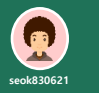


Final Project

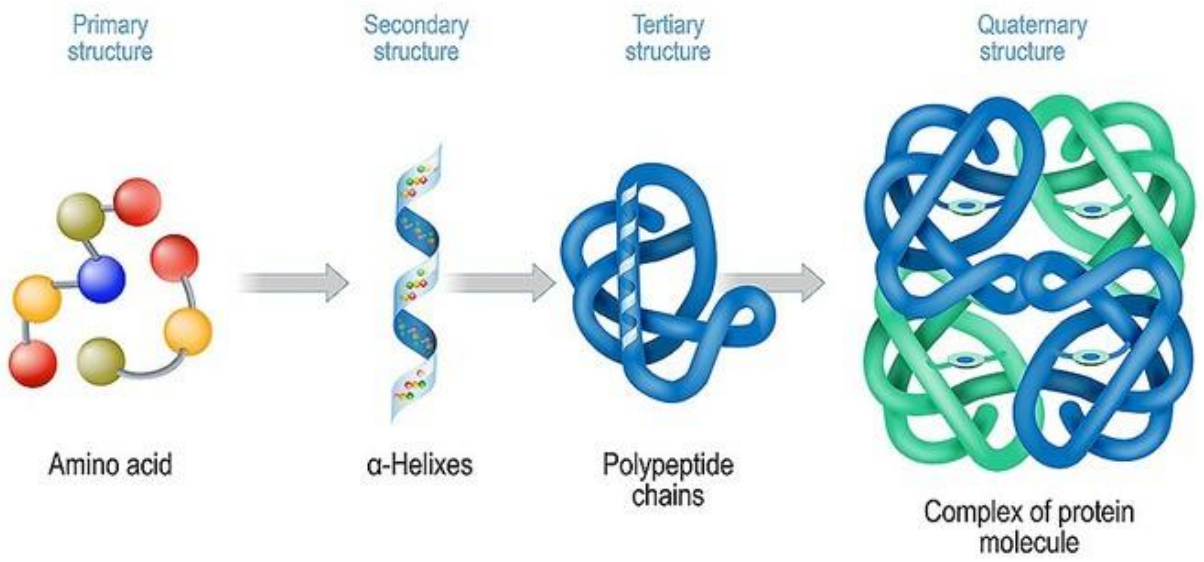
- CAFA 5 Protein Function Prediction

Final Project

- CAFA 5 Protein Function Prediction

Topic

A



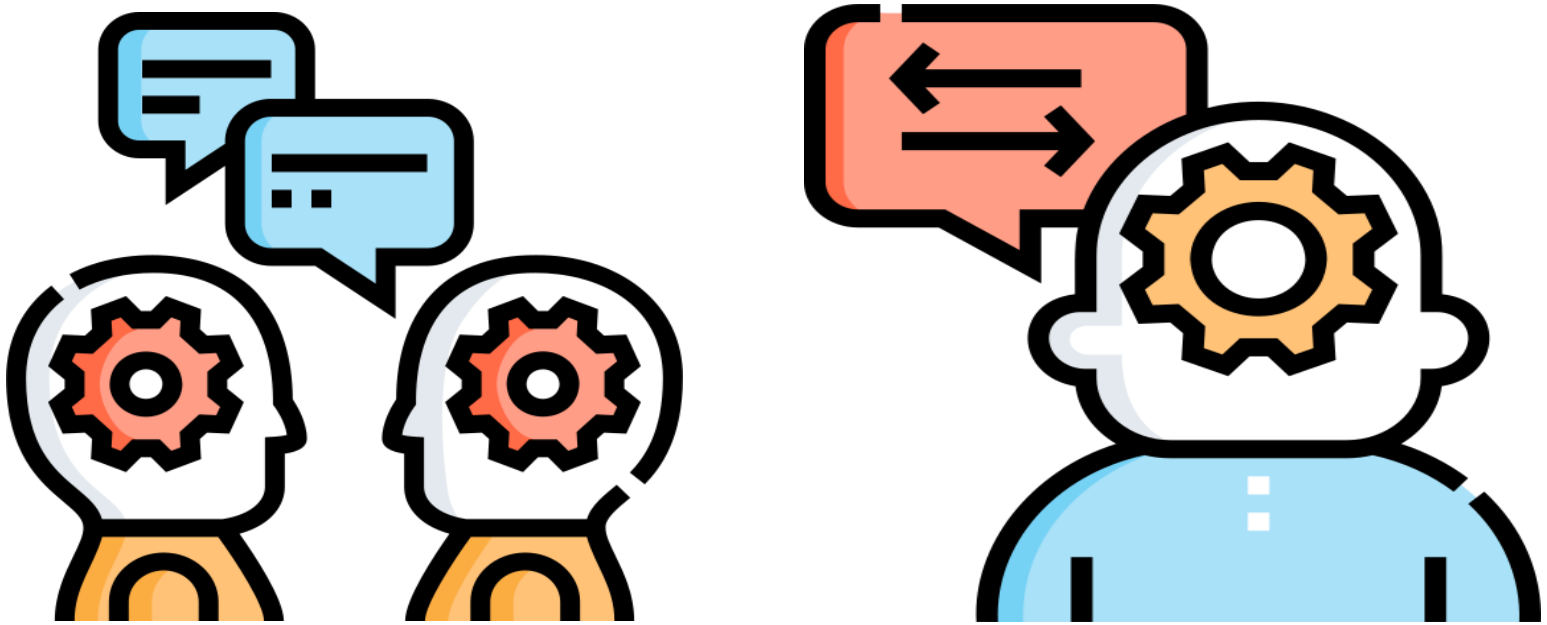
Proteins are extensive biomolecules and macromolecules that consist of one or more lengthy chains of amino acid residues. They carry out an immense array of functions within organisms and primarily vary from one another in terms of their sequence of amino acids. This sequence typically leads to the folding of proteins into a distinct 3D structure, which in turn dictates their activity.

Final Project

- CAFA 5 Protein Function Prediction

Topic

A



Natural language processing (NLP) is an interdisciplinary subfield of computer science and linguistics. Its primary concern lies in providing computers with the capability to comprehend and manipulate speech using machine learning approaches. The objective is to enable a computer to comprehend the contents of documents, encompassing the contextual intricacies of the language used within them.

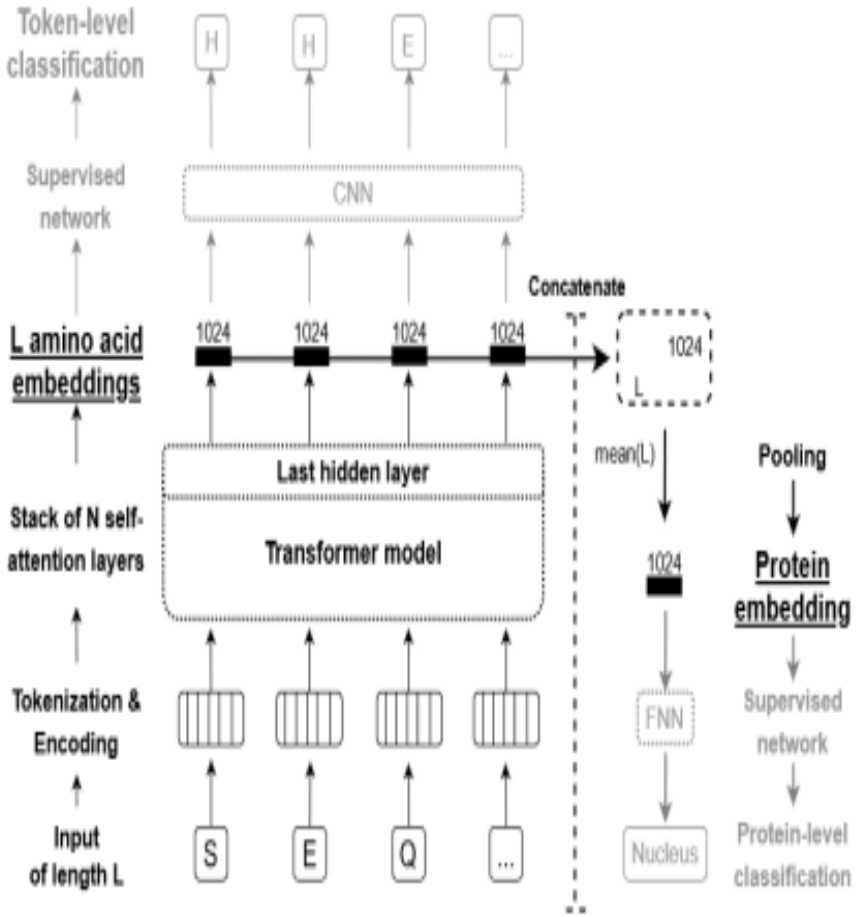
In analogy to NLP, these approaches interpret an entire protein sequence as a sentence and amino acids as single words. Protein sequences are constrained to adopt particular 3D structures optimized for accomplishing particular functions. These constraints mirror the rules of grammar and meaning in NLP.

Final Project

- CAFA 5 Protein Function Prediction

Topic

A



protein sequence is tokenized and positional encoding is added. The resulting vectors are passed through any of our ProtTrans models to create embeddings for each amino acid.

① CNN - > amino acid level embedding
These embeddings can be directly employed as inputs for prediction tasks at the level of individual tokens to predict the secondary structure of an amino acid.

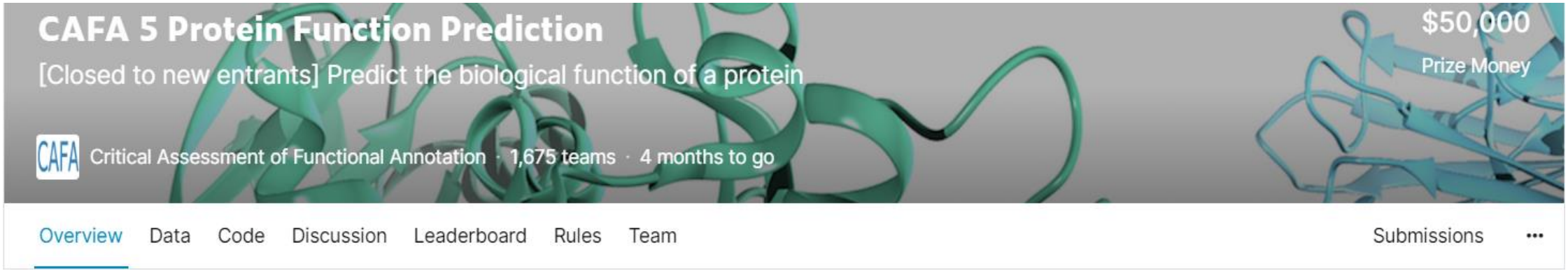
②FNN -> protein level embedding
those embeddings can be concatenated and pooled along the length-dimension to get fixed-size embedding .The resulting protein-level embedding can be used as input for predicting aspects of proteins

Final Project

- CAFA 5 Protein Function Prediction

Topic

A



Purpose

① Through this competition, we attempt to utilize natural language processing models to predict the functions of proteins.

② We aim to enhance the predictive accuracy by adjusting hyperparameters or fine-tuning the stages of the artificial neural network.

Final Project

- CAFA 5 Protein Function Prediction

Data

B

1. Protein Function Information

```
[Term]
id: GO:0000001
name: mitochondrion inheritance
namespace: biological_process
def: "The distribution of mitochondria, including the mitochondrial genome, into daughter cells after mito
synonym: "mitochondrial inheritance" EXACT []
is_a: GO:0048308 ! organelle inheritance
is_a: GO:0048311 ! mitochondrion distribution

[Term]
id: GO:0000002
name: mitochondrial genome maintenance
namespace: biological_process
def: "The maintenance of the structure and integrity of the mitochondrial genome; includes replication and
is_a: GO:0007005 ! mitochondrion organization
```

Final Project

- CAFA 5 Protein Function Prediction

Data

B

2. Protein Information – train & test dataset

```
>O73864 sp|O73864|WNT11_DANRE Protein Wnt-11 OS=Danio rerio OX=7955 GN=wnt11 PE=2 SV=1
MTEYRNFLLLFITSLSVIYPCTGISWLGLTINGSSVGWNQTHHCKLLDGLVPDQQQLCKR
NLELMHSIVRAARLTKSACTSSFSDMRWNWSSIESAPHFTPDLAKGTREAAFVVSLAAAV
VSHAIARACASGDLPSCSCAAMPSEQAAPDFRWGGCGDNLRYYGLQMGSAFSDAPMRNRR
SGPQDFRLMQLHNNAVGRQVLMDSLEMKCKCHGVSGSCSVKTCWKGLQDISTISADLKSK
YLSATKVIPRQIGTRRQLVPREMEVRPVGENELVYLVSSPDYCTQNAKQGSLGTTDRQCN
KTASGSESCGLMCCGRGYNAYTEVLVERCQCKYHWCCYVSCKTCKRTVERYVSK
>O95231 sp|O95231|VENTX_HUMAN Homeobox protein VENTX OS=Homo sapiens OX=9606 GN=VENTX PE=1 SV=1
MRLSSSPPRGPQQLSSFGSVDWLSQSSCSGPTHTPRPADFSLGSLPGPGQTSGAREPPQA
VSIKEAAGSSNLPAPERTMAGLSKEPNTLRAPRVRTAFTMEQVRTLEGVFQHHQYLSPLE
RKRLAREMQLSEVQIKTWFQNRRMKHKRQMQDPQLHSPFSGSLHAPPAFYSTSSGLANGL
QLLCPWAPLSGPQALMLPPGSFWGLCQVAQEALASAGASCCGQPLASHPPTPGRPSLGPA
LSTGPRGLCAMPQTGDAF
```

Protein Name

Organism Species

Sequence Version

Protein Existence

Organism Taxonomy ID

Gene Name

Protein Sequence

Final Project

- CAFA 5 Protein Function Prediction

Data

B

3. Target Information

Protein Function ID

Protein ID

Protein Aspect

EntryID	term	aspect
5363863 unique values	[null] 100%	[null] 100%
A0A009IHW8	GO:0008152	BPO
A0A009IHW8	GO:0034655	BPO
A0A009IHW8	GO:0072523	BPO
A0A009IHW8	GO:0044270	BPO
A0A009IHW8	GO:0006753	BPO
A0A009IHW8	GO:1901292	BPO
A0A009IHW8	GO:0044237	BPO
A0A009IHW8	GO:1901360	BPO
A0A009IHW8	GO:0008150	BPO
A0A009IHW8	GO:1901564	BPO
A0A009IHW8	GO:1901565	BPO
A0A009IHW8	GO:0009117	BPO
A0A009IHW8	GO:0006139	BPO
A0A009IHW8	GO:0044281	BPO

Final Project

- CAFA 5 Protein Function Prediction

Pre-trained Model

C

Types of models

①T5 ②Electra ③BERT ④Albert ⑤Transformer-XL ⑥XLNet

The reason for model selection

① Because T5 model include the encoder and the decoder, T5 provides the flexibility to apply different training methodologies and masking strategies.

② T5 allows the reconstruction of spans of tokens instead of individual tokens.

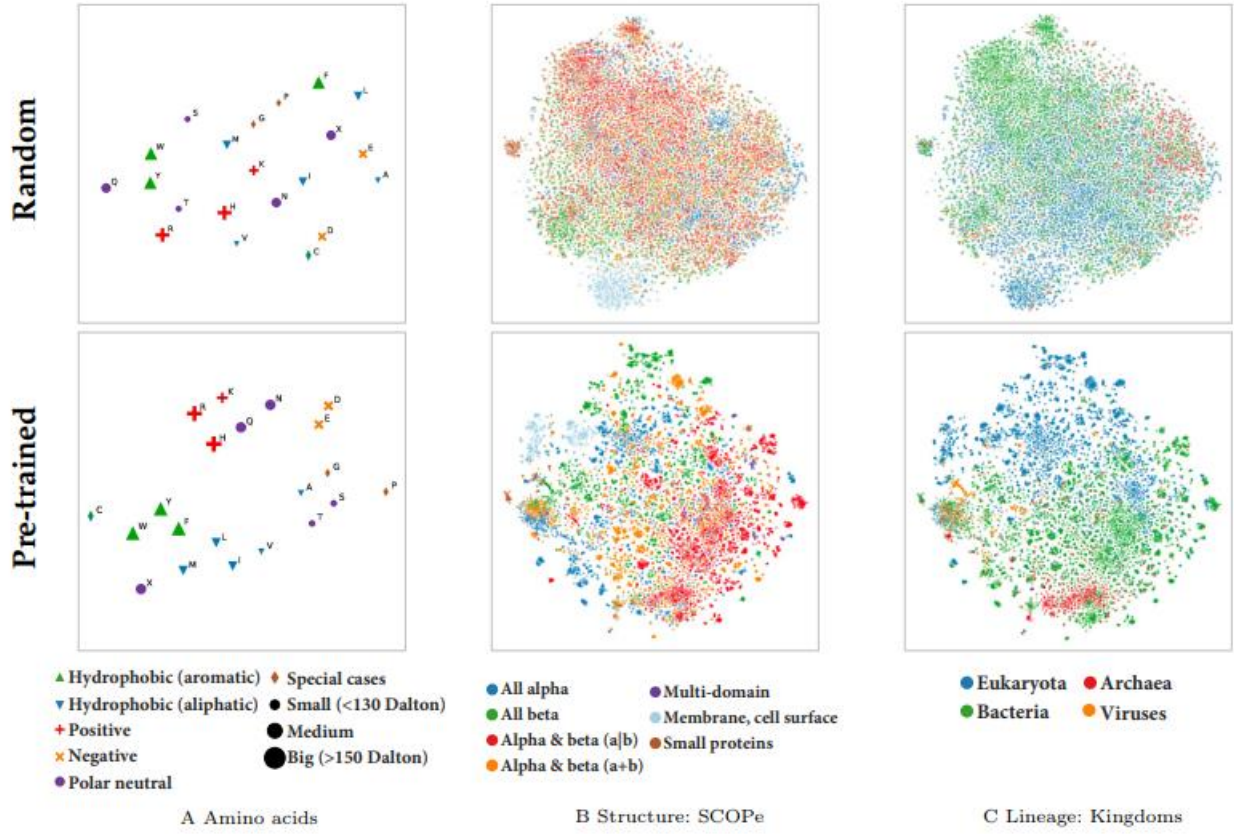
③ T5 learns a shared positional encoding for each attention head across all layers. Through this mechanism, the model learns to integrate the relative offsets between residue pairs from lower layers. As a result, the model becomes capable of making predictions that extend beyond the actual length of the positional encoding

Final Project

- CAFA 5 Protein Function Prediction

Pre-trained Model

C



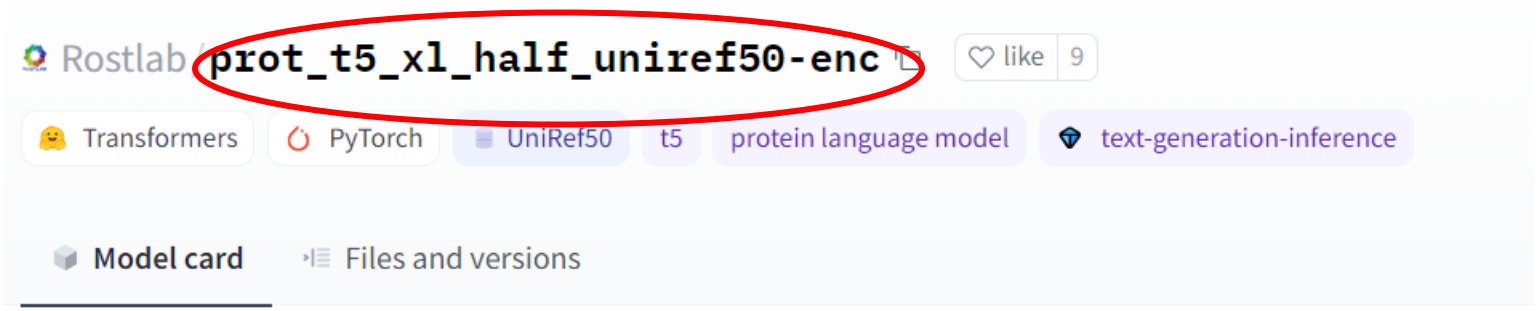
All three categories, (A), (B), and (C), have been more accurately classified, showcasing impressive results when we used the pre-trained models.

Final Project

- CAFA 5 Protein Function Prediction

Pre-trained Model

C



Encoder only ProtT5-XL-UniRef50, half-precision model

An encoder-only, half-precision version of the ProtT5-XL-UniRef50 model. The original model and it's pretraining were introduced in this paper and first released in this repository. This model is trained on uppercase amino acids: it only works with capital letter amino acids.

Model description

ProtT5-XL-UniRef50 is based on the `t5-3b` model and was pretrained on a large corpus of protein sequences in a self-supervised fashion. This means it was pretrained on the raw protein sequences only, with no humans labelling them in any way (which is why it can use lots of publicly available data) with an automatic process to generate inputs and labels from those protein sequences.

Final Project

- CAFA 5 Protein Function Prediction

Data Pre-processing

D

```
import os
import json
from typing import Dict
from collections import Counter

import random
import obonet
import pandas as pd
import numpy as np
from Bio import SeqIO

from transformers import T5Tokenizer, T5EncoderModel
import torch
device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')

# Load the tokenizer
tokenizer = T5Tokenizer.from_pretrained('Rostlab/prot_t5_xl_half_uniref50-enc', do_lower_case=False) #.to(device)

# Load the model
model = T5EncoderModel.from_pretrained("Rostlab/prot_t5_xl_half_uniref50-enc").to(device);
```

Final Project

- CAFA 5 Protein Function Prediction

Data Pre-processing

D

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
for dirname, _, filenames in os.walk('/kaggle/input/cafa-5-protein-function-prediction'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
from pathlib import Path
path = Path('/kaggle/input/cafa-5-protein-function-prediction')
```

Final Project

- CAFA 5 Protein Function Prediction

Data Pre-processing

D

```
import re
def get_embeddings(seq):
    sequence_examples = [" ".join(list(re.sub(r"[UZOB]", "X", seq)))]

    ids = tokenizer.batch_encode_plus(sequence_examples, add_special_tokens=True, padding="longest")

    input_ids = torch.tensor(ids['input_ids']).to(device)
    attention_mask = torch.tensor(ids['attention_mask']).to(device)

    # generate embeddings
    with torch.no_grad():
        embedding_repr = model(input_ids=input_ids,
                               attention_mask=attention_mask)

    # extract residue embeddings for the first ([0,:]) sequence in the batch and remove padded & special tokens ([0,:7])
    emb_0 = embedding_repr.last_hidden_state[0]
    emb_0_per_protein = emb_0.mean(dim=0)

    return emb_0_per_protein
```

Final Project

- CAFA 5 Protein Function Prediction

Data Pre-processing

D

```
fn = '/kaggle/input/cafa-5-protein-function-prediction/Train/train_sequences.fasta'
print("Sequence example:\n\n", next(iter(SeqIO.parse(fn, "fasta"))))
sequences = SeqIO.parse(fn, "fasta")
num_sequences = sum(1 for seq in sequences)
print()
print("Number of sequences in train:", num_sequences)
```

```
import tqdm
fn = '/kaggle/input/cafa-5-protein-function-prediction/Train/train_sequences.fasta'

sequences = SeqIO.parse(fn, "fasta")

ids = []
embeds = np.zeros((num_sequences, 1024))
i = 0
for seq in tqdm.tqdm(sequences):
    ids.append(seq.id)
    embeds[i] = get_embeddings(str(seq.seq)).detach().cpu().numpy()
    i += 1
    break #remove it for full calculation

np.save('train_embeds.npy', embeds)
np.save('train_ids.npy', np.array(ids))
```

Final Project

- CAFA 5 Protein Function Prediction

Data Pre-processing

D

```
fn = '/kaggle/input/cafa-5-protein-function-prediction/Test (Targets)/testsuperset.fasta'

sequences = SeqIO.parse(fn, "fasta")
num_sequences = sum(1 for seq in sequences)
print("Number of sequences in test:", num_sequences)
sequences = SeqIO.parse(fn, "fasta")


ids = []
embeds = np.zeros((num_sequences, 1024))
i = 0
for seq in tqdm.tqdm(sequences):
    ids.append(seq.id)
    embeds[i] = get_embeddings(str(seq.seq)).detach().cpu().numpy()
    i += 1
    break #remove it for full calculation

np.save('test_embeds.npy', embeds)
np.save('test_ids.npy', np.array(ids))
```

Final Project

- CAFA 5 Protein Function Prediction

Model Training

E

```
import pandas as pd
import numpy as np

import random
import os
from scipy import sparse

import tensorflow as tf
from tensorflow.keras.layers import Dense, BatchNormalization, Dropout, LeakyReLU
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import AUC
from tensorflow.keras.callbacks import LearningRateScheduler, ReduceLROnPlateau, EarlyStopping
```

```
def seed_everything(seed):
    random.seed(seed)
    os.environ['PYTHONHASHSEED'] = str(seed)
    np.random.seed(seed)
    torch.manual_seed(seed)
    torch.cuda.manual_seed(seed)
    torch.backends.cudnn.deterministic = True
    torch.backends.cudnn.benchmark = True

seed_everything(42)
```

Final Project

- CAFA 5 Protein Function Prediction

Model Training

E

```
# train data - creation of embeddings
train_terms = pd.read_csv('/content/data/Train/train_terms.tsv',sep = '₩t')
train_protein_ids = np.load('/content/data/train_ids.npy')
train_embeddings = np.load('/content/data/train_embeds.npy')

column_num = train_embeddings.shape[1]
train_df = pd.DataFrame(train_embeddings, columns = ["Column_" + str(i) for i in range(1, column_num+1)])
train_df
```

	Column_1	Column_2	Column_3	Column_4	Column_5	Column_6	Column_7	Column_8	Column_9	Column_10	...
0	0.049488	-0.032935	0.032473	-0.033723	-0.059505	0.075936	-0.045860	-0.104476	-0.072112	0.038806	...
1	-0.044616	0.064925	-0.080263	-0.075338	-0.004731	0.025408	-0.024685	-0.016568	-0.038980	-0.033870	...
2	-0.020128	-0.049779	0.007894	-0.000829	-0.047737	0.086453	-0.038107	-0.036379	0.029611	0.045029	...
3	-0.007515	0.060628	0.004045	0.027056	-0.021542	0.010380	-0.025064	-0.055834	0.068238	0.027764	...
4	0.013468	0.041516	0.018435	-0.035595	0.008770	0.018699	-0.015452	-0.038092	-0.038326	-0.012299	...
...	...	...	...	...	...	...	...	...	...	...	...
142241	0.032529	0.032490	0.026844	0.007486	-0.019346	0.033527	0.048294	-0.091146	0.007092	0.008843	...
142242	0.056026	0.035470	0.021987	0.028443	-0.001087	0.020100	-0.010549	-0.049460	0.027347	-0.028113	...
142243	0.016918	0.041331	0.000793	-0.005990	-0.039993	0.056007	-0.013786	-0.076081	0.006320	-0.002054	...
142244	0.061252	0.083402	0.044025	0.047916	0.024477	0.035183	-0.016680	-0.043163	-0.045596	-0.029994	...
142245	0.021600	0.065170	0.074923	-0.017102	0.020362	0.015444	-0.046874	-0.137347	0.036991	-0.015204	...

142246 rows × 1024 columns

Final Project

- CAFA 5 Protein Function Prediction

Model Training

E

```
# train data - creation of targets
num_of_labels = 1200

labels = train_terms['term'].value_counts().index[:num_of_labels].tolist()
train_terms_updated = train_terms.loc[train_terms['term'].isin(labels)]

train_size = train_protein_ids.shape[0]
train_labels = sparse.lil_matrix((train_size, num_of_labels))
series_train_protein_ids = pd.Series(train_protein_ids)

for i, label in enumerate(labels):
    label_related_proteins = train_terms_updated.loc[train_terms_updated['term'] == label, 'EntryID'].unique()
    train_labels[:, i] = series_train_protein_ids.isin(label_related_proteins).astype(float)

labels_df = pd.DataFrame.sparse.from_spmatrix(train_labels, columns=labels)
print(labels_df.shape)

labels_df.head()
```

	GO:0005575	GO:0008150	GO:0110165	GO:0003674	GO:0005622	GO:0009987	GO:0043226	GO:0043229	GO:0005488	GO:0043227	...
0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	...
1	1.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	...
2	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	...
3	1.0	1.0	1.0	1.0	0.0	1.0	1.0	0.0	0.0	1.0	...
4	1.0	0.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	...

5 rows × 1000 columns

Final Project

- CAFA 5 Protein Function Prediction

Model Training

E

```
# model definition
model = Sequential([
    BatchNormalization(input_shape=[train_df.shape[1]]),
    # batch normalization for bias removal, enhanced generalization,and improved training stability
    Dense(units=1024, kernel_initializer='he_normal'),
    # kernel initializer for role of weight initialization in neural networks and harmony with activation functions(ReLU or LeakyRelu)
    LeakyReLU(alpha=0.1),
    # activation functions for mitigation of the vanishing gradient problem and prevention of dead neurons
    Dropout(0.5),
    # dropout for prevention of overfitting and ensemble effect
    Dense(units=512, kernel_initializer='he_normal'),
    LeakyReLU(alpha=0.1),
    Dropout(0.5),
    Dense(units=256, kernel_initializer='he_normal'),
    LeakyReLU(alpha=0.1),
    Dropout(0.4),
    Dense(units=labels_df.shape[1], activation='sigmoid')
])
# model compiling
model.compile(
    optimizer=Adam(learning_rate=0.01),
    loss='binary_crossentropy',
    metrics=['binary_accuracy', AUC()]
)
# learning rate scheduling
def lr_schedule(epoch, lr):
    if epoch < 50:
        return lr
    else:
        return lr * tf.math.exp(-0.1)

lr_scheduler = LearningRateScheduler(lr_schedule)
lr_reducer = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5, min_lr=1e-6)
# model training
history = model.fit(
    train_df, labels_df,
    batch_size=512,
    epochs=200,
    callbacks=[lr_scheduler, lr_reducer, early_stopping]
)
# model saving
model.save('MLP_model.h5')
```

Final Project

- CAFA 5 Protein Function Prediction

Model Prediction & Submission

F

```
# test data - creation of embeddings
test_embeddings = np.load('/kaggle/input/t5embeds/test_embeds.npy')
test_id = pd.DataFrame(np.load('/kaggle/input/t5embeds/test_ids.npy'))

labels = train_terms['term'].value_counts().index[:1200].tolist()

column_num = test_embeddings.shape[1]
test_df = pd.DataFrame(test_embeddings, columns = ["Column_" + str(i) for i in range(1, column_num+1)])
test_df
```

	Column_1	Column_2	Column_3	Column_4	Column_5	Column_6	Column_7	Column_8	Column_9	Column_10	...
0	0.054705	0.063420	-0.015320	-0.016506	0.042195	0.021592	-0.118535	-0.063298	-0.046146	-0.102311	...
1	0.090373	0.089842	-0.023887	-0.011446	0.051465	0.020982	-0.110989	-0.066646	-0.041259	-0.087551	...
2	0.043588	0.039572	-0.014332	-0.011769	0.045109	0.015847	-0.103339	-0.047735	-0.022730	-0.091452	...
3	0.055668	0.049560	-0.019646	-0.006977	0.039897	0.021177	-0.108079	-0.047191	-0.031517	-0.100057	...
4	0.022637	0.014306	-0.002696	-0.034456	0.034854	0.020822	-0.114046	-0.050019	-0.026491	-0.097928	...
...	...	...	...	...	...	...	...	...	...	...	...
141860	0.081780	0.077098	0.006294	-0.073283	-0.064078	0.011976	0.018399	-0.102211	0.078021	0.018553	...
141861	0.041950	0.159886	0.033731	-0.087298	-0.002340	-0.015892	0.034585	-0.163370	-0.030956	-0.047006	...
141862	0.032748	0.141868	0.034142	-0.063111	-0.006216	-0.025474	0.036954	-0.156333	-0.021958	-0.040103	...
141863	0.052713	0.157013	0.043279	-0.065428	-0.010776	-0.034223	0.042106	-0.156888	-0.020948	-0.047612	...
141864	0.041877	0.134041	0.079094	-0.056902	0.008855	-0.055976	0.019338	-0.234666	-0.029368	-0.073774	...

141865 rows × 1024 columns

Final Project

- CAFA 5 Protein Function Prediction

Model Prediction & Submission

F

```
# Prediction of protein fuction probability
model = tf.keras.models.load_model('/kaggle/input/mlp-model11-reduced-test-size/MLP_model (12).h5')
predictions =  model.predict(test_df)
predictions = pd.DataFrame(predictions)
test_id.columns = ['Protein_ID']
pred_df = pd.concat([test_id, predictions], axis=1)
pred_df.set_index('Protein_ID', inplace=True)
pred_df.columns = labels
pred_df
```

Protein_ID	GO:0005575	GO:0008150	GO:0110165	GO:0003674	GO:0005622	GO:0009987	GO:0043226	GO:0043229	GO:0005488	GO:0043227	...
Q9CQV8	0.692375	0.702570	0.684302	0.659737	0.607045	0.461994	0.478053	0.455063	0.656592	0.413393	...
P62259	0.745857	0.725946	0.738817	0.693453	0.658088	0.535945	0.551942	0.526895	0.684225	0.473731	...
P68510	0.719347	0.734163	0.711474	0.670764	0.638079	0.544530	0.506015	0.485718	0.647047	0.411370	...
P61982	0.725249	0.744441	0.716388	0.675466	0.642500	0.551664	0.507917	0.487224	0.656580	0.407568	...
O70456	0.668797	0.703553	0.659741	0.644751	0.604590	0.480043	0.466734	0.449062	0.622589	0.385083	...
...	...	...	...	...	...	...	...	...	...	...	...
P08380	0.502896	0.481531	0.468847	0.351530	0.016422	0.033306	0.005889	0.004185	0.217432	0.005309	...
C0HK72	0.348229	0.601645	0.324898	0.443901	0.016219	0.071840	0.010584	0.007053	0.250859	0.007997	...
C0HK73	0.314438	0.658558	0.291926	0.388326	0.012364	0.087367	0.009310	0.006032	0.217101	0.006873	...
C0HK74	0.349512	0.659993	0.327774	0.448196	0.020260	0.099928	0.014084	0.009386	0.279324	0.010895	...
A0A3G2FQK2	0.459428	0.660059	0.459712	0.371048	0.073358	0.136053	0.064579	0.052323	0.248176	0.059886	...

141865 rows × 1200 columns

Final Project

- CAFA 5 Protein Function Prediction

Model Prediction & Submission

F

```
# Submission
temp_list = ['Label1', 'Label2', 'Label3']
df_submission = pred_df.stack().reset_index()
df_submission.columns = ['Protein Id','GO Term Id','Prediction']

df_submission.to_csv("submission.tsv",header=False, index=False, sep="₩t")
```

Submission and Description	Public Score
project3- submission(MLP) - Version 15 Succeeded · Innbum Baek · 9d ago	0.49807

Evaluation Metrics

-> Weighted F- Score : an evaluation metric used in multi-class classification problems
For each class, the F1 score is calculated using the following formula: F1 score = 2 * (precision * recall) / (precision + recall)
The weighted F-score takes into accounts the class distribution or importance and computes an average F1 score while applying class-specific weights.

Final Project

- CAFA 5 Protein Function Prediction

Improvements

G

① Creating a model that predicts which aspect among the three protein aspects it will have, and developing models based on protein aspects, appears to lead to more accurate predictions.

② Due to insufficient RAM usage in the virtual notebook environment, predictions were conducted using the top 1200 functions possessed by the largest number of proteins. However, in this competition, the objective is to predict 1500 proteins. Expanding the prediction to cover 1500 proteins could potentially yield better predictive results.

③ Using PCA or autoencoders to reduce the dimensions of the numerical embeddings, which consist of 1024 values for each protein, could have resulted in better outcomes by minimizing unnecessary components and addressing multicollinearity.

References

H

① Elnaggar, A., Heinzinger, M., Dallago, C., Rehawi, G., Wang, Y., Jones, L., ... & Rost, B. (2020). ProtTrans: Towards Cracking the Language of Life's Code Through Self-Supervised Learning. preprint. *Bioinformatics, July*.

② Sergeev, A., & Del Balso, M. (2018). Horovod: fast and easy distributed deep learning in TensorFlow. *arXiv preprint arXiv:1802.05799*.

Final Project

- CAFA 5 Protein Function Prediction

Thank you