Bachelor Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Automated Protein Annotation with Integration of Gene Ontology Inter-Relationships

Karolína Orendáčová

Supervisor: doc. Ing. Jiří Kléma, Ph.D.
January 2025

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Orendáčová Karolína**	Personal ID number:	**492083**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Open Informatics**		
Specialisation:	**Artificial Intelligence and Computer Science**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Automated Protein Annotation with Integration of Gene Ontology Inter-Relationships

Bachelor’s thesis title in Czech:

Automatická anotace proteinů s využitím hierarchických vztahů genové ontologie

Guidelines:

Automated protein function prediction involves inferring a protein's function from its known sequence. This function is typically described by a list of terms sourced from a predefined gene ontology, organized hierarchically. Predicting protein function entails making binary decisions for each term, determining whether it is applicable to the given sequence. The thesis will primarily explore the application of deep transfer learning, leveraging both protein-level information and the inter-relationships among annotations.
Requirements:
1. Learn about deep learning and transfer learning.
2. Get acquainted with the current state of the art in automated protein function prediction with special emphasis on recent deep learning tools.
3. Carry out a literature search for methods ad 1 and 2.
4. Design your own algorithm / modification of an existing algorithm for automatic prediction of protein function with deep transfer learning.
5. Compare your solution with the basic benchmarks (BLAST + kNN, priors), or with the methods discussed in the search with available implementation, use the traditional measures of evaluating the quality of classifiers (precision, recall, F1).

Bibliography / sources:

[1] Zhao, Yingwen, et al. "Protein Function Prediction with Functional and Topological Knowledge of Gene Ontology." IEEE Transactions on NanoBioscience (2023).
[2] Makrodimitris, S. et al. "Improving protein function prediction using protein sequence and GO-term similarities." Bioinformatics 35.7 (2019): 1116-1124.
[3] Dhanuka, Richa, Jyoti Prakash Singh, and Anushree Tripathi. "A Comprehensive Survey of Deep Learning Techniques in Protein Function Prediction." IEEE/ACM Transactions on Computational Biology and Bioinformatics (2023).
[4] Nauman, Mohammad, et al. "Beyond homology transfer: Deep learning for automated annotation of proteins." Journal of Grid Computing 17 (2019): 225-237.

Name and workplace of bachelor’s thesis supervisor:

doc. Ing. Jiří Kléma, Ph.D.   Intelligent Data Analysis FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **09.02.2024**   Deadline for bachelor thesis submission: **07.01.2025**

Assignment valid until: **21.09.2025**

doc. Ing. Jiří Kléma, Ph.D.	prof. Dr. Ing. Jan Kybic	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.

Date of assignment receipt	Student’s signature

Acknowledgements

I would like to express my sincere gratitude to my supervisor, doc. Ing. Jiří Kléma, Ph.D., whose patience, guidance, and continuous support throughout this journey have been invaluable. I am also incredibly thankful to my family and friends for their constant support, love, and encouragement.

Declaration

Abstract

Automated protein function prediction is essential for efficiently annotating large-scale genomic data. This thesis proposes a novel approach, which integrates a convolutional neural network with transfer learning to assign Gene Ontology (GO) terms to protein sequences. The convolutional neural network is designed to process GO terms arranged in GO graph paths individually according to their hierarchical distance from the root term. The hierarchical relationships between GO terms are leveraged to reduce the dataset at every level, thereby significantly streamlining the training process. To assess the efficacy of the proposed method, it is benchmarked against a range of existing approaches for automated protein function annotation.

Keywords: protein annotation, deep learning, transfer learning

Supervisor: doc. Ing. Jiří Kléma, Ph.D.

Abstrakt

Automatizace predikce funkce proteinů je nezbytná pro efektivní anotaci rozsáhlých genomických dat. Tato práce představuje nový přístup k propojení konvoluční neuronové sítě s transferovým učením za účelem přiřazení termů Genové Ontologie (GO) sekvencím proteinů. GO termy jsou uspořádány do cest v grafu GO podle jejich hierarchické vzdálenosti od kořene grafu, a konvoluční síť je poté zpracovává jeden po druhém. Hierarchické vztahy mezi jednotlivými GO termy jsou na každé úrovni využity k významné redukci datasetu, což vede k významnému časovému zefektivnění procesu trénování. Navržená metoda je porovnána s řadou stávajících metod pro automatickou anotaci funkce proteinů.

Klíčová slova: anotace proteinů, hluboké učení, transferové učení

Překlad názvu: Automatická anotace proteinů s využitím hierarchických vztahů genové ontologie

Contents

Figures

Tables

Chapter 1

Introduction

Proteins are essential molecules composed of amino acid chains that play a variety of roles in living organisms. These molecules are responsible for nearly all biological processes and have diverse functions, including acting as enzymes, structural components, signaling molecules, and transporters. Identifying protein functions is imperative for the advancement of scientific understanding and has far-reaching implications in many fields, including drug discovery, disease diagnostics, and biomedical research. Traditionally, the functional attributes of proteins have been determined via in vitro experiments, a process that is both time-consuming and financially taxing. According to data from the UniProt Consortium [3], only a fraction of all known proteins have had their functions experimentally verified. Given the increasing volume of genomic data, there is a growing need for computational approaches capable of efficiently predicting protein functions on a large scale.

Automated protein function prediction (AFP) [4] refers to the algorithmic assignment of functional annotations, typically Gene Ontology (GO) terms, to proteins. A variety of approaches have been developed to address this challenge, each leveraging different aspects of protein data. Traditional sequence-similarity-based methods, such as BLAST [5], identify proteins with similar sequences to predict functions, under the assumption that proteins sharing significant sequence similarity are likely to have similar functions. Hierarchical methods [6] seek to enhance prediction accuracy through the utilization of the hierarchical structure of ontologies, such as Gene Ontology. In recent years, deep learning techniques [7, 8, 9] have significantly broadened the horizons of automated function prediction due to their ability to model complex, non-linear relationships within biological data. While traditional machine learning relies on the assignments of weights to hand-crafted features, deep learning methods work directly with raw input data, extracting features and performing classification in one integrated process. Deep learning is often complemented by the application of transfer learning, a machine learning technique in which a model developed for one task is adapted for a different, yet related, task. In the context of AFP, transfer learning enables models trained on large datasets to be fine-tuned for specific tasks or datasets with limited labeled data. This is particularly valuable in addressing the long-tail distribution of GO terms [10]: the majority of GO terms (5323) annotate fewer than 2000 proteins, compared to just 459 GO terms with larger datasets.

While protein function prediction is conventionally treated as a multi-label classification task, this thesis explores a novel approach that combines convolutional neural network (CNN) with transfer learning, leveraging the hierarchical structure of Gene Ontology. GO terms are organized into paths based on their hierarchical relationships within the GO graph, with root term on level 0. This structure enables the utilization of knowledge acquired from training on levels closer to the root to inform predictions on deeper levels, where the number of annotated proteins becomes increasingly limited. The CNN model is designed to process terms one at a time in the order in which they occur within the path. The hierarchical structure inherent in the GO graph leads to a cascading effect, wherein proteins annotated by a child term are also annotated by all ancestors of this term. At each level, a set of proteins occurs that are "annotated" by the parent term but "not annotated" by the child term. Additionally, once a protein is "not annotated" by a term in a path, it will not be annotated by any terms on hierarchically deeper levels of the path. This enables the efficient reduction of the dataset by excluding proteins that are "not annotated" at the parent level, thereby streamlining the training process for deeper levels. Furthermore, data augmentation and truncation methods are employed to address the issue of data imbalance. The proposed approach is evaluated under different configurations and compared against several benchmark methods.

Chapter 2
Theoretical Background

This chapter introduces the theoretical concepts and methods relevant to automated protein function prediction. It discusses various approaches to AFP, including the traditional sequence similarity-based and hierarchical methods, as well as the recently popular deep learning models. Additionally, it provides an overview of key concepts and techniques necessary for the thesis, such as deep learning, transfer learning, cross-validation, and data augmentation techniques.

2.1 Automated Protein Function Prediction

Automated protein function prediction (AFP) is a computational method that assigns functional annotations, typically Gene Ontology (GO) terms, to proteins or genes with unknown functions based on the analysis of proteins or genes with known functions [4]. In order to evaluate the performance of AFP algorithms, the Critical Assessment of Functional Annotation (CAFA) challenges have been designed. This experiment assesses the algorithms on a set of proteins with newly assigned GO annotations [11].

2.2 Gene Ontology

The Gene Ontology (GO) resource stands as the most widely used database for gene function [10]. It provides three sub-ontologies for describing genes across various domains of molecular biology: Molecular Function (MF), Biological Process (BP), and Cellular Component (CC), which can be represented as directed acyclic graphs with GO terms as nodes and relations between them as edges [12].

A number of relations between GO terms have been defined, such as *is a*, *part of* and *regulates*. However, for the purposes of this text, only the *is a* relation is considered. *Is a* relation forms the fundamental structure of GO. When we say that A is a B, we imply that node A is a subtype of node B. For instance, *small molecule binding* is a *binding*.

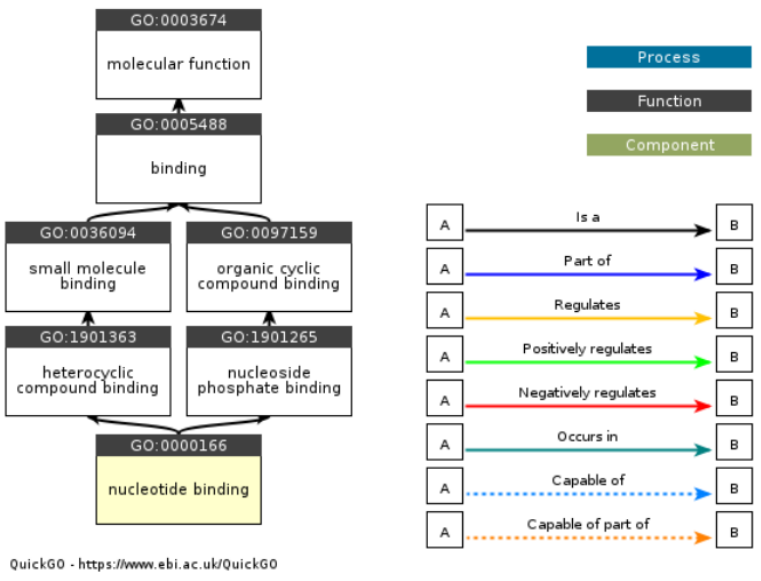

Figure 2.1: Hierarchy of GO Term GO:0000166 [1]

2.3 Deep Learning

Deep learning [13], a subset of machine learning, involves training multi-layer neural networks to model complex patterns in data. This approach has revolutionized numerous fields by enabling systems to learn representations directly from raw inputs, leading to significant advancements in tasks such as image and speech recognition, natural language processing, and more. Common types of neural networks used in deep learning include:

- **Feedforward Neural Networks** (FNNs): The simplest type, where information moves in one direction from input to output.
- **Convolutional Neural Networks** (CNNs): Specialized for processing grid-like data, such as images, by applying convolutional filters.
- **Recurrent Neural Networks** (RNNs): Designed for sequential data, allowing information to persist over time.
- **Long Short-Term Memory Networks** (LSTMs): A type of RNN that can learn long-term dependencies.

2.3.1 Convolutional Neural Network

Convolutional Neural Networks [14] are a class of deep learning models primarily designed to process data with grid-like topologies, such as images. However, they have been effectively adapted for sequence processing tasks by employing one-dimensional (1D) convolutions. In this context, filters (or kernels) move over the input sequence, performing element-wise multiplication and summation to extract important features, such as local patterns and dependencies. Typically, 1D CNNs consist of several convolutional layers, followed by activation functions such as ReLU (Rectified Linear Unit) to introduce non-linearity. After the convolutional layers, pooling layers are often applied to reduce the dimensionality of the feature maps, preserving only the most important information while decreasing computational complexity. Finally, fully connected layers integrate the extracted features and output predictions, such as classifications or regression values. This approach makes CNNs suitable for sequence classification tasks, including protein function prediction.

2.4 Transfer Learning

Transfer learning is a machine learning approach where a model trained on one task is repurposed for a different, but related task [15][16]. By leveraging knowledge gained from the original task, transfer learning allows models to perform well even with limited data in the new task, reducing the need for extensive retraining. This approach is particularly beneficial in fields with limited labeled data, as it enables the reuse of learned features from a pre-trained model.

Key techniques in transfer learning include freezing layers and fine-tuning. Freezing layers involves keeping certain layers of the model (typically the earlier ones) unchanged during training, as they often capture general features that are useful across tasks. Fine-tuning, by contrast, adapts the model to the new task by modifying specific layers or training the model on a more focused dataset. This process can involve updating the entire model or only adjusting task-specific layers, depending on the nature of the problem.

2.5 K-Fold Cross Validation

The K-fold cross-validation [17] is a widely used technique for the evaluation of machine learning models, which seeks to maximize the utilization of available data. In this method, the dataset is divided into K equally sized subsets, or folds. The model is trained on K-*1* folds and tested on the remaining fold, a process which is repeated K times, with each fold serving as the test set exactly once. This process ensures that every data point is utilized for both training and validation, thereby providing a comprehensive evaluation. The results from each iteration are then averaged to produce a single performance metric, thereby reducing variability and mitigating the risk of overfitting. K-fold cross-validation is particularly valuable when dealing with limited data, as it allows the model to be trained and tested on multiple splits, ensuring robust and reliable assessments of its generalization capabilities.

2.6 Data Augmentation

Data augmentation [18] is a technique employed to artificially increase the size and diversity of a dataset by creating modified versions of the existing data. In machine learning, particularly in tasks where labeled data is limited, this approach enhances the model's generalization ability and robustness. Augmentation can involve various transformations, such as rotations, translations, scaling, or flipping for images, or substitutions, insertions, and deletions for sequences. In the context of protein function prediction, augmentation may include the following: amino acid sequence shuffling, the introduction of mutations, or the generation of synthetic variants while preserving the biological relevance of the data. By exposing the model to diverse representations of the same underlying information, data augmentation reduces the risk of overfitting and improves performance on unseen data.

This thesis incorporates the following data augmentation techniques:

- **Random Insertion** introduces random elements, in this case amino acids, into arbitrary positions within the sequence. The inserted elements are selected from the set of possible sequence components, ensuring compatibility with the original data format.
- **Random Deletion** removes randomly selected elements from the sequence, resulting in shorter sequences. The deletions can occur at any position within the sequence, thereby allowing the sequence to mimic potential gaps or missing data points.
- **Random Substitution** replaces specific elements in the sequence with other valid alternatives. The substituted values are chosen from the same set as the original components, maintaining the integrity of the sequence format while introducing variability.
- **Local Shuffle** rearranges the order of elements within a small, predefined window in the sequence. This process preserves the overall content while disrupting the precise arrangement, thus simulating localized changes in sequence order.

2.7 AFP Methods

Automated protein function prediction (AFP) methods aim to algorithmically assign functional annotations to proteins based on various features such as amino acid sequences or structural information. The methods include sequence similarity-based approaches, hierarchical classification methods, and more advanced techniques incorporating machine learning. In recent years, deep learning models have become increasingly prevalent.

2.7.1 Sequence Similarity-based AFP Methods

Sequence similarity-based methods for protein annotation rely on the assumption that proteins with similar sequences are likely to have similar functions. These methods compare the sequence of a target protein with sequences of known proteins to infer its function based on the annotations of the most similar sequences.

BLAST-kNN

The Basic Local Alignment Search Tool (BLAST) algorithm is a tool for comparing biological sequences, such as nucleic acid or protein sequences, against large databases. BLAST is primarily used to find similar or homologous sequences within these databases. It does this by identifying short, significant matches between the query sequence and sequences stored in the database [5].

BLAST starts by identifying short, exact matches called "seeds" between the query sequence and sequences in the database. These seeds are extended to form local alignments, with alignment scores calculated based on sequence similarity. BLAST assigns scores to aligned regions using a scoring matrix (e.g., BLOSUM62) [19]. An e-value (expect-value) is then calculated to estimate the change in the two alignments that would occur by chance, with lower e-values indicating more significant matches. BLAST returns a list of database sequences with significant similarities to the query, ranked by their e-values or similarity scores [20].

The k-Nearest Neighbors (kNN) algorithm is a supervised machine learning method commonly used to assigning class labels to data points based on the majority class among the k closest training examples in a feature space [21].

During the training phase, labeled examples with features and class labels are collected and stored. Unlike traditional models, kNN retains the entire training dataset. In the prediction phase, when a new, unlabeled data point is provided, the algorithm calculates distances to all training data points. It identifies the k training data points with the shortest distances, known as the "k nearest neighbors". For classification, the algorithm assigns a class label to the new data point based on the majority class among its k nearest neighbors, typically using a majority rule mechanism [22].

The performance and behavior of the algorithm is significantly affected by the value of k. Too low a value of k can lead to overfitting, where the classifier is influenced by noisy or random fluctuations in the data, while too high a value of k can lead to underfitting, where the model becomes too generalized [21].

The BLAST-kNN algorithm initially utilizes BLAST with a specified cut-off e-value to identify all proteins in the training set. This generates a set, Z_i, of proteins that are homologous to protein p_i. The score, $S_B(p_i, GO_j)$, between protein p_i and GO term GO_j is then calculated using the following equation:

$$S_B(p_i, GO_j) = \frac{\sum_{p_k \in Z_i} I(p_k, GO_j) \times B(p_i, p_k)}{\sum_{p_k \in S_{p_i}} B(p_i, p_k)} \tag{2.1}$$

where $B(p_i, p_k)$ is the similarity score between p_i and p_k by BLAST and $I(p_k, GO_j)$ is a binary indicator: 1 if GO_j belongs to protein p_k; otherwise 0 [23].

Motif-based Protein Function Classifiers

Although there is significant evidence that structure is maintained among homologous proteins (i.e., those encoded by genes that have evolved from a common ancestor), it is important to note that there is a strong correlation between sequence similarity and structure. However, the evidence for conservation of function is less clear [24]. On the other hand, motif methods focus on short, highly conserved regions of protein often corresponding to functional regions of a protein. The presence of these protein motifs often provides crucial insights into a protein's function despite the lack of global similarity to known proteins [25].

Motif-based methods can be categorized into two classes: the first class extracts conserved patterns from sets of functionally related sequences through local multiple sequence alignment, while the second class adopts a combinatorial approach, constructing a motif dictionary from a given set of sequences with disregard to any functional family memberships of the sequences. Because of their capacity to detect weaker sequence similarities, pattern-based searches are often more sensitive than sequence database searches [24].

2.7.2 Hiearchical Classification AFP Methods

Hierarchical classification is an organizational approach to sort classes to be predicted into a hierarchical structure, such as trees or DAGs with nodes representing the classes. In the context of protein function prediction, the classes to be predicted (protein functions) are naturally organized into class hierarchies [6]. One of these hierarchies is the Gene Ontology.

True Path Rule

The core principle of the True Path Rule (TPR) algorithm is that annotations for a class within the hierarchy are automatically propagated to their ancestral nodes, whereas unannotated nodes cannot receive annotations for their descendant nodes. In other words, when a gene is annotated with a specific functional term (or functional class), it is automatically recursively annotated with all the parent classes and their ancestors. In case of GO, if a gene g is not annotated to a class c gene g can receive annotations for some of the descendants of c if there exists at least one ancestor within the descendant of c to which gene g is already annotated [26].

2.7.3 AFP Methods Using Transfer Learning

In machine learning, transfer learning is a method that utilizes knowledge acquired from a specific task to improve performance on a related task [16]. Transfer learning has become increasingly popular in recent years due to its ability to reduce the amount of data required for training models.

This section describes a selection of methods that employ TL.

DeepSeq

The DeepSeq algorithm [7] is primarily based on a convolutional neural network. It uses embedding instead of the commonly used one-hot encoding technique. This method places similar symbols closer to each other in an x-dimensional space. Each amino acid is represented as a 23-dimensional vector. The embedding layer is part of the model, so the similarity is learned automatically from the data, which is characteristic of end-to-end models.

The output then passes through several convolutional layers. Due to the large number of nodes, average max pooling is used instead of flattening layer, which significantly reduces the number of parameters. The covariate shift problem is solved by the batch normalization layer.

DeepSeq outperformed BLAST with an F1 score of 0.71132, while BLAST achieved an F1 score of 0.53915.

SeqVec

In SeqVec [27], unsupervised protein embeddings are utilized. For each sequence of length L, amino acid features are extracted using an unsupervised ELMo-based language model that outputs a feature vector for each amino acid. Additionally, protein-level features are computed by averaging the features over all L amino acids, resulting in a fixed-length vector representation for each protein. One-hot encoding, amino acid frequency-based protein-level representation and k-mer count-based protein-level representation are used to compare ELMo with simpler representations.

The study tested k-nearest neighbors (k-NN), logistic regression (LR), and a multilayer perceptron (MLP) with one hidden layer, as well as several convolutional neural networks. The models that utilize pretrained embeddings outperform those that use other features on all evaluation metrics.

DeeProtGO

DeeProtGO [8] utilizes a feed-forward deep neural network. It integrates information from multiple sources, including features derived from protein sequence and functional annotations. The model has an encoding sub-network for each data type, allowing it to learn specific features from each input. Each one consists of two fully connected layers with exponential linear units as activation functions, batch normalization and dropout for model regularization.

The output - set of learned features - is then concatenated into a vector, which is used as the input to the classification sub-network.

PFmulDL

PFmulDL [9] represents a significant leap in AFP methodology by integrating a recurrent neural network with a multi-kernel convolutional neural network for the first time. This approach has enabled the annotation of a remarkable number of protein families compared to existing methods, while demonstrating substantial improvements in prediction accuracy for the rare classes, without compromising performance for the major classes.

The Gene Ontology (GO) terms are classified into three categories based on the average number of proteins annotated by them: major ($average > 1000$), medium ($1000 > average > 500$), and rare ($500 > average$). When predicting a protein's association with a specific GO term, it is assigned to all parent terms of that particular GO term.

Protein sequences are encoded using only the first 2000 amino acids. The protein sequence is encoded using a one-hot strategy, where each amino acid is represented by a 21-dimensional vector, resulting in a 2000×21 matrix for each sequence. The encoded protein sequences serve as input to construct a model based on a multi-kernel CNN. This model is then further refined through a pre-training process. The CNN's dense layer is then fed into an RNN, resulting in a fully connected layer. The dimension of this layer is reduced by setting the number of neural units to number of GO classes to enable their annotation.

2.7.4 Evolutionary Scale Modeling

The ESM (Evolutionary Scale Modeling) family of models, introduced by [28], represents a significant advancement in the field of computational biology, leveraging deep learning to analyze protein sequences. These models, based on the Transformer architecture, are trained on large-scale protein sequence datasets to learn meaningful embeddings that capture structural and functional characteristics. By treating protein sequences as language sequences, ESM models excel in a variety of biological tasks, including structural prediction, mutational effect estimation, and functional annotation.

ESM-2 builds on its predecessors with improved accuracy and scalability, making it one of the most powerful tools for protein function prediction. In functional annotation tasks, ESM-2 enables zero-shot predictions by generating embeddings that encode sequence-level and evolutionary-scale information, allowing researchers to infer protein functions without requiring labeled training data for each specific function.

Chapter 3

Proposed Solution and Implementation

The proposed AFP method integrates a convolutional neural network with transfer learning, leveraging the hierarchical structure of Gene Ontology. GO terms are organized into paths based on the GO graph, where each level of the path is the distance from the sub-ontology root term. The model is designed to process the data path by path, training on one term a at a time in the hierarchical order. Due to the hierarchical relationships in the GO graph, proteins annotated by a term inherit the annotations of all of this term's ancestors. In addition, once protein is "not annotated" by a term, it is also "not annotated" by all of this term's descendants. This knowledge is utilized to significantly reduce the dataset by excluding proteins "not annotated" at the immediate parent level. To address the issue of data imbalance, a variety of data augmentation and truncation methods were implemented.

3.1 Data Representation

The majority of proteins listed in the *Swiss-Prot* database have a sequence length of less than 2000 amino acids [3]. For this reason, the maximum sequence length is set at 2000 amino acids. To handle sequences that do not meet this length requirement, post-padding with zeroes is implemented [9][7]. Protein sequences are converted into a numerical representation [7] and annotated with a binary representation of GO terms, in which a value of 1 indicates the protein in question has been annotated by the GO term on this index, whereas a value of 0 indicates the absence of such annotation.

3.2 Data Preprocessing

To effectively implement the hierarchical structure required, several preprocessing steps are performed. Firstly, the mappings in the OBO file are updated by replacing *alt ids* with their corresponding *ids*, substituting obsolete terms with valid ones, and ensuring that each term includes all its ancestors with an *is a* relationship. This step enables the creation of directed acyclic graphs (DAGs) for each ontology.

All potential paths for every term to be considered are generated. Given that the hierarchy is a DAG, it is possible for parents to have children on different levels. Consequently, the terms are topologically sorted in paths using Kahn's algorithm [29], ensuring that no child term comes before any of its parent terms.

The dataset is divided into training, validation and test sets according to the following process. Initially, the dataset is split into a "training" and a test set using K-fold cross-validation, with K set at 5. The "training" set is then sub-divided into a training and a validation set by randomly selecting 10% of the training set as the validation set. This approach is adopted to ensure the consistency of the test set, thereby facilitating more reliable comparisons. The utilization of K-fold cross-validation ensures the systematic assessment of the entire dataset.

3.3 Convolutional Neural Network

Data in the form of sequences of length 2000 are fed to the network through the input layer.

Embedding layer encodes similar symbols by placing them closer together in an *output*-dimensional space, where *output* dimension was chosen to be equal to the number of unique amino acids [7], same as the *input* dimension.

Convolutional layers with 256 and 128 filters of length 8, respectively, are applied to capture local patterns and features within the sequences. To address the vanishing gradient problem, both layers use the rectified linear unit (ReLU) activation function [30]. Each Convolutional layer is followed by a Dropout layer with a rate of 0.3 to avoid overfitting. After each Dropout layer, a Max-Pooling layer is added to downsample the feature maps and reduce computational complexity.

Global Average Pooling layer calculates the average value for each feature over the entire sequence, reducing the spatial dimensions to a single value for each feature.

Batch Normalization layer normalizes the activations of the previous layer, aiding in training stability.

The output layer is a Dense layer with a single unit for binary classification using a sigmoid activation. The sigmoid activation function compresses the output values to a range between 0 and 1, providing a probability for binary classification.

The error is calculated using Binary Cross-Entropy (BCE) loss [31] and the optimization algorithm utilized is the Adam (Adaptive Moment Estimation) optimizer [32], which is frequently chosen [9][7][33] for its efficiency in handling sparse gradients and adapting the learning rate during training. Based on experiments, the learning rate for training without transfer learning is set to 1e-4. Early stopping is employed during training to prevent overfitting by monitoring the model's performance on a validation set.

3.4 Convolutional Neural Network with Transfer Learning

The Gene Ontology DAGs are constructed based on the *is a* relations between terms. The *is a* relation forms the foundational structure of Gene Ontology, indicating a hierarchical connection between terms. When we say "A is a B," it means that term A is a more specific subtype of term B. This structure ensures that when a protein is associated with a specific GO term, it inherits annotations not only from that term but also from all its ascendant terms, tracing back to the root node of Biological Process (BP), Molecular Function (MF), or Cellular Component (CC) [34].

The training process is executed in a path-by-path manner, with the model being trained on a single term at a time. The training starts with the immediate child of the root term and progresses hierarchically down the path. In the case of multiple parents for a term, one is selected randomly to ensure diversity in the training process. Model weights are always saved and then loaded in the model created for training the child term. Initializing the child term model with pretrained weights from the parent term enables the model to leverage previously learned features.

The inheritance of annotations from parent terms implies that after given sequence encounters the first term annotated by 0, all subsequent child terms along the path will be annotated 0. At each level, a set of proteins occurs that are annotated with 1 for the parent term but 0 for the current term. These "change points" highlight the most relevant areas for model refinement, allowing the network to focus on learning the differences in annotations across the hierarchy.

Leveraging this knowledge, the training dataset for a child term is reduced to include only sequences that are annotated by the parent term (parent term annotation is 1).

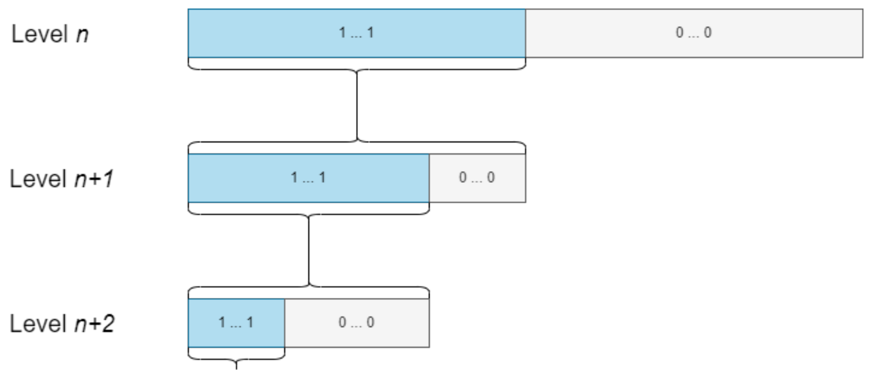

Figure 3.1: Dataset reduction while maintaining the original balance. Graph created using draw.io [2].

As demonstrated in 3.1, the aforementioned approach potentially generates significantly imbalanced datasets, which may adversely affect the results. To address the issue of imbalanced datasets, two approaches were implemented.

One of these techniques is data augmentation, which is employed as shown in 3.2 to artificially augment the size of the minority class by generating new, synthetic examples based on the existing data. The augmentation techniques employed in this project comprise random insertion, random deletion, random substitution, and local shuffle, applied in a randomized manner.

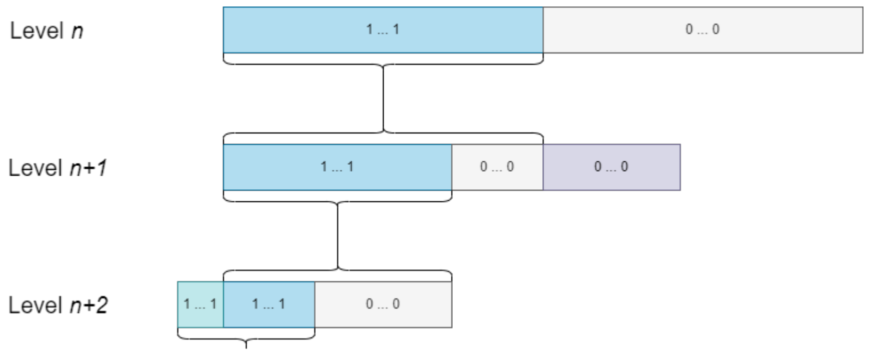

Figure 3.2: Dataset reduction combined with data augmentation. Created using draw.io [2].

Another strategy is data truncation, illustrated in 3.3, in which the majority class is randomly reduced to match the size of the minority class. This technique is designed to prevent the model from being biased toward the majority class by limiting the number of samples from the dominant class, thereby creating a more balanced training set.

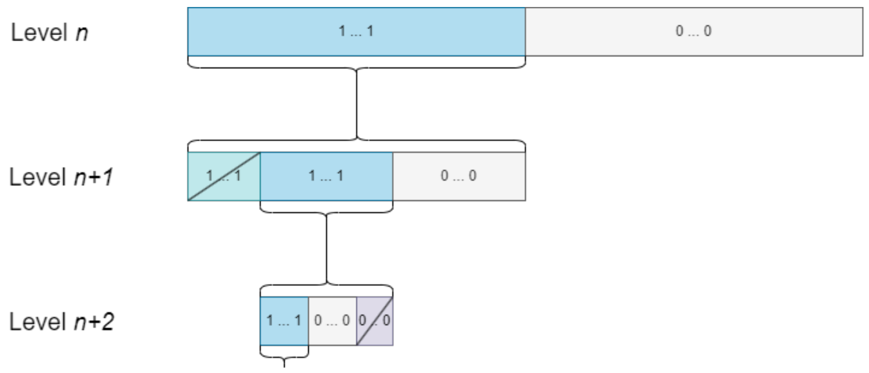

Figure 3.3: Dataset reduction combined with data truncation. Created using draw.io [2].

The optimizer used in this approach is the same as in the baseline model, Adam. Early stopping was also utilized to prevent overfitting by monitoring the model's performance on a validation set. If the validation performance stopped improving after a number of epochs, training was halted to ensure the model generalizes well without being excessively fitted to the training data. To identify the most effective learning rate for the transfer learning process, a range of values was tested: 1e-4, 1e-5, and 1e-6.

The model has been trained with all layers active, as well as with the top layers - specifically the embedding and first convolutional layer - frozen.

Freezing these layers has allowed the model to retain the general features learned during earlier training phases, while fine-tuning the deeper layers for the specific task. This approach has aimed to strike a balance between preserving valuable pre-trained knowledge and adapting the model to the new dataset more efficiently.

Chapter 4

Evaluation

Protein sequences and their GO term annotations used for training the neural network were obtained from CAFA3 training data [11], the go.obo file necessary for creating hierarchical relationships between terms was downloaded from the GO Consortium website [35].

The proposed method is benchmarked against several established approaches for automated protein function prediction, including BLAST-kNN, priors-based annotation, and ESM-2. BLAST-kNN, which uses sequence similarity and nearest-neighbor algorithms for protein prediction, provides a strong baseline for comparison due to its widespread use and robustness. The PRIORS algorithm is a simplistic approach where all targets are assigned the same annotations and the score of each term is calculated as the probability of that term occurring in database [34].

4.1 Evaluation Metrics

4.1.1 F_1-score

Precision and Recall

When classifying samples, there are four possible outcomes:

- true positive (TP), correctly classified as positive,
- false positive (FP), incorrectly classified as positive,
- true negative (TN), correctly classified as negative,
- false negative (FN), incorrectly classified as negative.

Precision measures how many of the items classified as positive are actually relevant or true positives. In other words, it assesses the accuracy of positive predictions. A high precision means that most of the positive predictions made by a model are correct.

$$precision = \frac{TP}{TP + FP} \tag{4.1}$$

Recall measures how many of the actual positive items were correctly classified as positive. It quantifies the ability of a model to identify all positive instances in the dataset. High recall indicates that the model can effectively find most of the positive instances.

$$recall = \frac{TP}{TP + FN} \tag{4.2}$$

F_1-score

The F_1 score is a metric used to measure the balance between precise and comprehensive predictions. Maximizing F_1 score means finding the ideal trade-off between accuracy and completeness in classification models.

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{4.3}$$

4.1.2 Area Under ROC Curve

A Receiver Operating Characteristic (ROC) curve demonstrates the trade-off between true positive rate (TPR) and false positive rate (FPR) at different decision thresholds [36].

$$TPR\ (Sensitivity) = \frac{TP}{TP + FN} \tag{4.4}$$

$$FPR\ (1 - Specificity) = \frac{FP}{FP + TN} \tag{4.5}$$

The Area Under the Receiver Operating Characteristic Curve (AUROC) is a widely used evaluation metric for classification problems, particularly in scenarios where the dataset is imbalanced. It indicates the ability of a model to differentiate between positive and negative classes at various classification thresholds. The AUROC value ranges from 0 to 1, with a value of 1 indicating perfect classification and 0.5 suggesting a random classifier.

One of the main advantages of AUROC over metrics like the F_1-score is its resistance to class imbalance. The F_1-score, which balances precision and recall, may be influenced by the distribution of positive and negative samples. In contrast, AUROC assesses model performance across all classification thresholds, offering a more comprehensive evaluation of the model's capacity to differentiate between classes, independent of their respective proportions.

4.2 CNN Evaluation

The training of the neural network was carried out on a NVIDIA GeForce RTX 4070 Ti Super GPU, using cuDNN for enhanced computational efficiency.

Due to computational constraints, only a subset of GO terms were selected to demonstrate the algorithm's performance. These terms were randomly chosen from those annotating a sufficient number of proteins, with the selection process considering the ontology type and level within the GO term hierarchy. For instance, terms hierarchically closer to root term were preferred to ensure that their descendant terms also annotated a sufficient amount of sequences. The number of annotations considered for selection varied depending on the ontology and the level within the hierarchy. This approach attempts to make the selected terms representative of the broader dataset while balancing computational efficiency.

The training terms were selected from levels up to level 6 of the GO term hierarchy, with the root term at level 0. For each term, all possible paths were generated, utilizing exclusively the *is a* relation within the GO ontology to maintain the integrity of each ontology and to avoid overlap with other ontologies. As a result, a total of 93 terms were included in the dataset: 29 from the Cellular Component ontology, 28 from the Biological Process ontology, and 36 from the Molecular Function ontology. The slight inconsistencies in the number of terms across the ontologies arose from the fact that each term generated a different number of paths, which led to the unequal distribution.

4.2.1 CNN without Transfer Learning

When training the CNN without transfer learning, a fixed learning rate of 1e-4 is employed throughout the process. Furthermore, no dataset reduction or filtering is applied, and all available data is utilized for training without modification. This setup establishes a baseline for the subsequent evaluation of the effects of transfer learning and data reduction techniques in follow-up experiments.

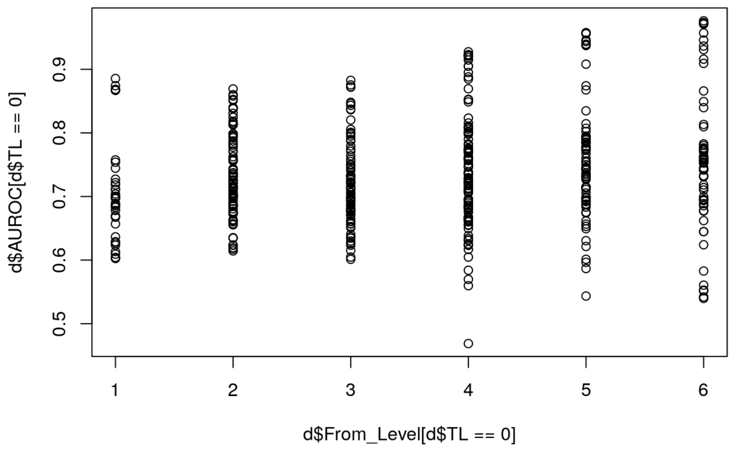

Figure 4.1: The impact of Level on AUROC.

To examine the impact of various factors on AUROC, a linear model was employed. The resulting analysis indicated that Level (distance of GO term from root term) had a small but significant positive effect on AUROC, suggesting that terms further in the path perform slightly better. Of the ontologies examined, the MF ontology displayed a statistically significant positive effect. Overall, the model accounted for approximately 13% of the variance in AUROC, suggesting that while some factors do influence the results, a significant portion remains unexplained.

The AUROC scores presented in Table 4.1 provide an evaluation of the neural network's performance across different levels of the GO hierarchy and for each ontology: Molecular Function (MF), Biological Process (BP), and Cellular Component (CC). The Level indicates the distance of the hierarchically highest GO terms from the sub-ontology root term. For instance when Level is 2, the AUROC is averaged from GO terms on levels 2 to 6; if Level is 6, the AUROC is averaged from GO terms on level 6, as it is the lowest level chosen to train on.

Level	MF	BP	CC
1	0.7794 ± 0.0803	0.6893 ± 0.0518	0.6938 ± 0.0529
2	0.7573 ± 0.0840	0.7127 ± 0.0536	0.7207 ± 0.0741
3	0.7692 ± 0.0984	0.7126 ± 0.0643	0.7121 ± 0.0826
4	0.7923 ± 0.1167	0.7295 ± 0.0386	0.7131 ± 0.1012
5	0.7864 ± 0.1240	0.7439 ± 0.0315	0.7125 ± 0.1157
6	0.7835 ± 0.1315	0.7464 ± 0.0286	0.7007 ± 0.1416

Table 4.1: Mean AUROC scores and their standard deviations for neural network without TL from various levels for each ontology.

The results indicate that the best performance overall is achieved on the Molecular Function ontology, with a maximum AUROC score of 0.7923 when training is initiated from level 4. The Biological Process and Cellular Component ontologies gradually improve with increasing depth in the hierarchy. However, it should be noted, that for the Molecular Function and Cellular Component ontologies, the standard deviation increases with deeper levels.

Level	MF	BP	CC
1	598 ± 165	329 ± 82	410 ± 95
2	611 ± 150	369 ± 84	453 ± 118
3	682 ± 110	370 ± 95	481 ± 100
4	711 ± 102	401 ± 91	533 ± 76
5	719 ± 105	420 ± 81	552 ± 62
6	743 ± 94	441 ± 95	592 ± 22

Table 4.2: Mean training times in seconds and their standard deviations for neural network without TL from various levels for each ontology.

As demonstrated in Table 4.2, training requires a significant amount of time per term in the context of non-transfer learning. These results will serve as a baseline for comparison with the training times achieved through the implementation of transfer learning.

4.2.2 CNN with Transfer Learning

Transfer learning, in the form of dataset reduction, was applied to the CNN as described in 3.4. Overall, transfer learning led to improvement in performance in 36% cases of use. Specifically, the improvement was 39% when applying data truncation in dataset reduction and 40% when using data augmentation. Despite the transfer learning application's inability to enhance prediction accuracy, the dataset reduction achieves a substantial reduction in training time per term, a crucial consideration when dealing with large datasets.

The following tables demonstrate the performance of the CNN combined with transfer learning. Transfer learning was applied experimentally at all possible levels, with Level referring to the specific level in the ontology from which the transfer learning process was initiated. Given the specific implementation of transfer learning, the first level at which the application of transfer is feasible is level 2. At this level, the dataset is reduced based on the annotations from level 1 and the pretrained weights from the parent at level 1 are utilized. The learning rates tested during these experiments were 1e-4, 1e-5, and 1e-6. Due to time constraints, not all possible combinations of levels and learning rates were explored.

Level	LR	MF	BP	CC
2	1e-4	0.7415 ± 0.1131	0.5608 ± 0.0805	0.7100 ± 0.1030
	1e-5	0.6467 ± 0.1267	0.5707 ± 0.0819	-
	1e-6	-	0.6175 ± 0.0579	-
3	1e-4	0.7431 ± 0.1191	0.5647 ± 0.0674	0.6799 ± 0.1115
	1e-5	0.7022 ± 0.1184	0.6013 ± 0.0625	-
	1e-6	-	0.5937 ± 0.0651	-
4	1e-4	0.7518 ± 0.1484	0.6065 ± 0.0662	0.7349 ± 0.0945
	1e-5	0.7723 ± 0.1290	0.6335 ± 0.0584	-
	1e-6	-	0.6351 ± 0.0455	-
5	1e-4	0.7502 ± 0.1422	0.6548 ± 0.0620	0.7332 ± 0.0844
	1e-5	0.7623 ± 0.1503	0.6903 ± 0.0666	-
	1e-6	-	0.7212 ± 0.0353	-
6	1e-4	0.7759 ± 0.1388	0.5368 ± 0.0864	0.7023 ± 0.0850
	1e-5	0.7903 ± 0.1330	0.5946 ± 0.0772	-
	1e-6	-	0.6559 ± 0.0793	-

Table 4.3: Mean AUROC scores and their standard deviations for neural network with TL from various levels and learning rates for each ontology on the original dataset.

Table 4.3, results of training with TL on dataset with preserved original data imbalance, indicates notable distinctions across the ontologies. For the

Molecular Function, training from higher levels achieves superior results with a higher learning rate (1e-4), while for deeper levels (level 4 and beyond), slowing the learning rate down to 1e-5 yields better results. When compared to the training without TL, the results are similar in terms of AUROC scores but demonstrate increased variability in performance, as evidenced by the standard deviation. In the Biological Process ontology, the results are consistently lower compared to other ontologies and training without TL. For the Cellular Component ontology, maximum efficacy is attained when training from mid-range levels (levels 4 and 5), with superior overall performance in comparison to the training without TL. The training with TL accomplishes comparable AUROC score to training without TL but with decreased standard deviation, indicating improved reliability in predictions.

Level	LR	MF	BP	CC
2	1e-4	34 ± 51	16 ± 28	60 ± 96
	1e-5	26 ± 40	17 ± 33	-
	1e-6	-	12 ± 16	-
3	1e-4	10 ± 13	44 ± 77	41 ± 65
	1e-5	11 ± 14	25 ± 46	-
	1e-6	-	47 ± 94	-
4	1e-4	5 ± 3	26 ± 59	32 ± 50
	1e-5	4 ± 3	17 ± 34	-
	1e-6	-	21 ± 43	-
5	1e-4	4 ± 3	6 ± 5	19 ± 37
	1e-5	4 ± 5	6 ± 5	-
	1e-6	-	7 ± 6	-
6	1e-4	3 ± 1	56 ± 28	2
	1e-5	3 ± 1	65 ± 36	-
	1e-6	-	61 ± 49	-

Table 4.4: Mean training times in seconds and their standard deviations for neural network with TL from various levels and learning rates for each ontology on the original dataset.

Applying transfer learning significantly reduces the training times per term across all cases, as seen in Table 4.4. The training time reduction is significant, with a minimum of approximately 5 minutes, with some instances showing reductions exceeding 10 minutes.

Level	LR	MF	BP	CC
2	1e-4	0.7163 ± 0.0838	0.6673 ± 0.0481	0.7227 ± 0.0958
	1e-5	0.6949 ± 0.1189	0.6524 ± 0.0452	-
	1e-6	0.6383 ± 0.1266	0.6150 ± 0.0719	-
3	1e-4	0.7490 ± 0.1131	0.6680 ± 0.0439	0.6987 ± 0.1063
	1e-5	0.7203 ± 0.1102	0.6665 ± 0.0430	-
	1e-6	0.7096 ± 0.1038	0.6019 ± 0.0745	-
4	1e-4	0.7809 ± 0.1172	0.6852 ± 0.0419	0.7048 ± 0.0895
	1e-5	0.7633 ± 0.1265	0.6892 ± 0.0398	-
	1e-6	0.7863 ± 0.1117	0.6578 ± 0.0575	-
5	1e-4	0.7590 ± 0.1381	0.6906 ± 0.0532	0.7231 ± 0.0843
	1e-5	0.7560 ± 0.1543	0.7008 ± 0.0361	-
	1e-6	0.7752 ± 0.1335	0.7280 ± 0.0385	-
6	1e-4	0.7779 ± 0.1331	0.6818 ± 0.0431	0.6842 ± 0.1159
	1e-5	0.7974 ± 0.1208	0.7111 ± 0.0470	-
	1e-6	0.7899 ± 0.1244	0.6844 ± 0.0737	-

Table 4.5: Mean AUROC scores and their standard deviations for neural network with TL from various levels and learning rates for each ontology on the augmented dataset.

The results from Table 4.5 imply that applying data augmentation when reducing dataset yields notable improvements for Biological Process ontology compared to the original dataset, with consistently low standard deviations; however, the results are still slightly worse than without TL. For Molecular Function ontology, results demonstrate higher AUROC scores at lower levels (level 4 and 6), where the CNN performs comparable to training without TL. However, slowing down the learning rate is inconsequential at these levels. For Cellular Component ontology, the performance is comparable to results without TL but generally slightly worse.

Level	LR	MF	BP	CC
2	1e-4	189 ± 83	119 ± 112	190 ± 151
	1e-5	109 ± 72	108 ± 130	-
	1e-6	115 ± 85	60 ± 63	-
3	1e-4	30 ± 33	103 ± 94	123 ± 156
	1e-5	25 ± 27	89 ± 84	-
	1e-6	35 ± 36	265 ± 237	-
4	1e-4	12 ± 26	66 ± 52	52 ± 108
	1e-5	8 ± 8	63 ± 46	-
	1e-6	8 ± 11	35 ± 54	-
5	1e-4	6 ± 4	36 ± 46	31 ± 59
	1e-5	5 ± 4	28 ± 18	-
	1e-6	4 ± 3	13 ± 13	-
6	1e-4	3 ± 1	21 ± 21	2
	1e-5	3 ± 1	21 ± 25	-
	1e-6	3 ± 1	86 ± 61	-

Table 4.6: Mean training times in seconds and their standard deviations for neural network with TL from various levels and learning rates for each ontology on the augmented dataset.

As demonstrated in Table 4.6, training times per GO term are also significantly reduced by employing TL on the augmented dataset, although the difference is less pronounced on higher levels. This finding aligns with expectations, as the size of the dataset decreases at a slower rate with the augmented dataset.

Level	LR	MF	BP	CC
	1e-4	0.7070 ± 0.1290	0.5624 ± 0.0838	0.6854 ± 0.1076
2	1e-5	0.6510 ± 0.1064	0.6145 ± 0.0728	0.6467 ± 0.0917
	1e-6	-	0.6418 ± 0.0653	-
	1e-4	0.7394 ± 0.1074	0.5860 ± 0.0722	0.6788 ± 0.1123
3	1e-5	0.7200 ± 0.1072	0.6133 ± 0.0627	0.6572 ± 0.1025
	1e-6	-	0.6415 ± 0.0608	-
	1e-4	0.7772 ± 0.1229	0.6264 ± 0.0679	0.7167 ± 0.0891
4	1e-5	0.7914 ± 0.1098	0.6582 ± 0.0503	0.6999 ± 0.0747
	1e-6	-	0.6787 ± 0.0389	-
	1e-4	0.7492 ± 0.1434	0.6729 ± 0.0489	0.7275 ± 0.0769
5	1e-5	0.7621 ± 0.1540	0.7235 ± 0.0333	0.7344 ± 0.0685
	1e-6	-	0.7387 ± 0.0238	-
	1e-4	0.7702 ± 0.1444	0.6620 ± 0.0974	0.7365 ± 0.0722
6	1e-5	0.7989 ± 0.1224	0.7157 ± 0.0557	0.6817 ± 0.1080
	1e-6	-	0.7440 ± 0.0257	-

Table 4.7: Mean AUROC scores and their standard deviations for neural network with TL from various levels and learning rates for each ontology on the truncated dataset.

Table 4.7 demonstrates results of neural network with transfer learning utilizing further dataset reduction by data truncation. For Molecular Function ontology, there is a positive correlation between higher learning rate and better performance on higher levels, while at level 4 and deeper, performance improves with slower learning rate. The standard deviations remain relatively high, indicating some variability. For the Biological Process ontology, lower learning rates yield consistently superior results across all levels. The performance on deeper levels (level 5 and beyond) is comparable to training without TL, and low standard deviations suggest stable performance. In contrast, for Cellular Component ontology, higher learning rates (1e-4) generally provide better results.

Level	LR	MF	BP	CC
2	1e-4	14 ± 20	5 ± 8	22 ± 57
	1e-5	14 ± 24	5 ± 7	14 ± 32
	1e-6	-	5 ± 9	-
3	1e-4	6 ± 7	13 ± 21	8 ± 12
	1e-5	6 ± 8	14 ± 29	11 ± 13
	1e-6	-	18 ± 40	-
4	1e-4	5 ± 16	8 ± 46	5 ± 6
	1e-5	3 ± 9	11 ± 66	7 ± 8
	1e-6	-	9 ± 47	-
5	1e-4	2 ± 3	3 ± 4	24 ± 55
	1e-5	3 ± 4	3 ± 3	18 ± 46
	1e-6	-	3 ± 2	-
6	1e-4	2 ± 1	5 ± 6	2
	1e-5	2 ± 1	9 ± 14	2
	1e-6	-	4 ± 3	-

Table 4.8: Mean training times in seconds and their standard deviations for neural network with TL from various levels and learning rates for each ontology on the truncated dataset.

The training times per term, as demonstrated in Table 4.8, are lowest in the case of a truncated dataset even on higher levels, as would be expected, given that the dataset decreases at a high rate from the very beginning.

CNN with Layer Freezing

Layer freezing is a method employed in transfer learning that aims to preserve knowledge from pretrained models by keeping specific layers static during subsequent training. This enables the model to keep previously learned features while simultaneously adapting the unfrozen layers to the new dataset. In this experiment, the embedding layer and the first convolutional layer were chosen to be frozen. Due to time constraints, the learning rate was set to 1e-4 for all experiments and no augmentation or truncation of the reduced datasets was applied.

Level	MF	BP	CC
2	0.6658 ± 0.1141	0.5763 ± 0.0815	0.7006 ± 0.0965
3	0.6872 ± 0.1190	0.5790 ± 0.0767	0.6661 ± 0.1099
4	0.7529 ± 0.1414	0.6182 ± 0.0713	0.6931 ± 0.0938
5	0.7381 ± 0.1694	0.6714 ± 0.0695	0.7228 ± 0.0667
6	0.7810 ± 0.1405	0.5771 ± 0.1037	0.7006 ± 0.0848

Table 4.9: Mean AUROC scores and their standard deviations for neural network with TL from various levels combined with layer freezing for each ontology on the original dataset.

Table 4.9 presents the results of training model with TL combined with frozen top layers. There are notable discrepancies in the performance across the ontologies. For Molecular Function and Cellular Component ontologies, the best performance is generally observed at the lower levels, however Biological Process ontology achieves the best results at mid-range levels. The results suggest that the employed layer freezing strategy proved to be ineffective. Potential improvements could include freezing different layers or applying the layer freezing later in the training process.

4.2.3 Optimal Strategy

The observed results suggest that the optimal approach would likely be a combination of using the CNN model without transfer learning on the heavily annotated high level terms and applying transfer learning from lower levels. While transfer learning does not usually result in improvement, and when it does, the gains are usually modest, the time saved by training on gradually decreasing datasets is substantial. This is particularly important considering the size of datasets the model is given.

Different ontologies require different approaches. For the Molecular Function ontology, transfer learning should be applied from level 4 and use lower learning rates (1e-5). Either data augmentation or truncation should be applied during the dataset reduction. A combination of these approaches could potentially lead to improvements, where on some levels augmentation and on other levels truncation would be used. For the Biological Process ontology, transfer learning should be initiated at level 5 or lower, using lower learning rates and utilizing the truncated datasets, as they yielded the best results (AUROC about 74% on low levels using learning rate 1e-6). In the case of Cellular Component ontology, transfer learning should also be employed at deeper levels. In contrast to MF and BP, the higher learning rate (1e-4) should be utilized as slowing learning rate led to performance degradation. The truncated dataset appears to be the most effective for CC, as it exhibits the lowest standard deviations.

4.3 Classifier Comparison

The proposed method - with and without transfer learning - was compared against established approaches such as BLAST-kNN, PRIORs and ESM-2, combined with logistic regression classifier. Performance metrics included F_1 scores, calculated at a threshold of 0.5, and AUROC scores. Given the inherent class inbalance in the datasets, AUROC is particularly informative as it assesses a model's discriminative ability across all thresholds, providing a more comprehensive evaluation than F_1 scores, which tend to be sensitive to class distribution and may not capture performance in imbalanced datasets.

The CNN-TL (optCNN-TL) results for each ontology were selected through the optimal strategy explained above: For MF, transfer learning is applied from level 4, using a truncated dataset and a learning rate of 1e-5. For BP, transfer learning starts at level 5, with a truncated dataset and a learning rate of 1e-6. For CC, transfer learning is applied from level 6, with a truncated dataset and a learning rate of 1e-4.

Classifier	MF		BP		CC	
	F_1	**AUROC**	F_1	**AUROC**	F_1	**AUROC**
PRIORS	0.02	0.5	0.0	0.5	0.02	0.5
BLAST-kNN	0.30	0.66	0.30	0.64	0.36	0.62
ESM2+LogReg	0.37	0.86	0.28	0.78	0.25	0.83
CNN-noTL	0.33	0.78	0.22	0.69	0.27	0.69
optCNN-TL	0.21	0.80	0.12	0.74	0.25	0.74

Table 4.10: Comparison of mean F_1 and AUROC scores for different classifiers.

As demonstrated in Table 4.10, PRIORS shows the lowest performance across all ontologies, with F1 scores close to 0 and AUROC at 0.5, indicating random predictions. This is expected given the simplistic nature of its annotation method. BLAST-kNN performs better than PRIORS, with F1 scores around 0.3 and AUROC values ranging from 0.62 to 0.66, indicating moderate performance. ESM2+LogReg achieves the highest AUROC for MF and CC (0.86 and 0.83 respectively), but its F1 scores remain relatively low, particularly for BP.

For the neural network classifiers, CNN-noTL shows better performance than PRIORS and BLAST-kNN, with F1 scores of 0.33 for MF and 0.27 for CC, and higher AUROC values (0.78 for MF and 0.69 for both BP and CC). However, the optimal CNN with transfer learning application (optCNN-TL) shows a decrease in performance, with the F1 score dropping to 0.21 for MF and 0.12 for BP, although its AUROC remains competitive, especially for MF (0.80). While the CNN models with and without transfer learning outperform traditional methods like PRIORS and BLAST-kNN in AUROC, they do not match the performance of ESM2.

Chapter 5

Conclusion

This thesis explored a range of methods for Automated Protein Function Prediction (AFP), bioinformatics task that aims to predict protein functions based on sequence data and annotations. Throughout the research, different AFP approaches were examined, from sequence similarity-based methods to more advanced techniques such as hierarchical classification and transfer learning.

A novel approach to AFP was proposed, combining Convolutional Neural Networks (CNNs) with transfer learning, specifically utilizing the hierarchical structure of Gene Ontology. The core idea was to reduce the dataset at each deeper level of the GO hierarchy by excluding proteins that were not annotated by the parent terms, thereby focusing the model on proteins more relevant to the current level. Transfer learning was applied by utilizing pretrained weights from higher-level terms on their children. Given the inherent imbalance of the dataset, particularly at deeper levels, data augmentation and truncation techniques were employed.

The results showed that while this approach to transfer learning overall did not lead to drastic improvements, it provided notable advantages when implemented under optimal conditions. The results indicated that the optimal transfer learning strategy varied depending on the ontology: for MF and BP, transfer learning from level 4 or 5 with a truncated dataset and a lower learning rate worked best, while for CC, transfer learning from level 5 or 6 with truncated datasets and a high learning rate yielded the most promising results. In comparison to baseline methods, such as PRIORS or BLAST-kNN, the CNN model demonstrated competitive performance, with a noticeable improvement in AUROC scores across all ontologies. While transfer learning sometimes led to marginal improvements in terms of AUROC scores, the greatest benefit was the reduction in training time, especially when using truncated datasets. This approach can be particularly beneficial when dealing with large datasets, where training without transfer learning would be time-consuming.

Moving forward, the method could be improved by utilizing embeddings from ESM2 and fine-tuning the model accordingly. Additionally, exploring different CNN architectures for the transfer learning process, experimenting with different learning rates or freezing different layers during training could lead to more effective use of transfer learning and potentially improve the overall prediction accuracy in automated protein function prediction.

Bibliography

[1] QuickGO Ancestor Chart for GO:0000166. `https://www.ebi.ac.uk/QuickGO/term/GO:0002516`. Accessed: 17 January 2024.

[2] JGraph. diagrams.net, draw.io, 10 2021.

[3] Uniprot: the universal protein knowledgebase in 2023. *Nucleic Acids Research*, 51(D1):D523–D531, 2023.

[4] Stavros Makrodimitris, Roeland CHJ Van Ham, and Marcel JT Reinders. Automatic gene function prediction in the 2020's. *Genes*, 11(11):1264, 2020.

[5] David W Mount. Using the basic local alignment search tool (blast). *Cold Spring Harbor Protocols*, 2007(7):pdb–top17, 2007.

[6] Giorgio Valentini. Hierarchical ensemble methods for protein function prediction. *International Scholarly Research Notices*, 2014, 2014.

[7] Mohammad Nauman, Hafeez Ur Rehman, Gianfranco Politano, and Alfredo Benso. Beyond homology transfer: Deep learning for automated annotation of proteins. *Journal of Grid Computing*, 17:225–237, 2019.

[8] Gabriela A Merino, Rabie Saidi, Diego H Milone, Georgina Stegmayer, and Maria J Martin. Hierarchical deep learning for predicting go annotations by integrating protein knowledge. *Bioinformatics*, 38(19):4488–4496, 2022.

[9] Weiqi Xia, Lingyan Zheng, Jiebin Fang, Fengcheng Li, Ying Zhou, Zhenyu Zeng, Bing Zhang, Zhaorong Li, Honglin Li, and Feng Zhu. Pfmuldl: a novel strategy enabling multi-class and multi-label protein function annotation by integrating diverse deep learning methods. *Computers in Biology and Medicine*, 145:105465, 2022.

[10] Suzi A Aleksander, James Balhoff, Seth Carbon, J Michael Cherry, Harold J Drabkin, Dustin Ebert, Marc Feuermann, Pascale Gaudet, Nomi L Harris, et al. The gene ontology knowledgebase in 2023. *Genetics*, 224(1):iyad031, 2023.

[11] Function Special Interest Group the cafa challenge. `https://biofunctionprediction.org/cafa`. Accessed: 17 January 2024.

[12] Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, 2000.

[13] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[14] Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J Inman. 1d convolutional neural networks and applications: A survey. *Mechanical systems and signal processing*, 151:107398, 2021.

[15] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III 27*, pages 270–279. Springer, 2018.

[16] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[17] R Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. *Morgan Kaufman Publishing*, 1995.

[18] Jason Wei and Kai Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*, 2019.

[19] Steven Henikoff and Jorja G Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, 1992.

[20] Sergey A Shiryev, Jason S Papadopoulos, Alejandro A Schäffer, and Richa Agarwala. Improved blast searches using longer words for protein seeding. *Bioinformatics*, 23(21):2949–2951, 2007.

[21] Hetal Bhavsar and Amit Ganatra. A comparative study of training algorithms for supervised machine learning. *International Journal of Soft Computing and Engineering (IJSCE)*, 2(4):2231–2307, 2012.

[22] Sayali D Jadhav and HP Channe. Comparative study of k-nn, naive bayes and decision tree classification techniques. *International Journal of Science and Research (IJSR)*, 5(1):1842–1845, 2016.

[23] Ronghui You, Shuwei Yao, Hiroshi Mamitsuka, and Shanfeng Zhu. Deepgraphgo: graph neural network for large-scale, multispecies protein function prediction. *Bioinformatics*, 37(Supplement_1):i262–i271, 2021.

[24] Xiangyun Wang, Diane Schroeder, Drena Dobbs, and Vasant Honavar. Automated data-driven discovery of motif-based protein function classifiers. *Information Sciences*, 155(1-2):1–18, 2003.

[25] Asa Ben-Hur and Douglas Brutlag. Sequence motifs: highly predictive features of protein function. In *Feature Extraction: Foundations and Applications*, pages 625–645. Springer, 2006.

[26] Giorgio Valentini. True path rule hierarchical ensembles for genome-wide gene function prediction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(3):832–847, 2010.

[27] Amelia Villegas-Morcillo, Stavros Makrodimitris, Roeland CHJ van Ham, Angel M Gomez, Victoria Sanchez, and Marcel JT Reinders. Unsupervised protein embeddings outperform hand-crafted sequence and structure features at predicting molecular function. *Bioinformatics*, 37(2):162–170, 2021.

[28] Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C Lawrence Zitnick, Jerry Ma, et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15):e2016239118, 2021.

[29] Arthur B Kahn. Topological sorting of large networks. *Communications of the ACM*, 5(11):558–562, 1962.

[30] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

[31] Shekoufeh Gorgi Zadeh and Matthias Schmid. Bias in cross-entropy-based training of deep survival networks. *IEEE transactions on pattern analysis and machine intelligence*, 43(9):3126–3137, 2020.

[32] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[33] Vladimir Gligorijević, P Douglas Renfrew, Tomasz Kosciolek, Julia Koehler Leman, Daniel Berenberg, Tommi Vatanen, Chris Chandler, Bryn C Taylor, Ian M Fisk, Hera Vlamakis, et al. Structure-based protein function prediction using graph convolutional networks. *Nature communications*, 12(1):3168, 2021.

[34] Tobias Hamp, Rebecca Kassner, Stefan Seemayer, Esmeralda Vicedo, Christian Schaefer, Dominik Achten, Florian Auer, Ariane Boehm, Tatjana Braun, Maximilian Hecht, et al. Homology-based inference sets the bar high for protein function prediction. In *BMC bioinformatics*, volume 14, pages 1–10. Springer, 2013.

[35] Gene ontology resource. `https://geneontology.org/docs/download-ontology`. Accessed: 30 November 2023.

[36] Jin Huang and Charles X Ling. Using auc and accuracy in evaluating learning algorithms. *IEEE Transactions on knowledge and Data Engineering*, 17(3):299–310, 2005.