Biological Data Project on Protein Function Prediction

Ferlin Valeria
valeria.ferlin@studenti.unipd.it

Piazza Mattia
mattia.piazza@studenti.unipd.it

Tiso Elia
elia.tiso@studenti.unipd.it

Contents

1 Introduction

The objective of this project is to accurately predict protein function in terms of its Gene Ontology (GO) sub-ontology. Specifically, we aim to classify proteins into the three primary GO sub-ontologies:

- *Molecular Function (MF)* describing specific biochemical activities of proteins.
- *Cellular Component (CC)* identifying the subcellular localization of proteins.
- *Biological Process (BP)* capturing broader biological roles that proteins play.

The dataset used for this task consists of approximately 124,000 proteins, where GO terms appear at least 50 times to ensure a more balanced distribution. Our training data comprises multiple files providing diverse information about the proteins:

- `train_set.tsv` – contains GO annotations and their corresponding sub-ontologies.
- `train_ids.txt` – lists all protein IDs.
- `train.fasta` – provides the amino acid sequences of the proteins.
- `train_embeddings.h5` – includes ProtT5 embeddings representing each protein.
- `train_protein2ipr.dat` – contains InterPro domain annotations.
- `go-basic.obo` – defines the GO ontology structure and relationships.

Additionally, the test set consists of approximately 1,000 proteins and includes analogous files: `test_ids.txt`, `test_embeddings.h5`, `test.fasta`, and `test_protein2ipr.dat`. Furthermore, the file `blast_test_results.tsv` provides BLAST alignment results of the test proteins against those in the training set, though its utility for our predictive task remains limited.

2 Analysis

We started our analysis by consolidating all files related to the training set into a dataframe with columns ('ID', 'embeddings', 'sequence', 'ipr', 'domain', 'familyID', 'start', 'end'). This allowed us to inspect and manipulate the available data efficiently.

Upon inspecting the dataset, we noticed that a significant portion of the available information was not useful for our predictive task. Consequently, we focused exclusively on the *protein embeddings* extracted from the provided `train_embeddings.h5` file. These embeddings, derived from the ProtT5 model, represent protein sequences in a high-dimensional vector space of size **1024**.

Since dealing with such high-dimensional data can be computationally expensive, we performed Principal Component Analysis (PCA) to assess the explained variance (Figure 1). We found that approximately **300** principal components were sufficient to capture 90% of the variance.

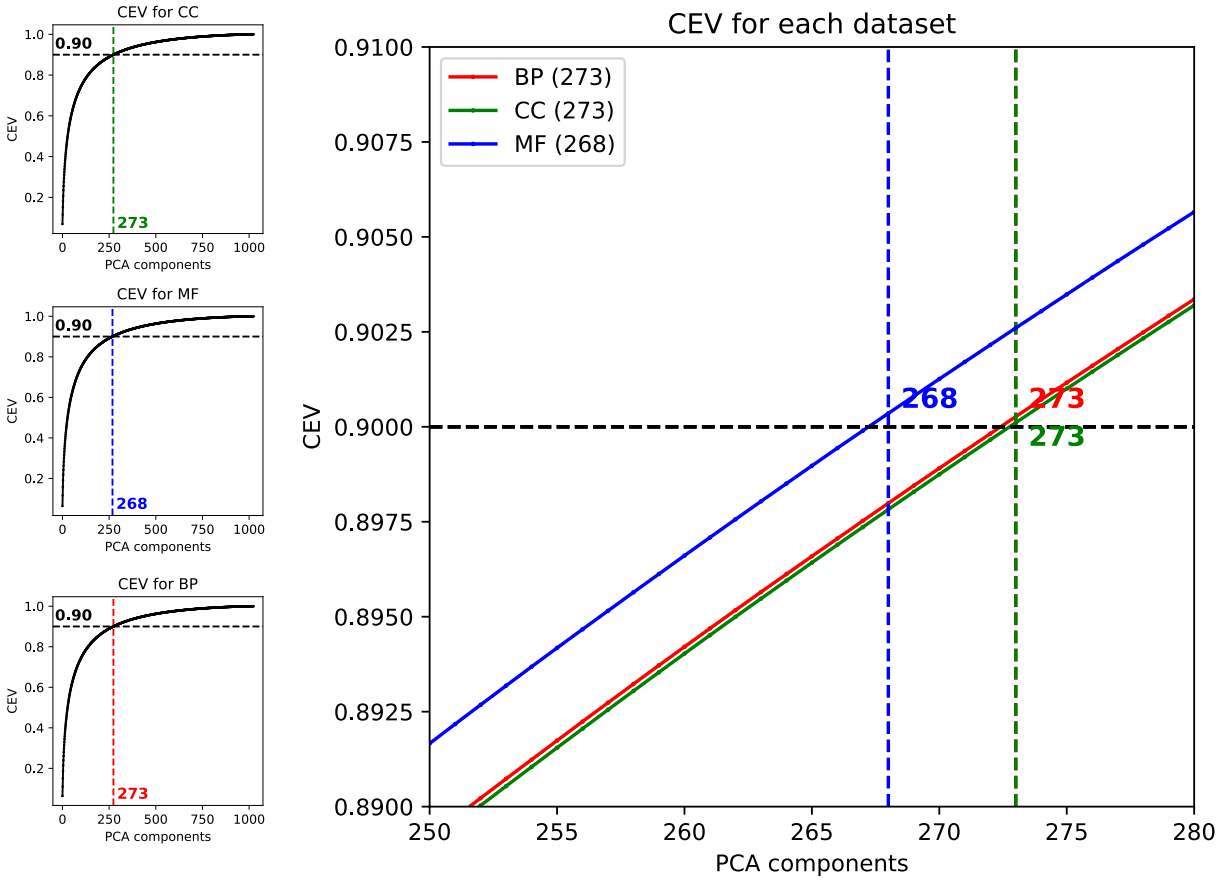


Figure 1 : *PCA component analysis onto each of the sub-dataframes by GO. We highlight the number of components needed to reach the target variance.*

However, since our selected models were capable of handling the full embedding space without performance degradation, we opted to retain all 1024 features.

3 Models

To predict protein function, we developed models for each of the three GO function categories. Since these sub-ontologies are independent, we trained separate models for each category. To maximize predictive accuracy, we explored various architectures and methodologies, selecting the most effective approach for each case. Our evaluation relied on *weighted Precision-Recall (PR) Curves* and *weighted F1-score (F1)*, ensuring that our models consistently outperformed the CAFA baselines.

Taking into consideration the unbalanced multi-label nature of our problem, we focused on optimising the weighted F1 (1), defined as

$$F1_{\text{weighted}} = \sum_{i=1}^{L} w_i \cdot F1_i, \quad \text{where} \quad w_i = \frac{N_i}{\sum_{j=1}^{L} N_j}, \tag{1}$$

where:

- L is the total number of labels.
- $F1_i$ is the F1 (2) for the i-th label.
- N_i is the number of true instances for the i-th label.
- w_i is the weight for each label, computed as the proportion of true instances.

We recollect that the F1 for each label is defined as

$$F1_i = \frac{2 \cdot P_i \cdot R_i}{P_i + R_i} \quad \text{where} \quad P_i = \frac{TP_i}{TP_i + FP_i}, \quad R_i = \frac{TP_i}{TP_i + FN_i} \tag{2}$$

where:

- P_i is the Precision.
- R_i is the Recall.
- TP_i is the number of true positives for label i.
- FP_i is the number of false positives for label i.
- FN_i is the number of false negatives for label i.

Although we initially explored models such as CatBoost, SGDClassifier, LightGBM, and Random Forests, we quickly shifted to a Neural Network approach. Given the vast number of high-dimensional training samples - and especially the many classes in our multilabel classification problem - the versatility of NNs, combined with GPU acceleration, proved far more effective.

3.1 Cellular Component

In this section, we compare the parameters and results of the five neural network models (NN_1, NN_2, NN_3 and NN_4) used in our analysis for the CC dataset.

Model	*Hidden Layer Sizes*	*Dropout*	*Learning Rate*	*Batch Size*	*Epochs*
NN_1	[4096, 4096]	[0.5, 0.4]	0.002	512	75
NN_2	[4096, 2048]	[0.1, 0.1]	0.001	1024	75
NN_3	[2048, 1024]	[0.2, 0.2]	0.001	2048	75
NN_4	[4096, 2048, 1024]	[0.2, 0.2, 0.2]	0.005	512	75

Table 1 : *Model parameters. Dropout rates indicate the probability of deactivating a hidden unit during training.*

Metric	NN_1	NN_2	NN_3	NN_4
Macro P	0.41	0.54	0.46	0.53
Macro R	0.44	0.30	0.17	0.27
Macro F1	0.42	0.36	0.22	0.33
Weighted P	0.71	0.75	0.74	0.74
Weighted R	0.71	0.63	0.62	0.65
Weighted F1	0.71	0.67	0.65	0.68
Samples P	0.75	0.81	0.82	0.80
Samples R	0.77	0.70	0.69	0.72
Samples F1	0.72	0.71	0.71	0.72

Table 2 : *Model results with best and worst values highlighted.*

Some observations on Tables 1 and 2:

- **NN**$_1$: While the Precision was the worst, overall we can consider it the best one.
- **NN**$_2$: Improved Precision but lower Recall compared to NN$_1$.
- **NN**$_3$: Reduced hidden layer sizes and increased batch size reduced macro F1.
- **NN**$_4$: Added an additional hidden layer and maintained dropout rates.

Overall, NN$_1$ (hereafter NN$_{\text{CC}}$) provided the best balance between Precision, Recall, and F1.

3.2 Molecular Function

In this section, we present a comparison of the parameters and results of the five neural network models (NN$_1$, NN$_2$, NN$_3$ and NN$_4$) utilized in our analysis of the MF dataset.

Model	*Hidden Layer Sizes*	*Dropout*	*Learning Rate*	*Batch Size*	*Epochs*
NN$_1$	[4096, 2048]	[0.2, 0.2]	0.001	512	50
NN$_2$	[4096, 4096]	[0.4, 0.3]	0.001	512	100
NN$_3$	[4096, 4096]	[0.4, 0.3]	0.001	1024	150
NN$_4$	[8192, 4096, 2048]	[0.4, 0.3, 0.3]	0.001	512	100

Table 3 : *Model parameters.*

Metric	*NN$_1$*	*NN$_2$*	*NN$_3$*	*NN$_4$*
Macro P	0.69	0.62	0.68	0.65
Macro R	0.46	0.57	0.47	0.54
Macro F1	0.53	0.58	0.53	0.57
Weighted P	0.77	0.74	0.77	0.74
Weighted R	0.65	0.70	0.66	0.69
Weighted F1	0.68	0.71	0.69	0.71
Samples P	0.83	0.79	0.83	0.80
Samples R	0.70	0.75	0.71	0.75
Samples F1	0.72	0.73	0.73	0.74

Table 4 : *Model results with best and worst values highlighted.*

Some observations on Tables 3 and 4:

- **NN**$_1$: Balanced performance with moderate Precision, Recall, and F1.
- **NN**$_2$: Best Recall and F1 but lowest - however still good - Precision.
- **NN**$_3$: Best Precision, however average results otherwise.
- **NN**$_4$: Best sample metrics.

Overall, NN$_2$ (hereafter NN$_{\text{MF}}$) provided the best balance between Precision, Recall, and F1.

3.3 Biological Process

In this section, we present a comparison of the parameters and results of the five neural network models (NN$_1$, NN$_2$, NN$_3$, NN$_4$, and NN$_5$) utilized in our analysis of the BP dataset.

Model	*Hidden Layer Sizes*	*Dropout*	*Learning Rate*	*Batch Size*	*Epochs*
NN_1	[8192, 4096]	[0.5, 0.4]	0.001	512	120
NN_2	[2048, 1024]	[0.2, 0.2]	0.003	1024	75
NN_3	[4096, 2048]	[0.4, 0.4]	0.005	1024	100
NN_4	[4096, 2048, 2048]	[0.2, 0.2, 0.2]	0.003	1024	75
NN_5	[2048, 2048, 2048]	[0.2, 0.2, 0.2]	0.003	1024	75

Table 5 : *Model parameters.*

Metric	NN_1	NN_2	NN_3	NN_4	NN_5
Macro P	0.43	0.52	0.58	0.56	0.55
Macro R	0.34	0.22	0.22	0.22	0.21
Macro F1	0.37	0.29	0.29	0.29	0.28
Weighted P	0.54	0.58	0.62	0.61	0.60
Weighted R	0.47	0.39	0.39	0.37	0.38
Weighted F1	0.50	0.44	0.44	0.44	0.44
Samples P	0.59	0.62	0.65	0.66	0.64
Samples R	0.53	0.45	0.46	0.44	0.44
Samples F1	0.50	0.46	0.46	0.46	0.46

Table 6 : *Model results with best and worst values highlighted.*

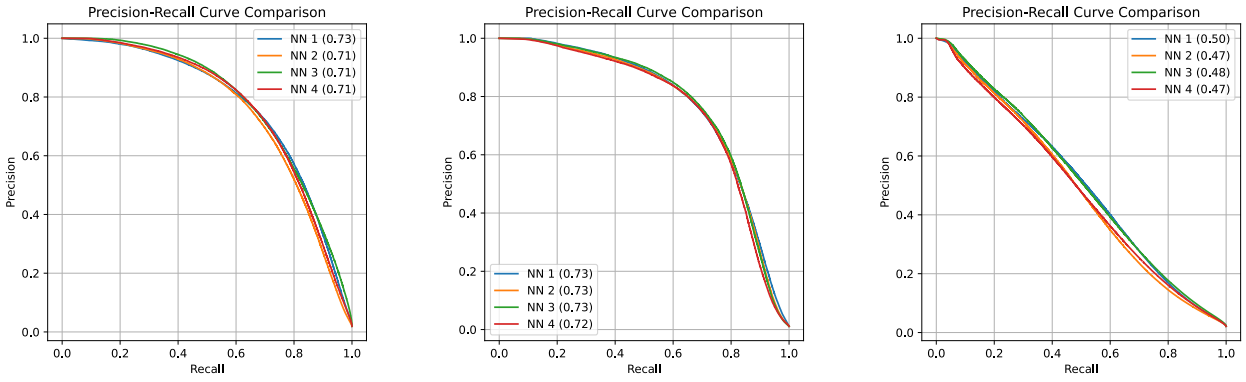
Some observations on Tables 5 and 6:

- **NN_1**: Best macro and samples Recall, indicating strong ability to capture positive instances.
- **NN_2**: Lower Recall metrics but competitive Precision, showing a more conservative prediction style.
- **NN_3**: Highest weighted Precision but lowest macro Recall.
- **NN_4**: Balanced performance across all metrics without extreme highs or lows.
- **NN_5**: Consistently lower performance across several metrics, with no clear area of strength.

Overall, NN_1 demonstrated the best recall performance, while NN_3 had the highest weighted precision. Depending on the use case, NN_1 could be preferable for recall-focused tasks, whereas NN_3 excels when precision is key. However, NN_1 (hereafter NN_{BP}) provided the best results for our target metric.

4 Results

Considering the analysis and tables in 3 - Models and the PR Curves (Figure 2) we can now summarize our results and identify the best models.



(a) *PR Curve for CC dataset.* **(b)** *PR Curve for MF dataset.* **(c)** *PR Curve for BP dataset.*

Figure 2 : *Comparison of Precision-Recall Curves for the models.*

4.1 Best models

Overall, the best models identified for each problem are the following. Their architecture is shown on page 9 (Figures 5a to 5c respectively).

	Type	*Precision*	*Recall*	*F1-Score*
NN_{CC}	Weighted	0.71	0.71	0.71
	Macro	0.41	0.44	0.42
NN_{MF}	Weighted	0.74	0.70	0.71
	Macro	0.62	0.57	0.58
NN_{BP}	Weighted	0.54	0.47	0.50
	Macro	0.43	0.34	0.37

Table 7 : *Best models results.*

Model	*Hidden Layer Sizes*	*Dropout*	*Learning Rate*	*Batch Size*	*Epochs*
NN_{CC}	[4096, 4096]	[0.5, 0.4]	0.002	512	75
NN_{MF}	[4096, 4096]	[0.4, 0.3]	0.001	512	100
NN_{BP}	[8192, 4096]	[0.5, 0.4]	0.001	512	120

Table 8 : *Best models parameters.*

4.2 Comparison with IPR and Naïve

Let us compare the results of the best-performing NNs with those of the IPR and Naïve models. The comparison focuses on the weighted F1, which are the target performance metric, as well as PR Curves.

The results for the IPR and Naïve models are summarized in Figure 3.

Model	*P*	*R*	*F1*	*F1 Increment*
Naïve CC	1.00	0.67	0.60	0.11
IPR CC	0.46	0.92	0.26	0.45
NN_{CC}	0.71	0.71	0.71	-
Naïve MF	1.00	0.69	0.45	0.26
IPR MF	0.91	0.77	0.62	0.09
NN_{MF}	0.74	0.70	0.71	-
Naïve BP	1.00	0.32	0.34	0.16
IPR BP	0.71	0.78	0.23	0.27
NN_{BP}	0.54	0.47	0.50	-

Figure 3 : *Comparison of model results for P, R, F1, and F1 increment across Naïve, IPR, and NN models (weighted).*

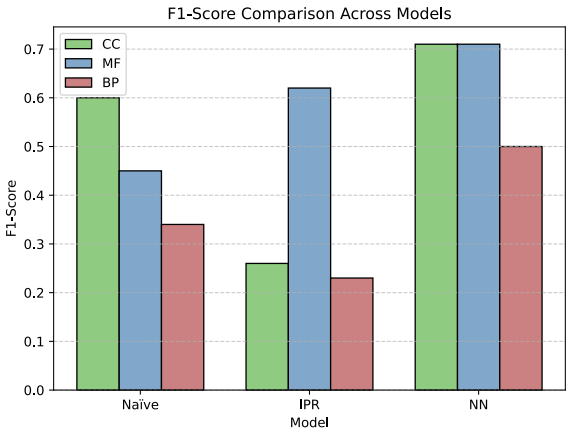


Figure 4 : *Comparison of F1 across models.*

From the results in Figure 3, we observe the following key points:

- **IPR:** High Recall in CC (0.92) and MF, but low F1 due to lower Precision, particularly in BP (F1: 0.23) and CC (F1: 0.26).
- **Naïve:** Perfect Precision (1.00) across all datasets, but low Recall, especially for BP (0.32), resulting in poor F1 (e.g., 0.34 for BP).
- **NNs:** NN_{MF} has the best performance (F1: 0.71), balancing Precision and Recall, while NN_{CC} and NN_{BP} show more inconsistency, especially in the macro metrics.

The key takeaway is that NNs, especially NN_{MF}, offer a better balance between Precision and Recall compared to both the IPR and Naïve models. This is particularly evident in the weighted F1 (Figure 4) and PR Curves

(Figure 2), where NN_{MF} demonstrates consistent performance across all datasets. NN_{CC} and NN_{BP}, while not outperforming NN_{MF}, also show competitive results with better F1 than the IPR and Naïve models.

Overall, the comparison highlights the strength of the neural network models, with NN_{MF} standing out as the best-performing model for this task.

IPR		
O43747	GO:0030117	1.0
O43747	GO:0006886	1.0
O43747	GO:0030121	1.0
O43747	GO:0005794	1.0
etc.		

Naïve		
O43747	GO:0003674	1.0
O43747	GO:0005488	0.66
O43747	GO:0005515	0.507
O43747	GO:0003824	0.447
etc.		

NN		
A0A0B4JCV4	GO:0110165	1.000
A0A0B4JCV4	GO:0005575	1.000
A0A0B4JCV4	GO:0008150	1.000
A0A0B4JCV4	GO:0003674	1.000
etc.		

Table 9 : *Comparison of the first outputs.*

Let us consider as an example the very first ID: O43747. As we can see in Table 10, it is clear that our model, while not being able to fully capture the 82 different labels annotated[1], outperforms the two baselines.

Model	*TP*	*FP*	*FN*	*Total*
Ipr	5	0	36	5
Naïve	7	195	34	202
NN (0.05, 50)	19	94	22	113
NN (0.05, 30)	14	59	27	73
NN (0.25, 30)	12	47	29	59
NN (0.50, 30)	6	18	35	24

Table 10 : *Performance metrics for ID: O43747. For the NN model, we set different values* (a, b) *such that* a *is the cutoff for the probability of a predicted class to be considered and* b *is the maximum number of classes to be predicted per sub-ontology.*

These results show that our model significantly improves upon both baselines. Compared to the Naïve approach, it drastically reduces False Positives while maintaining a much higher number of True Positives. Similarly, while the IPR method yields no False Positives, it struggles to identify relevant annotations, leading to a very low Recall. In contrast, our model achieves a better balance between Precision and Recall, successfully capturing a larger portion of the annotated labels while keeping False Positives under control. This highlights its effectiveness in refining annotation predictions.

5 Conclusion

In this project, we tackled the challenge of protein function prediction using machine learning models, particularly focusing on Neural Networks (NNs) trained on ProtT5 embeddings. Our study aimed to classify proteins within the three major Gene Ontology (GO) sub-ontologies: Molecular Function (MF), Cellular Component (CC), and Biological Process (BP).

Our initial exploration of various machine learning models, including CatBoost, LightGBM, and Random Forests, revealed their limitations in handling the high-dimensional, multi-label nature of the problem. This led us to shift towards Neural Networks, leveraging their ability to process large-scale embeddings and model complex relationships.

The results showed that NNs outperformed traditional methods, particularly in the MF and CC sub-ontologies, where they provided a good balance between Precision and Recall. However, BP classification remained the most challenging, likely due to the broader and more hierarchical nature of biological processes. While our models outperformed both the IPR and Naïve baselines, especially in terms of F1, the overall predictive performance still leaves room for improvement.

Key observations from our findings:

- NN_{MF} (Molecular Function) emerged as the best-performing model, achieving the highest F1.
- NN_{CC} (Cellular Component) provided competitive results, with a solid balance of Precision and Recall.

[1] https://www.ebi.ac.uk/QuickGO/annotations?geneProductId=O43747

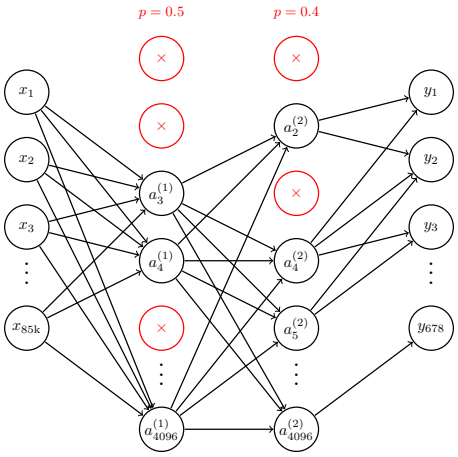
- NN_{BP} (Biological Process) struggled the most, reinforcing the need for context-aware and structured prediction approaches.

Future improvements could involve:

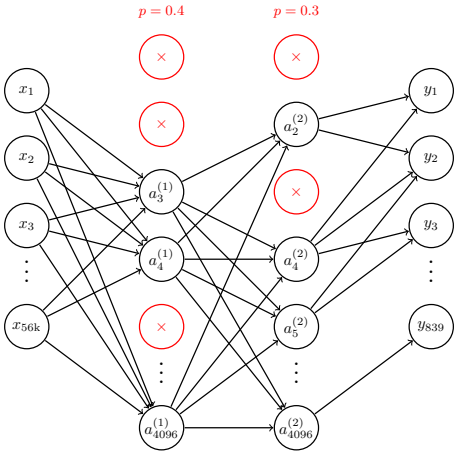
- Graph-based models to better leverage the hierarchical relationships within the GO ontology.
- Transformer-based architectures (e.g., ProtBERT) for richer sequence-based feature extraction.
- Ensemble learning to combine the strengths of multiple classifiers.
- Attention-based architectures to focus on relevant features within protein sequences and domains.

In conclusion, while our neural network models show promise, protein function prediction remains a complex challenge. Advancing the field will require more sophisticated architectures and integration of additional biological knowledge beyond just sequence embeddings.

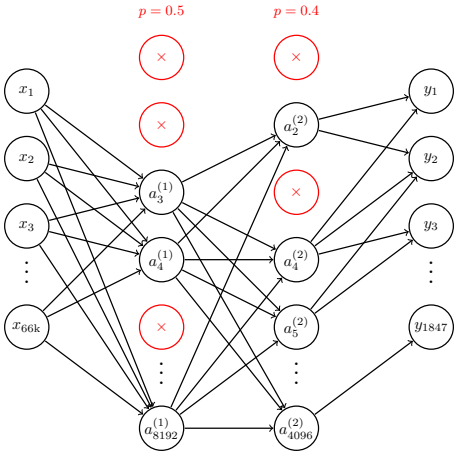
A Additional figures



(a) *NN_{CC} architecture.*



(b) *NN_{MF} architecture.*



(c) *NN_{BP} architecture.*

Figure 5 : *Best models architecture.*