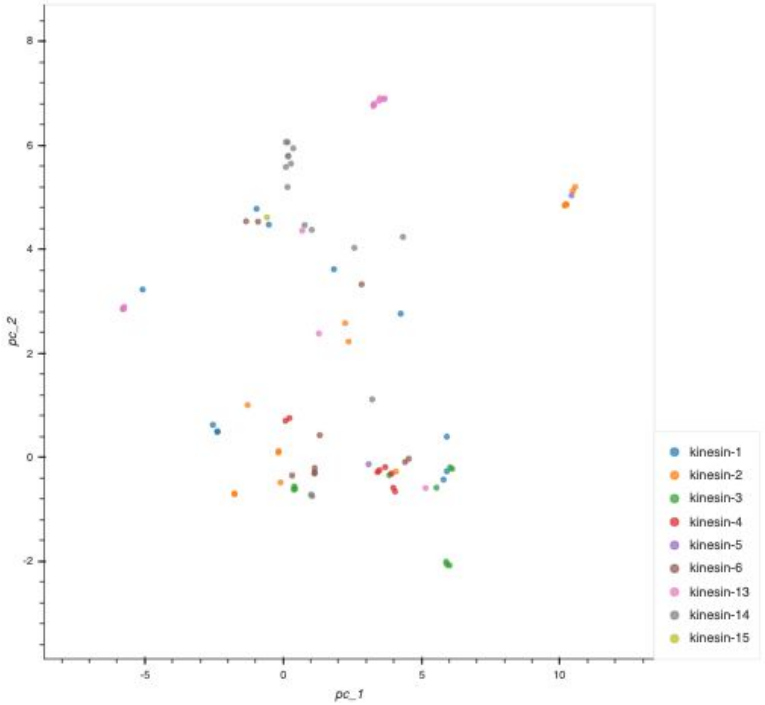


12 Layer ESM transformer

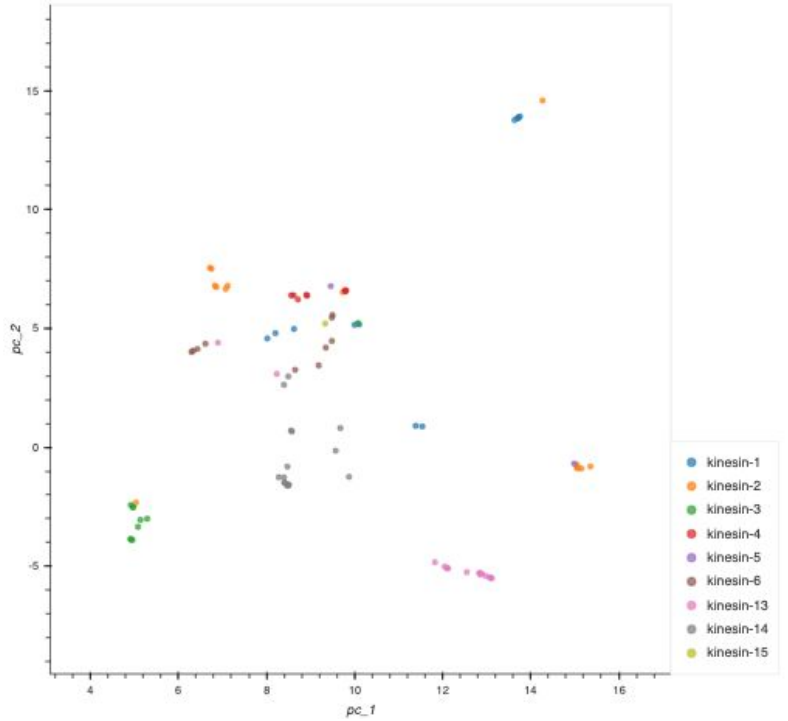


12 Layer ESM transformer + evotune on dynein+kinesin

Visualization: direction & velocity



12 Layer ESM transformer



12 Layer ESM transformer + evotune on dynein+kinesin

Visualization: direction & velocity

- 12 Layer ESM transformer + evotune on dynein+kinesin outperformed in grouping the kinesins with the same directionality and similar speed.
- However, it should be noted that only kinesin family 14 display the minus end directionality, and so the directionality classification essentially reduced to clustering family 14 VS non family 14
- Since there are no full amino acid sequence for velocity data available, the kif_speed dataset was augmented by BLASTp against each of the segment provided by the Duke KIF homepage, and so learning sequence similarity may effectively group together data in this current dataset.
- Better training dataset/ augmentation method is needed. Semi-supervised learning is a potential direction to go with. (with 20 labeled datapoint and other data points unlabeled)