

 Open access • Posted Content • DOI:10.1101/2020.01.31.929604

Annotating Gene Ontology terms for protein sequences with the Transformer model

— [Source link](#) 

Dat Duong, Lisa Gai, Ankith Uppunda, Don Le ...+3 more authors

Institutions: University of California, Los Angeles

Published on: 02 Feb 2020 - bioRxiv (Cold Spring Harbor Laboratory)

Topics: Transformer (machine learning model), UniProt and Convolutional neural network

Related papers:

- [Deep_CNN_LSTM_GO: Protein function prediction from amino-acid sequences.](#)
- [Classification of G-protein coupled receptors based on a rich generation of convolutional neural network, N-gram transformation and multiple sequence alignments.](#)
- [protein2vec: Predicting Protein-Protein Interactions Based on LSTM](#)
- [DEEPGONET: Multi-Label Prediction of GO Annotation for Protein from Sequence Using Cascaded Convolutional and Recurrent Network](#)
- [DeepFunc: A Deep Learning Framework for Accurate Prediction of Protein Functions from Protein Sequences and Interactions.](#)

Share this paper:    

View more about this paper here: <https://typeset.io/papers/annotating-gene-ontology-terms-for-protein-sequences-with-2efgclojgl>

1 Annotating Gene Ontology terms for protein sequences with 2 the Transformer model

3 Dat Duong, Lisa Gai, Ankith Uppunda, Don Le, Eleazar Eskin, Jingyi Jessica Li, Kai-Wei
4 Chang.

5 University of California Los Angeles, California, USA.

6 datdb@cs.ucla.edu, eeskin@cs.ucla.edu, jli@stat.ucla.edu, kwchang@cs.ucla.edu

7 Abstract

8 Predicting functions for novel amino acid sequences is a long-standing research problem. The
9 Uniprot database which contains protein sequences annotated with Gene Ontology (GO) terms,
10 is one commonly used training dataset for this problem. Predicting protein functions can then
11 be viewed as a multi-label classification problem where the input is an amino acid sequence and
12 the output is a set of GO terms. Recently, deep convolutional neural network (CNN) models have
13 been introduced to annotate GO terms for protein sequences. However, the CNN architecture
14 can only model close-range interactions between amino acids in a sequence. In this paper, first,
15 we build a novel GO annotation model based on the Transformer neural network. Unlike the
16 CNN architecture, the Transformer models all pairwise interactions for the amino acids within a
17 sequence, and so can capture more relevant information from the sequences. Indeed, we show
18 that our adaptation of Transformer yields higher classification accuracy when compared to the
19 recent CNN-based method DeepGO. Second, we modify our model to take motifs in the protein
20 sequences found by BLAST as additional input features. Our strategy is different from other
21 ensemble approaches that average the outcomes of BLAST-based and machine learning predictors.
22 Third, we integrate into our Transformer the metadata about the protein sequences such as 3D
23 structure and protein-protein interaction (PPI) data. We show that such information can greatly
24 improve the prediction accuracy, especially for rare GO labels.

25 1 Introduction

26 Predicting protein functions is an important task in computational biology. With the cost of
27 sequencing continuing to decrease, the gap between the numbers of labeled and unlabeled
28 sequences continues to grow [18]. Protein functions are described by Gene Ontology (GO)
29 terms [16]. Predicting protein functions is a multi-label classification problem where the
30 input is an amino acid sequence and the output is a set of GO terms. GO terms are
31 organized into a hierarchical tree, where generic terms (e.g. cellular anatomical entity)
32 are parents of specific terms (e.g. perforation plate). Due to this tree structure, if a GO
33 term is assigned to a protein, then all its ancestors are also assigned to this same protein.

34 When analyzing only the amino acid sequence data to predict protein functions, there
35 are two major trends. The first trend relies on string-matching models like Basic Local
36 Alignment Search Tool (BLAST) to match the unknown sequence with labeled proteins
37 in the database [11]. Recently, Zhang *et al.* [18] combined BLAST with Position-Specific
38 Iterative Basic Local Alignment Search Tool (PSI-BLAST) to retrieve even more labeled

39 proteins which are possibly related to the unknown sequence. The key idea behind
40 BLAST methods is to retrieve proteins that resemble the unknown sequence; most likely,
41 these retrieved proteins will contain similar evolutionarily conserved regions and motifs
42 (e.g. kinase domain) that match well with the unknown amino acid sequence. Then, all
43 GO labels assigned to these retrieved proteins are assigned to the unknown sequence [18].
44 BLAST methods do not explicitly estimate how much a specific motif affects the predicted
45 probability for a GO label.

46 The second trend transforms the amino acid sequences into features, then applies
47 classification methods to these features. For example, DeepGO converts an amino acid
48 sequence into a string of k -mers, where each k -mer is represented by a vector so that the
49 amino acid sequence is represented as a matrix m [8]. Given m , the next objective is to
50 find a function $f(m, g)$ that returns the correct assignment for the label g . Neural network
51 models like DeepGO and related work on DNA sequences use the convolutional neural
52 network (CNN) as the key component for this function f [8, 19].

53 In principle, neural network methods are different from BLAST methods; for a specific
54 GO label, a neural network model tries to extract amino acid patterns in the train data that
55 are most correlated with label g . When a test sample contains these amino acid patterns,
56 then f will return a high predicted probability for g given this test sample. To obtain
57 higher accuracy, one may need a very complex neural network model that can extract
58 more information from the training samples. However, due to the complex nature of deep
59 neural networks, it is difficult to say how such patterns, if found, are being analyzed by
60 the models.

61 In this paper, first, we introduce a novel method GOAT: **GO** annotation based on
62 the Transformer framework [17]. In the context of protein sequences, the Transformer
63 architecture is better than a convolutional neural network, because the Transformer models
64 all the pairwise interactions for the amino acids in the same sequence and can possibly
65 capture meaningful long-range relationship among these amino acids.

66 Second, we reconcile the gap between BLAST and neural network models. As ex-
67 plained, BLAST models retrieve sequences with conserved domains and motifs similar
68 to the unlabeled protein, but do not apply machine learning techniques on these motifs
69 to predict the GO labels. Neural network models want to correlate these motifs and the
70 specific labels via the function f , but are not guaranteed to discover such amino acid
71 patterns, especially when the annotated datasets are sparse. For this reason, we identify
72 motifs in the protein sequences via tools like PROSITE [14], and then use this information
73 as extra features for the label classification.

74 Third, we evaluate how protein metadata such as 3D structures and protein-protein
75 interaction (PPI) network data can affect the prediction. Our work focuses on neural
76 network models, whereas previous work such as Zhang *et al.* [18] focused on BLAST-
77 based methods.

78 In the following sections, we describe two baseline methods and the Transformer
79 architecture in our software GOAT. Next, we introduce three types of metadata as extra
80 features for GOAT: Domain (e.g. motifs), 3D-structure, and PPI network information.
81 We find that in the absence of metadata, GOAT is better than DeepGO [8], especially for
82 rare GO terms. Additionally, GOAT with protein metadata obtains higher recall rates
83 than the BLAST-based model in [18]. Our ablation study shows that motif data and 3D

84 folding information are meaningful factors, but PPI network is the most important. We
85 hope that our software GOAT will serve as a meaningful baseline for future research on
86 neural network models for GO annotations. GOAT is available at [https://github.com/
87 datduong/GOAnnotationTransformer](https://github.com/datduong/GOAnnotationTransformer).

88 2 Method

89 2.1 BLAST and PSI-BLAST

90 We describe our first baseline which is a very competitive BLAST-based method by Zhang
91 et al. [18]. Zhang et al. [18] consider several best matched proteins from BLAST and
92 additional sequences attained by PSI-BLAST when predicting annotations. The authors
93 [18] refer to their BLAST and PSI-BLAST method as the sequence-based module of their
94 software MetaGO. Here, we refer to it as $\text{MetaGO}_{\text{BLAST}}$, and briefly describe this method.
95 Let N^{blast} and N^{psiblast} be the number of sequences in the train data retrieved by BLAST
96 and PSI-BLAST that best match the unknown protein, and q be a GO label. Next define
97 $N^{\text{blast}}(q)$ and $N^{\text{psiblast}}(q)$ as the number of sequences having the label q in N^{blast} and N^{psiblast} .
98 Let $S_n^{\text{blast}}(q)$ and $S_n^{\text{psiblast}}(q)$ be the sequence-similarity scores of the n^{th} retrieved sequence in
99 $N^{\text{blast}}(q)$ and $N^{\text{psiblast}}(q)$ which contains the GO label q . The assigned prediction probability
100 for label q with respect to the unknown query sequence is

$$\text{score}(q) = w \frac{\sum_n^{N^{\text{blast}}(q)} S_n^{\text{blast}}(q)}{\sum_n^{N^{\text{blast}}} S_n^{\text{blast}}} + (1 - w) \frac{\sum_n^{N^{\text{psiblast}}(q)} S_n^{\text{psiblast}}(q)}{\sum_n^{N^{\text{psiblast}}} S_n^{\text{psiblast}}} \quad (1)$$

101 where $w = \max_n S_n^{\text{psiblast}}$ so that BLAST has a stronger weight when very close homologs
102 are found [18]. In other words, the prediction scores for the GO labels are scaled by how
103 similar the query protein is to the known sequences.

104 2.2 Convolutional neural network

105 We describe our second baseline DeepGO Kulmanov et al. [8] which is built from the
106 convolution neural network (CNN) architecture. We consider the DeepGO version which
107 does not correct for consistent predicted probabilities. Consistency is defined as the fact
108 that if a GO label is assigned to a protein then all its ancestors must also be assigned
109 to the same protein. This consistency is corrected for each GO label, where the final
110 prediction is the maximum of its own predicted probability and the probabilities of its
111 descendants. In other words, in DeepGO when a descendant of a specific label has high
112 predicted probability then this label will also have high predicted probability. There
113 are other types of correction [10] which were not compared in DeepGO. We believe this
114 research direction requires its own analysis. Moreover, consistency correction relies on
115 the per-term prediction accuracy. For this reason, we focus comparing GOAT to only the
116 basic DeepGO.

117 DeepGO converts an amino acid sequence, for example $p = \text{MARS} \dots$, into a list of
118 overlapping 3-mers, e.g. MAR ARS \dots . Each 3-mer is represented by a vector of in \mathbb{R}^{128} ,
119 so that p is represented by a matrix $E_p \in \mathbb{R}^{128 \times (L-2)}$ where L is the sequence length. A
120 1D-convolution layer, 1D-maxpooling and Flatten are then applied to E_p , so that we have
121 $v_p = \text{flatten}(\text{maxpool}(\text{conv1d}(E_p)))$ as the vector representing this k-mer sequence. To
122 predict a GO label i , DeepGO fits a logistic regression layer $\text{sigmoid}(B_i^T v_p + b_i)$ with the
123 binary cross entropy as the objective loss function.

124 To integrate metadata about the protein (e.g. PPI network), DeepGO concatenates
125 this information into v_p before sending it through the logistic layer. For example, in the
126 original paper, Kulmanov et al. [8] convert proteins from a PPI network into vectors. Let
127 $c_p \in \mathbb{R}^d$ be the vector representing protein p in this PPI network; then the classification
128 layer becomes $\text{sigmoid}(B_i^T [v_p, c_p] + b_i)$ where $[v_p, c_p]$ is the concatenation of the two vectors.
129 We emphasize that the vector c_p does not have to be from the PPI network; in the results,
130 we evaluate the DeepGO model with vectors representing 3D structures of proteins (the
131 outcome of this experiment is not shown in table but explained in result section).

132 2.3 GO annotation method with Transformer

133 We introduce our novel GO annotation method with Transformer (GOAT), available at
134 <https://github.com/datduong/GOAnnotationTransformer>. GOAT takes as an input a
135 sequence of amino acids and GO labels. For illustration purposes, consider the protein
136 MAP Kinase-activated Protein Kinase 5 (UniProtKB O54992), which we will refer to by its
137 id O54992 for brevity. Let L denote the sequence length, and let $g_1 \dots g_G$ denote the names
138 of the labels to be predicted, where G is the number of labels to be predicted. Our input
139 will be the string $MSEDS \dots LPHEPQ g_1 \dots g_G$ of total length $L + G$. Next, define E as
140 the embeddings for the amino acids, for example E_M and E_S are the vectors representing
141 the amino acids M and S respectively.

142 Let E^G be the embeddings for the GO labels, so that $E_{g_1}^G$ and $E_{g_2}^G$ are the vectors
143 representing the first and second GO label g_1 and g_2 respectively. E^G is analogous to
144 Word2vec embedding; except in this case, instead of having a vector for each word in a
145 corpus, we will have a vector for each GO label in the train and test datasets. In this paper,
146 we set the vectors represent the amino acids and GO labels to be in the same dimension;
147 that is, E_M and $E_{g_1}^G$ are vectors of the same size. To reduce the number of parameters,
148 for the GO labels, we fix E^G as the definition-based GO embeddings from [6] instead of
149 setting it as a trainable parameter.

150 We add a position vector P_j to the j^{th} amino acid in the sequence; for example, the
151 first and second amino acid M and S will have the following two vectors, $E_M + P_1$ and
152 $E_S + P_2$. We observe that the position embedding makes sense for amino acids so that
153 the same amino acid appearing at different locations will be treated differently. However,
154 position embedding does not apply to GO labels; that is, the ordering of the labels should
155 not affect the prediction outcome. For this reason, we do not add position embedding to
156 the GO labels.

157 Next, we introduce the region-type embeddings R to highlight the fact that some
158 amino acids belong to known motifs. Again consider the protein O54992, which is 473
159 residues long and contains two key regions: a kinase motif at position 22-304 and a coiled

160 coil domain at position 409-440. In this case the 25th amino acid is *T* and is inside the
 161 kinase motif; for this reason, it will have the vector $E_T + P_{25} + R_{\text{kinase}}$. Likewise, the 410th
 162 amino acid is *N* and will have the vector $E_N + P_{410} + R_{\text{coiled coil}}$. Amino acids outside any
 163 key regions will not have region embedding added to them. Motifs can be found by using
 164 PROSITE; fortunately, many labeled sequences in Uniprot already have this information
 165 [5, 14].

166 We now describe how the Transformer architecture in GOAT analyzes the input se-
 167 quence. The original Transformer has 12 layers of encoders and each layer has 12 in-
 168 dependent identical units (referred to as heads in the original paper) [17]. To keep our
 169 software manageable for all users, in this paper, we use only one head and so we will
 170 exclude description of head. We will use 12 layers. The first layer takes as arguments the
 171 vectors representing the input string. Here we simplify the notation, let w_j be the vector
 172 representing the j^{th} element in the input string. For protein O54992 of length $L = 473$
 173 and G number of GO labels, from the input string $MSEDS \dots LPHEPQ g_1 \dots g_G$, we will
 174 have for example $w_{25} = E_T + P_{25} + R_{\text{kinase}}$, and $w_{474} = E_{g_1}$. At the first layer, we have

$$o_{1j} = \sum_{k \in \{1:(L+G)\}} a_{jk} V^1(w_k) \quad (2)$$

$$a_{jk} = \text{softmax}(e_{jk}) \quad (3)$$

$$e_{jk} = Q^1(w_j)^\top K^1(w_k) \quad (4)$$

175 V^1, Q^1, K^1 are transformation functions for layer 1. o_{1j} is a weighted sum of the
 176 transformed vectors representing itself and the other entities in the input string. o_{1j}
 177 is then transformed as $p_{1j} = L_2^1(\text{gelu}(L_1^1 o_{1j}))$ where L_1^1 and L_2^1 are two linear transfor-
 178 mations with the gelu activation function in between. The final output of Layer 1 is
 179 $h_{1j} = \text{LayerNorm}(p_{1j} + o_{1j})$. Loosely speaking, the first layer computes all pairwise in-
 180 teractions of w_j and w_k for all k , where the attention a_{jk} in Eq. 3 indicates how much w_k
 181 contributes toward w_j .

182 The second layer takes the output of the first layer as its input, so that we have for any
 183 layer i

$$o_{ij} = \sum_{k \in \{1:L+G\}} a_{jk} V^i(h_{i-1,k}) \quad (5)$$

$$a_{jk} = \text{softmax}(e_{jk}) \quad (6)$$

$$e_{jk} = Q^i(h_{i-1,j})^\top K^i(h_{i-1,k}) \quad (7)$$

184 where V^i, Q^i, K^i are transformation functions for layer i . This layer i will have its own
 185 linear transformations L_1^i and L_2^i to transform o_{ij} . Again, loosely speaking, layer i computes
 186 all pairwise interaction for the output from the previous layer $i - 1$.

187 At layer 12, we focus only on the output $h_{12,k}$ corresponding to the GO labels. Let
 188 us denote h_{12,g_i} as the final output for the term g_i . We use a single linear classifier
 189 $\text{softmax}(Ch_{12,g_i})$ to return the presence and absence probability of g_i for the input protein.

190 The same transformation C is applied to all labels, so that a set S of GO terms having
191 similar $h_{12,g_i \in S}$ will have similar predictions.

192 At each label g_i , the output h_{12,g_i} encapsulates all the information from the amino acids
193 and from all the other labels, so that values which affect h_{12,g_i} will also affect the output
194 h_{12,g_j} at another label g_j . Intuitively, with this fact and the fact that all labels share the
195 same classifier C , the prediction at g_i and g_j are to some degree correlated. We suspect
196 that Transformer can model co-occurrences of labels. To validate this, from the T-SNE
197 plot of the vectors h_{12,g_i} in the result section, we observe that a label and its ancestors will
198 be nearby, even when their initial definition embeddings E_{g_i} in [6] are dissimilar, as is the
199 case when a term and its distant ancestors can have dissimilar definitions.

200 When there are metadata for the proteins, such as their embedding c_p from a PPI
201 network, we can concatenate such embeddings into h_{12,g_i} as in DeepGO. Next, we use a
202 two-layer fully connected classifier, $\text{softmax}(C_1(\text{relu}(C_2[c_p, h_{12,g_i}])))$ to return the presence
203 and absence probability of g_i , where C_1 and C_2 are shared for all the labels.

204 3 Results

205 We present three main results in the following sections. First, when using amino acid
206 sequence data alone as input, we show that our adaptation of Transformer in GOAT is
207 better than the CNN-based DeepGO at predicting protein functions. We will use the
208 name $\text{GOAT}_{\text{BASE}}$ and $\text{DeepGO}_{\text{BASE}}$ to indicate the base implementation in both models,
209 and ignore the subscripts when the context is clear. Second, we show that GOAT obtains
210 higher classification accuracy when it takes as extra inputs the motif information from
211 the protein sequences. We use the name $\text{GOAT}_{\text{MOTIF}}$ for this version of GOAT. We also
212 observe how the Transformer architecture in GOAT analyzes the motif information when
213 it predicts GO labels for an amino acid sequence. Third, because motif information is a key
214 input of GOAT, we integrate 3D-structure and PPI network metadata about the proteins
215 on top of our $\text{GOAT}_{\text{MOTIF}}$ to obtain even better prediction outcome. We use the name
216 $\text{GOAT}_{\text{MOTIF,3D}}$, $\text{GOAT}_{\text{MOTIF,PPI}}$, $\text{GOAT}_{\text{MOTIF,3D,PPI}}$ to indicate joint inputs in GOAT.

217 We use the datasets from the original DeepGO paper, which contain one dataset each
218 for the BP, MF, and CC ontologies. We remove proteins without any GO annotations. The
219 BP, MF and CC datasets have 27279, 18894, and 26660 train proteins, and 9096, 6305, and
220 8886 test proteins, respectively. In the DeepGO datasets, BP, MF and CC labels annotating
221 fewer than 250, 50, and 50 proteins are removed. The ancestors of all the terms annotating
222 one protein are also added into the ground truth label set. In total, the numbers of BP, MF
223 and CC terms in the label sets are 932, 589, and 439 respectively.

224 We measure the per-label accuracy using Macro and Micro AUC which are the un-
225 weighted and weighted averages of the AUC at each label, respectively. We are more
226 interested in the accuracy for rare labels because these labels are closer to the true func-
227 tions of the proteins. Rare labels affect Macro AUC more than Micro AUC; however a high
228 Macro AUC does not always guarantee that rare labels are correctly classified. For exam-
229 ple, the BP term *protein glycosylation* is more precise to a protein function than its ancestors
230 *metabolic process* and *cellular process*. However, by design, DeepGO datasets include all the
231 ancestors for a label. The label *protein glycosylation*, *metabolic process* and *cellular process* will

232 now occur 254, 12455 and 19232 times in the data, respectively. If *metabolic process* and
233 *cellular process* are correctly predicted, then Macro AUC is high but does not imply that
234 we can properly predict *protein glycosylation*.

235 For this reason, to evaluate the models, we also compute recall-at-k (R@k) which mea-
236 sures the per-protein accuracy. In practice, a classifier would return k of most probable
237 labels for an unknown sequence, which a curator can then review. These k labels are
238 referred to as *top-k labels* because they are the k labels having the highest predicted proba-
239 bilities for an input sequence. For one protein, R@k measures the fraction of correct labels
240 retrieved among the top-k labels. We report the final R@k which is the average R@k over
241 all test samples evaluated with respect to the label set of interest. We evaluate R@k on the
242 entire label set and also on the sets of rare labels.

243 3.1 Base GOAT

244 To compare the base architectures in GOAT and DeepGO, we fit these two models on
245 only amino acid sequences without any extra information such as motifs or PPI network
246 data. For DeepGO, we use the same hyperparameters as the original paper [8]. For the
247 Transformer parameters in GOAT, we set input embedding at size 256 (that is $E^G \in \mathbb{R}^{932 \times 256}$
248 for MF), and intermediate vectors at size 512. To keep our software GOAT manageable for
249 all users, we implement Transformer with only one head and 12 layers. We initialize the
250 GO embedding E^G as the pre-trained embedding $BERT_{LAYER12}$ in [6] which transforms the
251 GO definitions into vectors, where GOs having related definitions will have comparable
252 vectors. Using pre-trained GO embeddings reduces the number of parameters in the
253 Transformer, which can also reduce overfitting, use less GPU memory, and decrease run
254 time. We train Transformer on a single GTX 1080 Ti with 11GB memory for all three
255 datasets.

256 In BP, MF and CC data, $GOAT_{BASE}$ exceeds $DeepGO_{BASE}$ in Macro and Micro AUC,
257 indicating that the base implementation of GOAT attains better per-label accuracy (Table
258 1 row 3 and 5). To estimate per-protein accuracy, we select R@50, R@30, and R@30 for the
259 BP, MF and CC data, respectively (approximately 5% of total label size). On the whole
260 data, GOAT increases recall by a small amount. Recall for the entire label set can be
261 affected by common GO terms which are often easier to classify compared to rare labels.
262 As discussed in the previous section, accurately predicting rare terms is more important
263 for the proteins, because rarer labels describe more detailed biological events which reflect
264 the true properties of the unknown proteins.

265 Table 2 shows the R@k for 232 BP, 143 MF, and 110 CC rare labels which appear below
266 the 25th quantile occurrence frequency in the label sets. For recall rates to make sense in
267 this case, we compute recall rates for the proteins annotated by at least one of these rare
268 labels. In the test data, GOAT now has a noticeable improvement over DeepGO, especially
269 for larger sets of top-k labels (Table 2 row 3 and 5). This evaluation indicates that our
270 adaptation of the Transformer framework in GOAT can extract more useful information
271 from the amino acid sequences, and thus obtain higher prediction accuracy.

272 We next evaluate whether our adaptation of Transformer can learn the co-occurrences
273 of labels. Duong et al. [6] noticed in their GO embedding that when one of the child-
274 parent GO labels describes very broad biological events (e.g. low IC), then their vector

275 representations may be far apart. This fact implies that for Transformer to work well,
276 to some degree it must learn the co-occurrences of labels and adjust E^G so that any two
277 related GO labels (regardless of their frequencies in the train data, IC values and distance
278 to roots) will have comparable vectors. To observe that Transformer can implicitly learn
279 label co-occurrences, we compare the T-SNE plots of the input GO embedding E^G and its
280 output h_{12,g_i} from the Transformer layer 12 which is directly passed into the classification
281 layer.

282 For every input protein, we have a different value of h_{12,g_i} for the same label g_i be-
283 cause h_{12,g_i} is function of the vector representing the amino acids. We apply our trained
284 Transformer on the test set, and take the average h_{12,g_i} over each input proteins in test data
285 (denoted as \bar{h}_{12,g_i}). We compute \bar{h}_{12,g_i} from the test data because these proteins are not
286 seen in training and provide a more realistic evidence. We use \bar{h}_{12}^G to denote the set of
287 \bar{h}_{12,g_i} for all g_i .

288 Figure 1 shows the T-SNE plot of E^G for the MF labels in DeepGO dataset; we highlight
289 two terms GO:0008376 (red) and GO:0030291 (blue) and their ancestors. The dot size is
290 scaled by IC values, where smaller size implies lower IC (so the label is more common).
291 Smaller dots tend to cluster well together but large dots do not. For example, consider the
292 term GO:0016740 and its parent GO:0003824 (top right and far left red nodes) which should
293 often co-occur because in our dataset ancestors of an assigned GO label are included as the
294 ground truth labels. For Transformer to work well, it should reposition the h_{12,g_i} vectors
295 representing GO:0016740 and GO:0003824 closer together.

296 The T-SNE plot of \bar{h}_{12}^G from Transformer shows that the red and blue nodes have
297 clustered more closely compared to E^G . The blue dots now gather at the bottom of Fig,
298 as compared to being two separated groups in Fig. The two example cases GO:0016740
299 and GO:0003824 are now near one another, bottom of Fig. However, the lowest level
300 red node GO:0008376 remains far from its ancestors. The red and blue dots are not yet
301 tightly compacted into two dense clusters, and so there is room for further development.
302 In conclusion, Transformer weakly models the co-occurrences of labels even when such a
303 constraint is not explicitly enforced.

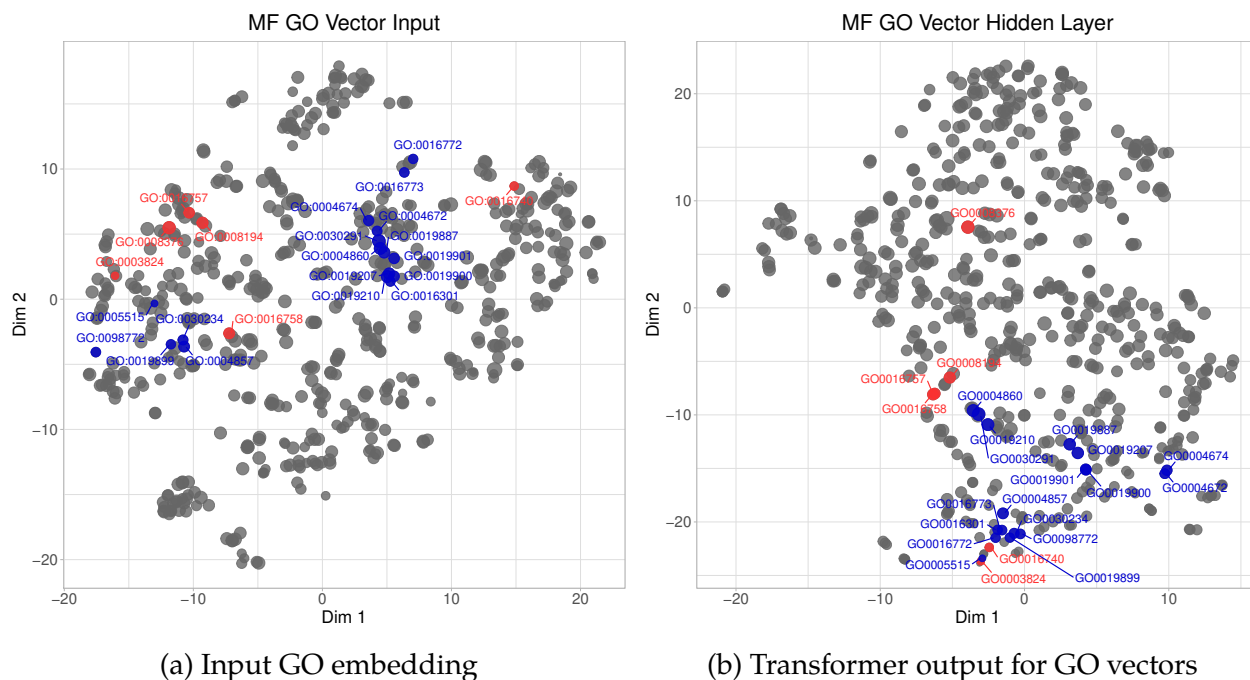


Figure 1: T-SNE of input GO embeddings and their transformed values created by Transformer layer 12.

304

Table 1: Macro AUC, Micro AUC and Recall-at-k (R@k) are evaluated on the entire 932 BP, 589 MF, and 439 CC labels in the original DeepGO data. k value in R@k corresponds to about 5% of total label size in each dataset.

		BP			MF			CC		
		Macro	Micro	R@50	Macro	Micro	R@30	Macro	Micro	R@30
BLAST Psi-BLAST										
1	Evalue10	65.99	81.34	48.02	76.51	87.12	69.75	64.17	89.36	77.24
2	Evalue100	66.61	84.41	48.84	78.67	90.34	68.06	65.62	92.54	80.23
DeepGO										
3	BASE	62.60	82.17	46.13	73.90	87.49	59.63	67.12	93.07	81.47
4	+PPI	82.16	90.31	56.97	84.97	92.40	70.19	87.51	96.76	87.98
GOAT										
5	BASE	67.69	84.46	47.40	78.67	89.43	60.46	74.94	93.95	82.98
6	+MOTIF	71.04	85.64	48.65	82.53	91.12	66.19	77.62	94.51	83.01
7	+MOTIF+3D	72.62	86.09	50.76	85.38	92.72	69.23	79.57	94.95	84.22
8	+MOTIF+PPI	83.68	90.49	57.81	88.92	93.98	72.16	90.76	97.23	87.95

Table 2: Recall-at-k (R@k) are evaluated on the most uncommon 232 BP, 143 MF, and 110 CC labels from the entire set of size 932 BP, 589 MF, and 439 CC labels in the original DeepGO data.

		BP			MF			CC		
		R@10	R@20	R@50	R@5	R@10	R@30	R@5	R@10	R@30
BLAST Psi-BLAST										
1	Evalue10	25.13	32.59	47.33	44.87	51.11	57.51	25.09	30.91	49.27
2	Evalue100	21.79	29.88	46.29	42.31	48.37	60.98	21.91	29.72	50.47
DeepGO										
3	BASE	11.88	20.21	39.75	17.42	27.07	50.47	13.28	24.57	48.56
4	+PPI	45.62	57.78	74.55	41.96	52.85	71.98	53.14	64.85	82.58
GOAT										
5	BASE	13.49	23.32	44.54	18.80	30.95	58.87	17.89	26.47	51.21
6	+MOTIF	17.27	27.75	48.11	30.45	41.72	64.76	19.06	31.03	58.94
7	+MOTIF+3D	23.66	34.27	55.06	36.19	47.90	72.23	27.36	39.72	67.69
8	+MOTIF+PPI	42.84	56.01	74.78	45.35	59.49	82.67	48.04	62.14	84.42

3.2 Domains and motifs in amino acid sequences as features

Before the introduction of neural network models, earlier methods often integrated BLAST as a key component. BLAST retrieves annotated proteins in the database that share the same conserved domains or motifs with the unknown sequence, and then assigns the GO labels of these retrieved proteins to the unknown query. Loosely speaking, key domains or motifs shared by the training sequences can then be considered as the key factors in BLAST-based methods. In this section, we evaluate whether neural network models can automatically learn these key patterns from the training sequences, and explain how to introduce these patterns as input features to GOAT.

For fair comparison, we select a very strong BLAST baseline $\text{MetaGO}_{\text{BLAST}}$ [18] and apply it to the same DeepGO datasets. We emphasize that $\text{MetaGO}_{\text{BLAST}}$ in [18], which matches multiple related sequences to the query, has much stronger performance than the BLAST baseline used in DeepGO where the authors select only a single best matching sequence [8]. We build the BLAST database from the DeepGO train data. For BLAST and PSI-BLAST, we perform the experiments using both e-value at 10 and 100. Lower e-values leave too many unmatched testing sequences; for example, in the CC dataset at e-value 1, only 7236 out of 8886 test samples match to some sequences in the train data. Moreover, in the context of finding possible protein functions, a higher e-value can allow for higher recall rates.

In BP and MF data, $\text{MetaGO}_{\text{BLAST}}$ is better than the base DeepGO and GOAT; for example, $\text{MetaGO}_{\text{BLAST}}$ yields better recall rates on rare labels (Table 1 and 2 row 1–3 and 5). We emphasize that $\text{MetaGO}_{\text{BLAST}}$ and $\text{GOAT}_{\text{BASE}}$ have comparable recall on rare MF labels for larger top-k label sets. Arguably, $\text{MetaGO}_{\text{BLAST}}$ and GOAT retrieve the same number of correct labels, but GOAT makes more spurious predictions. DeepGO however does not come close to $\text{MetaGO}_{\text{BLAST}}$ on recall rates. The convolutional network in DeepGO is likely not complex enough to learn motifs shared among the sequences with related

331 functions, which BLAST or PSI-BLAST can detect.

332 When evaluated on the entire CC data, the base DeepGO and Transformer outperform
333 $\text{MetaGO}_{\text{BLAST}}$. However, for rare CC labels, when compared to $\text{MetaGO}_{\text{BLAST}}$, Transformer
334 has higher $R@30$ but lower $R@5$ and $R@10$. To some degree, our current GOAT is not yet
335 better than $\text{MetaGO}_{\text{BLAST}}$.

336 We had hoped that the complex Transformer architecture in GOAT would have learned
337 key motifs missed by the simpler convolutional neural network in DeepGO, but this is not
338 the case. There are information about the amino acid sequence that we must explicitly
339 specify for GOAT to function better. For this reason, we use the motifs extracted by string-
340 matching methods as inputs to our method GOAT. For example, we can apply PROSITE
341 to scan the protein database for known motifs in a given input sequence [14]. Loosely
342 speaking, we are combining motif-based methods and the Transformer architecture into
343 one pipeline by taking the output of motif-based models and passing them as inputs to
344 our Transformer. Our strategy is different from an ensemble approach that would average
345 the prediction of independent predictors.

346 In the manually reviewed Uniprot database, each protein already has a Family & Do-
347 mains section describing its key regions [16]. We then add these region types of the amino
348 acids as input features into our method GOAT. We emphasize that not all domains are
349 meaningful at predicting labels in a certain ontology. For example, Serine/threonine-
350 protein kinase TBK1 (UniProtKB number Q9UHD2) has a kinase domain at position 9-310
351 (PROSITE annotation rule PRU00159). However, kinase domain involves in a wide range
352 of biological processing at various locations in the cell like metabolism, transcription,
353 cytoskeletal rearrangement and movement, and cell apoptosis and differentiation [7, 13].
354 Thus, this kinase domain in Q9UHD2 tells us which Molecular Functions are more likely
355 to be assigned, but this domain cannot tell which Biological Processes or Cellular Com-
356 ponents Q9UHD2 will have.

357 For each protein in the original DeepGO datasets, we download its sequence annotation
358 rule (e.g. PROSITE rule) from the 2019 data at <https://www.uniprot.org/>. We do not
359 consider region types for sequences that have changed in length; otherwise, we consider
360 region types only for portions that have not changed in amino acids composition. Uniprot
361 data divides region types into subgroups. Some subgroups require curated comments and
362 are not truly applicable for analyzing new proteins; for example, the subgroup Domain
363 Non-positional Annotation is not determined by sequence analysis.

364 We use the following six subgroups which can be found by sequence analysis: Zinc
365 finger (e.g. C2H2-type), Repeat (e.g. AA tandem repeat), Motif (e.g. LXXLL motifs),
366 Compositional bias (e.g. Asp/Glu-rich), Coiled coil (e.g. Leucine-zippers), Domain (e.g.
367 Ser/Thr kinase domain). In the original DeepGO datasets, we found 1629, 1450 and 1655
368 amino acid region types for the BP, MF and CC train data, respectively. Region types
369 found in test sequences but not seen in the train data are set as zero; effectively we treat
370 these cases as if the region types do not exist. To model the region types in the amino acid
371 sequence, we apply the region-type embedding R explained in section ; for example, in
372 BP ontology we will have a embedding $R \in \mathbb{R}^{1629 \times 256}$. We will use the name MOTIF to
373 denote all types of Domain information in these six subgroups reported by The UniProt
374 Consortium [16].

375 When evaluated on the entire data, $\text{GOAT}_{\text{MOTIF}}$ obtains better AUCs and comparable

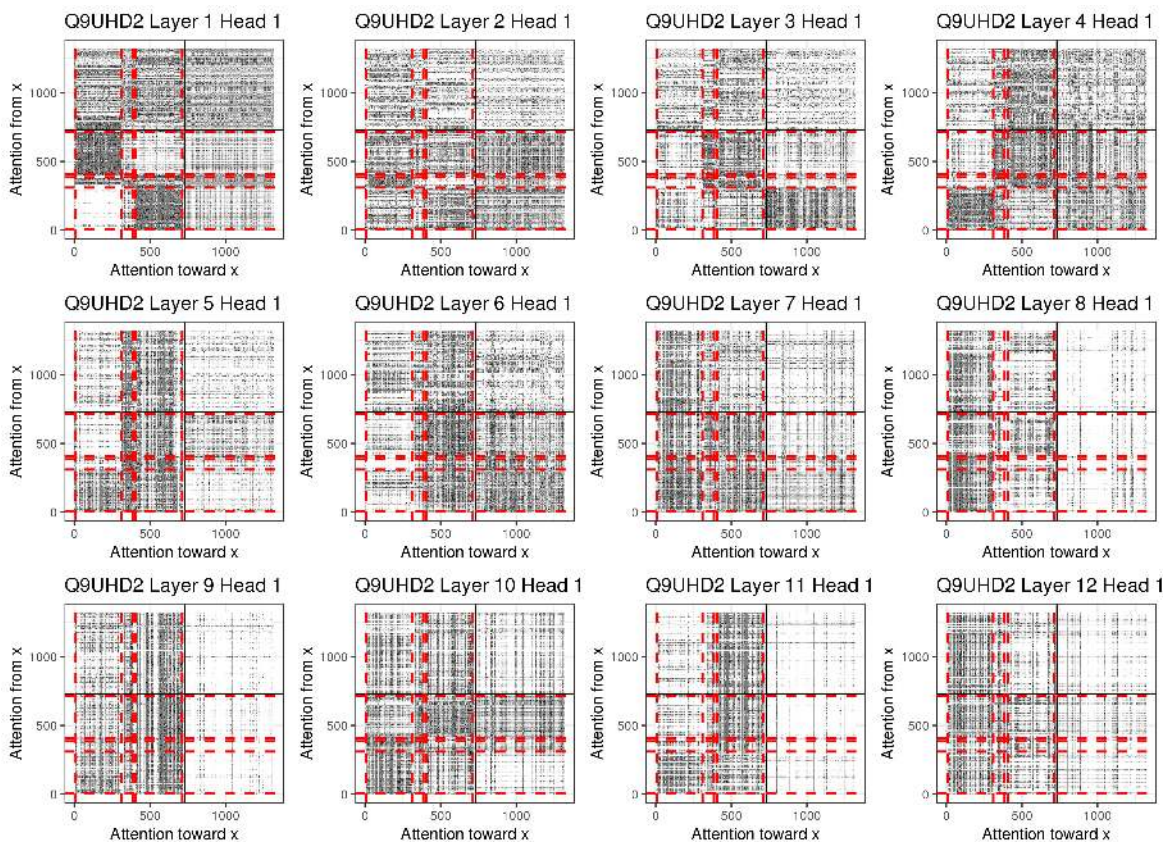
376 recall rates to MetaGO_{BLAST} (Table 1 row 6). When evaluated on rare labels, our GOAT
377 has subpar recalls when the top-k label set is small; however, for larger top-k label set,
378 especially in MF and CC data, GOAT retrieves more correct labels (Table 2 row 6). This
379 result implies that GOAT_{MOTIF} is a better classifier when the top-k label set is larger. Our
380 result also highlights an important point. Ideally, a neural network should teach itself the
381 patterns associated with certain key protein functions. However, a neural network may
382 fail to learn such key information, and needs these inputs to be explicitly provided. These
383 key domains within a sequence are often easily obtained, and in fact already available in
384 Uniprot.

385 To observe how the MOTIFS are analyzed by the Transformer architecture in GOAT, we
386 select the human protein Serine/threonine-protein kinase TBK1 (UniProtKB Q9UHD2) in
387 DeepGO test data. As described in section 3.2, we concatenate the amino acid sequence
388 and GO labels into one single input into GOAT. Q9UHD2 is 729 amino acids long, and
389 when predicting MF labels, the input into GOAT is then a string of amino acids and GO
390 labels of length 1318 (from 729 + 589 MF labels). We [...something about region-type emb]
391 integrate into the Transformer framework the three key domains in Q9UHD2 derived from
392 sequence analysis methods; these are Protein Kinase at 9-310, Ubiquitin-like at 309-385,
393 and Coiled coil at 407-713. The vertical and horizontal red lines indicate these regions in
394 the attention heatmap (Figure 2).

395 We plot the attention heatmap of α_{jk} in each layer (Figure 2). α_{jk} measures how much
396 position k contributes toward position j (Eq. 3). Each row in the heatmap adds to 1. The
397 heatmap is divided into four quadrants. The first quadrant shows the interactions of GO
398 labels among themselves (e.g. α_{jk} for $j, k \in [730, 1318]$), the second shows contribution
399 of amino acids toward the GO labels, the third shows interactions of amino acids among
400 themselves (e.g. α_{jk} for $j, k \in [1, 729]$), and the fourth shows contribution of GO labels
401 toward the amino acids.

402 Transformer without region-type embedding has a noisy attention heatmap (Appendix
403 Fig. 3). Transformer with region-type embedding displays meaningful patterns (Fig. 2).
404 For example, layer 1 illustrates the cross interactions between Protein Kinase and Coiled
405 coil domain (black boxes quadrant 3); whereas layer 4 and 8 show interactions within the
406 Protein Kinase and Coiled coil themselves. In layer 12, the final vectors representing GO
407 labels receive more attention from the Protein Kinase than the other regions (top right box
408 quadrant 2). Because these final output vectors are sent to the classification layer, we can
409 assume that the Protein Kinase region contributes more to the label annotation compared
410 the other domains. This observation is consistent with the true molecular functions of
411 Q9UHD2 which are: phosphoprotein binding, protein kinase activity, protein phosphatase
412 binding, and protein serine/threonine kinase activity.

Figure 2: Heatmap of the attention values α_{jk} in each layer when analyzing the protein kinase TBK1 (UniProtKB Q9UHD2). The three key regions of this sequence (separated by red lines) are explicitly given as inputs to the Transformer model. The first quadrant shows the interactions among the GO labels, the second shows contribution of amino acids toward the GO labels, the third shows interactions of amino acids among themselves, and the fourth shows contribution of GO labels toward the amino acids.



413 3.3 Protein vectors as features

414 Zhang et al. [18] evaluates how PPI network and 3D structure data affect the annotation
415 accuracy for methods built from BLAST. In this paper, we assess the contributions of
416 these components in the context of a neural network classifier. The amino acid sequences
417 have already been used in neural network models, as described in DeepGO and our
418 GOAT. Any other information about protein must come from some external resources
419 such as the STRING database [15]. For example, one can train a neural network on the
420 STRING database to transform interacting proteins into similar vectors [1, 4]. Integrating
421 these protein vector representations into annotation methods is motivated by the fact
422 that interacting proteins (ideally encoded into similar vectors) should have closely related
423 functions (e.g. found in the same biological processes and cellular locations).

424 It is only recently that proteins in knowledge graph have been transformed into vectors
425 via neural network models. In this paper, we will not build new models to encode
426 proteins from knowledge graph, and reserve this topic for future work. We will focus on

427 evaluating how much can protein vectors built from external data sources increase the
428 prediction accuracy.

429 3.3.1 Vectors representing protein 3D structure

430 We evaluate protein vectors that capture high-level 3D shapes of the proteins (e.g. α -
431 helices and β -sheets) [3]. [3] applied a 3-layer Bidirectional Long-Short Term Memory
432 (BiLSTM) to encode an amino acid sequence into a matrix. [3] trained their model on the
433 SCOP database and residue-residue contact prediction. The SCOP database describes the
434 major classes of 3D-structures often seen in proteins. Their model was trained on SCOPe
435 ASTRAL 2.06 dataset with 22,408 amino acid sequences, and each training epoch has
436 100,000 pairs sampled from these 22,408 sequences [3]. From SCOP, Bepler and Berger [3]
437 predict whether two protein sequences have no relationship, class-level relationship, fold-
438 level relationship, superfamily-level relationship, or family-level relationship (e.g. label
439 $y = 0, 1, 2, 3, 4$) [3]. For example, two proteins with the same Rossmann-fold structural
440 motif have a class-level relationship ($y = 1$ in this case). Residue-residue contact prediction
441 is applied within the same protein sequence; the objective is to predict whether each pair
442 of positions i, j within the same protein are close by (distance less than 8 angstroms) in
443 the 3D structure. Bepler and Berger [3] provide the pre-trained encoder which can return
444 a matrix for any amino acid sequence, even those not used in training. In this paper, we
445 take the mean-pool of this matrix to represent the entire sequence.

446 Because motif information is a key input of GOAT, we integrate 3D-structure data on
447 top of our GOAT_{MOTIF}. We emphasize that 3D-structures and motifs are targeting two
448 different kinds of information; for example, a kinase domain does not strictly entail a
449 specific folding pattern. For this reason, GOAT_{MOTIF,3D} improves upon GOAT_{MOTIF} (Table
450 1 row 7). More importantly, for a larger top-k label set, GOAT is much better than
451 MetaGO_{BLAST}, where our recall rates increase by about 9%, 12%, and 17% in BP, MF and
452 CC data respectively (Table 2 row 7). Our results indicate that GOAT find fewer correct
453 labels when the set of top-k labels is small; however, GOAT will retrieve more correct
454 labels than MetaGO_{BLAST} for larger set of top-k labels.

455 When we replace the PPI network in DeepGO with the vectors from SCOP in [3], the
456 performance significantly decreases. Macro AUC in DeepGO drops from 82.16 to 66.54
457 in BP, from 84.97 to 77.78 in MF, and from 87.51 to 68.93 in CC data. This decrement is
458 anticipated as we will argue in the next section, that protein interaction network has more
459 impact than 3D-structure information.

460 3.3.2 Vectors representing proteins in interaction network

461 We evaluate protein vectors that capture their relatedness in a protein-protein interaction
462 network. To be consistent with DeepGO, we use the same protein vectors in their paper
463 as input features for our GOAT. These vectors are created following the method in [1]. In
464 brief, DeepGO uses vectors representing the protein names in a protein-protein interaction
465 (PPI) network that has 8,478,935 proteins, and 11,586,695,610 edges total (derived from
466 STRING database). Following [1], DeepGO uses DeepWalk [?] to generate sentences from

467 the network, and apply Word2Vec [9] to these sentences to create the vector embedding
468 for the protein names. Effectively, interacting proteins will have similar vectors.

469 Because Domain information is a key input of GOAT, we integrate PPI network data
470 on top of our GOAT_{MOTIF}. GOAT_{MOTIF,PPI} exceeds the other Transformer models by large
471 margins (Table 1 row 8). On the entire label sets, in the presence of PPI network information
472 GOAT and DeepGO perform similarly, despite the fact that GOAT can better extract
473 information from the amino acid sequence (Table 1 row 4 and 8). Only for rare MF labels
474 does GOAT exceed DeepGO at every R@k by noticeable margins (Table 2 row 4 and 8).

475 Intuitively, it is reasonable that PPI network dominates the information from amino
476 acid sequences at classifying BP and CC labels, but not for MF labels. For example, two
477 interacting proteins can have distinct 3D structures and sequences (and thus motifs); yet,
478 they involve in the same biological process and sometimes found at the same cellular
479 components. The same two interacting proteins however can have dissimilar molecular
480 functions because they can induce very different chemical reactions.

481 We emphasize that vectors of PPI network in DeepGO does not need amino acid
482 sequence to retrieve vectors representing the proteins. However, this method will not
483 return vector representations for novel proteins not yet existed in the database. In practice,
484 for proteins not yet well studied, we may need a different approach and other metadata
485 for such proteins. We reserve this topic for future research work.

486 3.4 Evaluation on sparse GO labels

487 Many GO terms annotate only a few proteins because protein functions can be very
488 unique. In practice, parametric predictors must handle sparse labels to predict terms that
489 closely resemble the true protein functions. Yet, parametric models can fail when the
490 train data has too many sparse labels. In such cases, GOAT and DeepGO accuracy will
491 drop, because we need a label to have enough samples to reliably train the parameters in
492 a neural network model. It is then important to evaluate GOAT and DeepGO against the
493 nonparametric MetaGO_{BLAST} which does not have trainable parameters.

494 We evaluate GOAT and DeepGO on datasets that contain more rare labels. We reuse
495 the same proteins in the original DeepGO data but include labels with at least 50, 10, and
496 10 occurrences in the BP, MF and CC train data, whereas the same criteria in the original
497 DeepGO are 250, 50 and 50. Our larger datasets now have 2980 BP, 1697 MF and 989 CC
498 labels, respectively (versus the original 932 BP, 589 MF, and 439 CC labels). For each added
499 term, we include its ancestors as the gold-standard labels, so that most of the labels in the
500 original data now have higher occurrence frequencies.

501 We train the models on the entire larger dataset. In this experiment, we also include a
502 GOAT_{PPI} to evaluate the contribution of the PPI network data alone, and a GOAT_{MOTIF,3D,PPI}
503 to evaluate the most complete model. In this experiment, we are less interested in the
504 common labels, and evaluate R@k for the extra 2048 BP, 1108 MF and 550 CC labels which
505 are sparse compared to the labels in the original DeepGO data; for example, 95% of the
506 extra labels occur below 252, 34, and 68 times in the BP, MF and CC train data, respectively.
507 Table 3 shows the R@k for these sparse labels.

508 In Table 3, the base GOAT and DeepGO perform worst than MetaGO_{BLAST}. We em-
509 phasize that in this case, GOAT is still has better performance than DeepGO for MF and

510 CC data, but not for BP data (row 3 and 5). The PPI network data appears to be the most
 511 important factor; for example, $GOAT_{MOTIF,3D}$ is about the same as $MetaGO_{BLAST}$, whereas
 512 all the models with PPI network achieve higher R@k than $MetaGO_{BLAST}$ (Table 3).

513 When predicting biological processes, in the presence of PPI network embedding, the
 514 other protein metadata and the types of neural network model for amino acid sequences
 515 are not as important (Table 3 row 4, and 7–9). This result is an empirical evidence
 516 supporting our earlier hypothesis that PPI network can dominate sequence information;
 517 that is, two interacting proteins should be involved in the same biological processes even
 518 when their sequences display dissimilar motifs, 3D structures, or any other types of hidden
 519 information to be extracted by neural network models.

520 For MF and CC labels, integrating just PPI network embedding into DeepGO is not
 521 enough to raise recall rates; in this case, our GOAT model, Domain information and SCOP
 522 data are important complementary factors to the PPI network embedding. Indeed, for
 523 MF labels, in the presence of PPI network embedding, GOAT also benefits when having
 524 Domain and 3D-structure information as extra features (Table 3 row 7 and 9). For CC
 525 labels, the same observation holds true, except that Domain information now has more
 526 impact than 3D-structure data (Table 3 row 8 and 9).

Table 3: We increase the label size in the original DeepGO data from 932 BP, 589 MF, and 439 CC labels to 2980 BP, 1697 MF and 989 CC labels. Models are trained on the entire label sets, but R@k are evaluated only for the added 2048 BP, 1108 MF and 550 CC labels which are sparse. R@k are computed with respect to only proteins having these labels; there are 7850, 2671, and 1848 such proteins out of 9095, 6294, and 8886 samples in BP, MF and CC test data.

		BP			MF			CC		
		R@40	R@70	R@100	R@40	R@70	R@100	R@10	R@30	R@50
BLAST Psi-BLAST										
1	Evalue10	30.30	34.84	37.88	38.67	41.09	43.42	20.80	29.15	33.13
2	Evalue100	29.53	34.57	38.02	38.58	43.32	45.81	23.47	35.23	39.78
DeepGO										
3	BASE	23.14	27.98	32.01	16.62	23.83	29.60	22.49	31.53	37.96
4	+PPI	48.06	55.86	61.02	41.61	48.61	53.39	47.89	60.77	66.48
GOAT										
5	BASE	23.78	28.82	32.69	19.41	27.88	34.19	25.22	35.98	42.53
6	+MOTIF+3D	27.70	33.50	38.03	30.52	38.74	45.44	25.73	37.82	44.42
7	+PPI	43.51	52.52	58.54	41.77	52.46	58.51	41.98	60.02	68.70
8	+MOTIF+PPI	46.22	55.06	61.08	45.69	55.32	61.11	50.08	67.15	73.89
9	+MOTIF+3D+PPI	46.78	56.24	62.49	50.92	60.77	66.64	47.96	65.30	74.36

527 4 Discussion

528 In this paper, we introduce the novel **GO** annotation method with Transformer (GOAT).
 529 We show that for predicting protein annotations, our Transformer architecture in GOAT
 530 is better than the convolutional neural network in DeepGO. We then provide GOAT three
 531 types of extra features: Domain information, 3D-structure and PPI network data. These

532 features further increase the accuracy of GOAT, but PPI network information has the most
533 impact.

534 Previous software MetaGO of Zhang *et al.* [18] has also combined sequence data,
535 3D-structure and PPI network information to annotate GO labels. We emphasize that
536 in MetaGO, each type of metadata is used to build its own classifier, and then these
537 independent classifiers are then combined to produce the final prediction for a GO label
538 and an input sequence. For example, Zhang *et al.* [18] built their MetaGO_{BLAST} as an
539 independent unit from the their two classifiers that uses PPI network and 3D-structure
540 data. The reason for their strategy is that BLAST algorithm, which is similar to Smith-
541 Waterman, does not need the interacting partners and 3D-structure of the input sequences
542 [2]. Unlike BLAST-based methods, DeepGO and GOAT can jointly analyze the amino acid
543 sequences, PPI network and 3D-structure information. In the future work, we wish to
544 integrate components of MetaGO into GOAT, and vice versa.

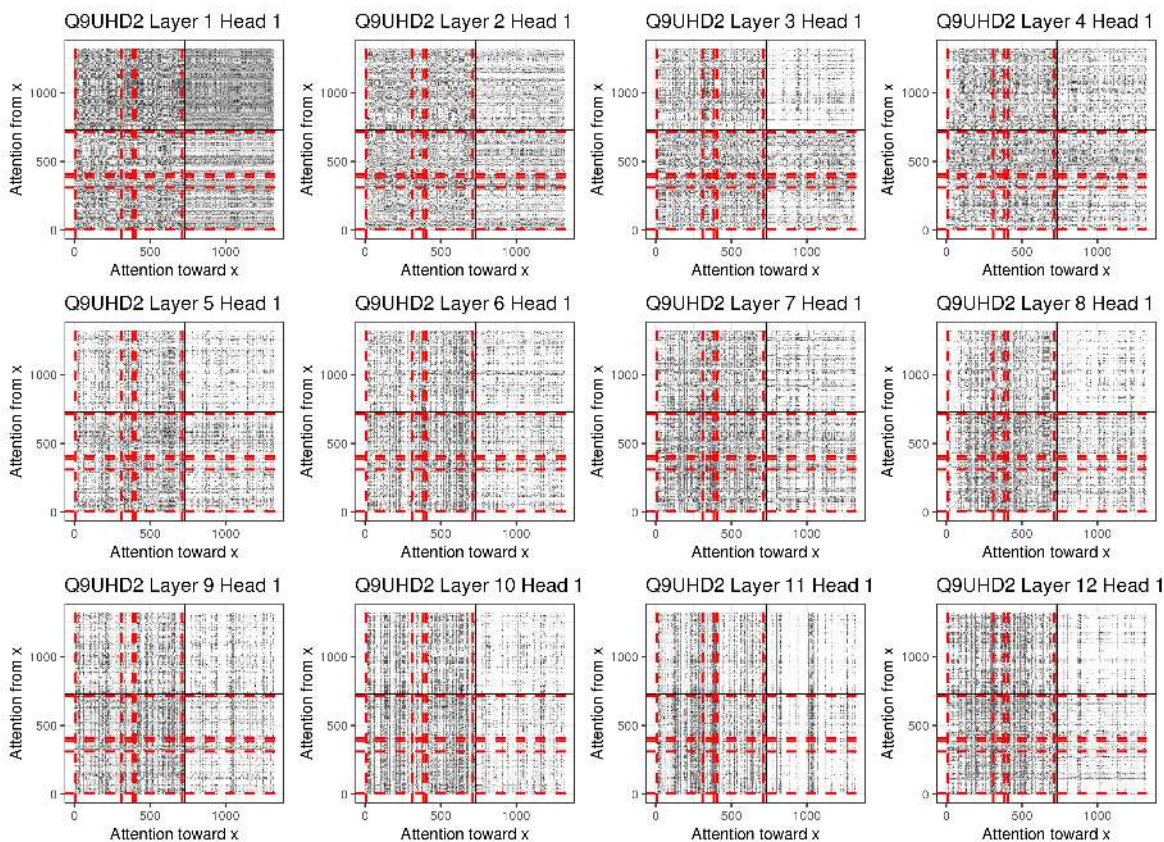
545 We discuss a key property of our Transformer in the context of GO embeddings.
546 This Transformer learns the co-occurrences among the labels; for example, the last layer
547 in Transformer returns comparable vectors for a child and parent GO label (Fig. 1). GO
548 embeddings produced by our Transformer are not equivalent to the embeddings produced
549 by factorizing the co-occurrence matrix of GO labels, because GO embeddings from our
550 Transformer are also affected by information from the amino acid sequence (Fig. 2). For
551 our future work, we will integrate embedding learned from co-occurrence frequencies
552 into our Transformer framework.

553 We outline a two key limitations of our adaptation of Transformer. First, in this paper, to
554 make our software GOAT accessible to many users, we have reduced the standard number
555 of parameters that Transformer often assume in other machine learning applications; for
556 example, Rives *et al.* [12] trained a language model on protein sequences using a 36-layer
557 Transformer. We expect that our GO annotation accuracy to increase if we train our model
558 with more parameters and on more samples from the Uniprot database.

559 Second, we do not pre-train our Transformer. For example, before predicting GO
560 labels, Transformer can be trained only on protein sequences with the following objective.
561 We can remove amino acids from a sequence, and then use Transformer to retrieve these
562 missing amino acids. Pre-training helps the parameters in Transformer to converge better
563 for the latter tasks; however pre-training requires a lot of data, for example Rives *et al.* [12]
564 pre-trained their model on 250 million sequences. For our future work, we will consider
565 training a large-scale Transformer model to predict GO labels for protein sequences.

566 5 Appendix

Figure 3: Heatmap of the attention values α_{jk} in each layer. Motifs of the sequences are not explicitly given as inputs to this Transformer model.



567 References

- 568 [1] Alshahrani, M., Khan, M.A., Maddouri, O., Kinjo, A.R., Queralt-Rosinach, N. and Hoehndorf, R. (2017). Neuro-symbolic representation learning on biological knowledge graphs. *Bioinformatics*, **33**(17), 2723–2730.
- 569
- 570
- 571 [2] Altschul, S.F., Gish, W., Miller, W., Myers, E.W. and Lipman, D.J. (1990). Basic local alignment search tool. *Journal of Molecular Biology*, **215**(3), 403–410.
- 572
- 573 [3] Bepler, T. and Berger, B. (2019). Learning protein sequence embeddings using information from structure. *arXiv preprint arXiv:1902.08661*.
- 574
- 575 [4] Chen, M., Ju, C.J.T., Zhou, G., Chen, X., Zhang, T., Chang, K.W. et al (2019). Multifaceted protein–protein interaction prediction based on siamese residual rnn. *Bioinformatics*, **35**(14), i305–i314.
- 576
- 577
- 578 [5] De Castro, E., Sigrist, C.J., Gattiker, A., Bulliard, V., Langendijk-Genevaux, P.S., Gasteiger, E. et al (2006). Scanprosite: detection of prosite signature matches and prorule-associated functional and structural residues in proteins. *Nucleic acids research*, **34**(suppl_2), W362–W365.
- 579
- 580

- 581 [6] Duong, D., Uppunda, A., Ju, C., Zhang, J., Chen, M., Eskin, E. et al (2019). Evaluating
582 representations for gene ontology terms.
- 583 [7] Dworkin, J. (2015). Ser/thr phosphorylation as a regulatory mechanism in bacteria. Current
584 opinion in microbiology, **24**, 47–52.
- 585 [8] Kulmanov, M., Khan, M.A. and Hoehndorf, R. (2017). Deepgo: predicting protein functions
586 from sequence and interactions using a deep ontology-aware classifier. Bioinformatics, **34**(4),
587 660–668.
- 588 [9] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J. (2013). Distributed represen-
589 tations of words and phrases and their compositionality. In Advances in neural information
590 processing systems, pages 3111–3119.
- 591 [10] Obozinski, G., Lanckriet, G., Grant, C., Jordan, M.I. and Noble, W.S. (2008). Consistent
592 probabilistic outputs for protein function prediction. Genome Biology, **9**(1), S6.
- 593 [11] Profiti, G., Martelli, P.L. and Casadio, R. (2017). The bologna annotation resource (bar 3.0):
594 improving protein functional annotation. Nucleic acids research, **45**(W1), W285–W290.
- 595 [12] Rives, A., Goyal, S., Meier, J., Guo, D., Ott, M., Zitnick, C.L. et al (2019). Biological structure
596 and function emerge from scaling unsupervised learning to 250 million protein sequences.
597 bioRxiv, page 622803.
- 598 [13] Schwartzberg, P.L. (1998). The many faces of src: multiple functions of a prototypical tyrosine
599 kinase. Oncogene, **17**(11), 1463.
- 600 [14] Sigrist, C.J., De Castro, E., Cerutti, L., Cuche, B.A., Hulo, N., Bridge, A. et al (2012). New and
601 continuing developments at prosite. Nucleic acids research, **41**(D1), D344–D347.
- 602 [15] Szklarczyk, D., Gable, A.L., Lyon, D., Junge, A., Wyder, S., Huerta-Cepas, J. et al (2019).
603 String v11: protein–protein association networks with increased coverage, supporting functional
604 discovery in genome-wide experimental datasets. Nucleic acids research, **47**(D1), D607–D613.
- 605 [16] The UniProt Consortium (2018). UniProt: the universal protein knowledgebase. Nucleic
606 Acids Research, **46**(5), 2699–2699.
- 607 [17] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N. et al (2017). Attention
608 is all you need. In Advances in neural information processing systems, pages 5998–6008.
- 609 [18] Zhang, C., Zheng, W., Freddolino, P.L. and Zhang, Y. (2018). Metago: Predicting gene
610 ontology of non-homologous proteins through low-resolution protein structure prediction and
611 protein–protein network mapping. Journal of molecular biology, **430**(15), 2256–2265.
- 612 [19] Zhou, J. and Troyanskaya, O.G. (2015). Predicting effects of noncoding variants with deep
613 learning–based sequence model. Nature methods, **12**(10), 931–934.