CAP 6545: Bioinformatics, 2023, pp. 1–4

doi:
Final Project Paper

FINAL PROJECT PAPER

Tackling the tRNA Thermometer Problem

Andres Graterol[1,*]

[1]CECS, University of Central Florida, 4000 Central Florida Blvd, 32816, Florida, United States
[*]graterol.andres0@knights.ucf.edu

Abstract

In this paper, we make an attempt at recreating and improving upon the work done by Cimen et al in the paper titled 'Building a tRNA thermometer to estimate microbial adaptation to temperature'. In the process of replicating the paper, I managed to develop helper scripts to reach comparable results. Additionally, work was done in the hopes of improving the results reported by the authors of the original paper, such as hyperparameter fine-tuning, and the use of transformer models. The scripts used to gather the data for this paper can be found here: `https://github.com/graterollin/bioinformatics-trna-thermometer`

Key words: OGT, tRNA Thermometer, CNN, Regression, Transformers

Introduction

In this paper, we are following the steps described in the work titled 'Building a tRNA thermometer to estimate microbial adaptation to temperature' (1) to try to achieve comparable results. After analyzing these results, we then propose two methods for improving the performance of the two models that make up the backbone of the original paper.

Background

The work done by Cimen et al. revolves mainly around constructing models that will achieve tasks related to organisms' optimal growth temperature (OGT). This work is deemed important, as creating a 'tRNA Thermometer', as the original authors coined it, could save future researchers a lot of valuable time in the lab as knowing the OGT of a species could skip the entire culturing process. To achieve this research goal, two models were produced, a temperature classifier, and an OGT predictor. Both models were trained and tested against both randomly split data and phylogenetic distance split data.

Data Collection and Preprocessing

A list of species labeled with their OGT was obtained from the work done by Sauer et al. (2). To get the data needed for the tRNA and rRNA sequences, such as the sequence and position for each genome, the authors of the original paper used the softwares tRNAscan-SE and barrnap. A single genome was then selected for each species, and the data was appended to the CSV file that contained all of the species and their OGT.

Input Data Formatting

The temperature classifier takes as input a CSV file that contains groupings of species, their OGTs, tRNAids, and Anticodons. As stated previously, most of this data was obtained by the use of tRNAscan-SE. The OGT predictor on the other hand, also takes a CSV file as input, but this one includes a genome assembly, species name, OGT, the OGT source, the tRNAid, the length of the sequence, the amino acid designator, the anticodon, anticodon position, Cove score, sequence, the structure, the 16S rRNA and the 23S rRNA sequence. Most of this data was obtained through the use of the barrnap software.

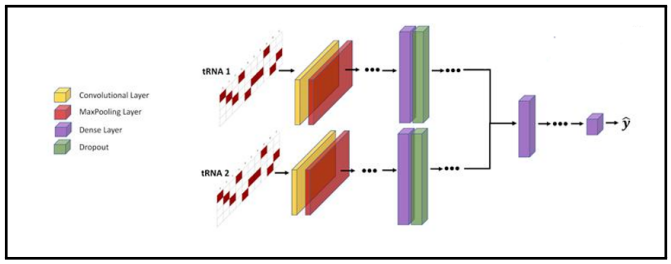

Fig. 1. Architecture of the CNN (1)

Temperature Classifier

The temperature classifier's objective was to take data from pairs of species and then output a prediction label that would classify whether species 1 or species 2 had the higher OGT. This was achieved through the use of a classification Convolutional Neural Network (CNN) whose structure can be seen in *figure 1*. Specifically, the authors used a two-branch version of a CNN, where each branch handles the tRNA for each species being compared. In detail, the Convolutional and MaxPooling layer(s) make up the first level of the classifier. The output from these layers are then fed into the hidden layer, consisting of Dense and Dropout layers. Here, the Dropout layer serves to

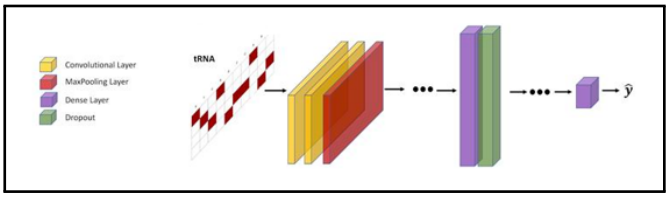

Fig. 2. Architecture of the Regression Model (1)

prevent overfitting After each branch goes through these levels, the output from each is then combined together in final Dense layer(s) and an output layer then makes the final prediction.

Species OGT Prediction

The goal of this logistic regression model is to predict the OGT for a given species. In contrast to the temperature classifier, this model only considers one species at a time. This logistic regression model also follows a CNN structure, and a visualization of its architecture can be seen in *figure 2*. In more detail, two Convolutional layers are followed by a MaxPool layer, which then feeds into the next level, consisting of a Dense and Dropout layer. The final layer then outputs the model's prediction for the species' OGT.

Implementation

Code Fixes and Additional Scripts

The authors of the original paper were generous enough to provide a link to a repository where they kept most of the input data as well as the scripts needed to replicate the output of the models they listed in the paper. However, I ran into some issues while attempting to replicate their experiment, so in this section I will list off some of the crucial changes that needed to be made in order to get comparable results as well as briefly mention some helper scripts I developed to aid the the process of obtaining said results.

The output from the temperature classifier only included a CSV file that followed the form:

```
Species1, OGT1, Species2, OGT2, Prediction
```

However, the paper had included results that had split up classification accuracy into bins depending the temperature range between the species being compared.

In order to see these results, I developed a script named *Classification_ResultAnalysis.py*. This script read through the output CSV file, and outputted the classification accuracy for species that fit into these temperature difference bins:

- Between 0 and 5 degrees Celsius
- Between 5 and 10 degrees Celsius
- Between 10 and 20 degrees Celsius
- Between 20 and 30 degrees Celsius
- 30 and above degrees Celsius

The most code changes came from attempting to run the scripts that had the data split according to phylogenetic distance. As a prerequisite for running these model scripts, a phylogenetic distance matrix is needed to be fed in as input. The authors failed to provide this file, so I had to go about replicating the file on my own. Here are the steps that I took in order to achieve this goal:

1. Created a script, *GetSpeciesList.py* that collects a list of all unique archaea and bacteria species put into a one-column text file.
2. Inputted text file into NCBI taxonomy to get taxonomy information related to each species, most importantly, a taxonomy ID (taxid).
3. Created a script, *BuildTaxidFile.py* that would leave only the taxid and species name as columns and output that as a new file.
4. Used the following line in a Linux environment to quickly drop the species name and leave only a column of taxids:

   ```
   awk {print \$1} <taxid\_name\_file> > <new\_taxid\_file>
   ```

5. Navigated to the NCBI common tree page and inserted the list of taxids. Expanded all nodes in the tree and downloaded the expanded tree in phylip format.
6. Deleted any newline characters and replaced all spaces with underscored to get to Newick format using the following line in a Linux environment:

   ```
   sed -i -e -E 's/\n//g;s/ /_/g
   ```

7. Ran the R script, *6_createDistMxFromTree.R* which uses the *ape* package to convert the phylip file into a final distance matrix in a text file format.

Lastly, some errors in the code needed to be fixed. More specifically, the phylogenetic split scripts were not grabbing in the data from the distance matrix correctly, and both types of scripts had early stopping functionality on their models, which caused the scripts on my end to terminate early every time.

Implementation Results Comparison and Analysis

Random Split Data Results

Temperature Difference	Archaea		Bacteria		Archaea + Bacteria	
	My Results	Paper Results	My Results	Paper Results	My Results	Paper Results
>0° C	0.522	0.814	0.600	0.756	0.529	0.728
>5° C	0.629	0.889	0.768	0.824	0.642	0.788
>10° C	0.726	0.93	0.746	0.830	0.701	0.853
>20° C	0.888	0.902	0.780	0.882	0.843	0.914
>30° C	0.991	0.954	0.889	0.921	0.969	0.950

Fig. 3. Classification Random Split Comparison

Figure 3 and *figure 4* show the results obtained from using the randomly split dataset for the classification and regression models, respectively. As can be seen in *figure 3*, the results I obtained gradually reach those achieved by the paper as the temperature difference between the two species grows. It is interesting to note that for the group for the species with higher than 30 degrees of difference, my model even obtained higher accuracies results for the Archaea and Archaea & Bacteria runs. To this point, I believe that larger datasets were used by the original authors. Through my research, I believe that the original bacteria dataset, for example, used to have 1

million entries, while the one that they provided only had 100 thousand. This might be due to the file size limits imposed by bitbucket, the repository that the authors chose to use. This discrepancy could be an explanation as to why our accuracies vary. Through my analysis, most samples in my dataset where species that had temperature differences larger than 30 degrees.

Source	Archaea		Bacteria		Archaea + Bacteria	
	RMSE	r^2	RMSE	r^2	RMSE	r^2
My Results	7.77	0.887	6.87	0.810	7.64	0.863
Paper Results	8.06	0.862	6.76	0.818	7.31	0.875

Fig. 4. Regression Random Split Comparison

Regarding the results from the regression model shown in *figure 4*, RMSE can be seen as the average difference between values predicted by a model and the true values. Meanwhile, r^2 is known as the coefficient of determination and it is a measure of variance. For logistic regression models, the lower the RMSE and the higher the r^2, the better fit the model is. The results I obtained here closer to the performance depicted by the paper, and from the results, my archaea run even proved to be a better fit.

Phylogenetic Split Data Results

Figures 5 and *6* show the performance of the two models that ran based off the phylogenetic distance matrix between species. For this data split, both models see a decline in performance. Looking more closely at *figure 5*, my results do not vary from

Temperature Difference	Archaea		Bacteria		Archaea + Bacteria	
	My Results	Paper Results	My Results	Paper Results	My Results	Paper Results
>0° C	0.532	0.684	0.497	0.538	0.506	0.564
>5° C	0.518	0.749	0.514	0.557	0.519	0.585
>10° C	0.573	0.807	0.544	0.612	0.512	0.605
>20° C	0.720	0.902	0.637	0.660	0.557	0.622
>30° C	0.903	0.954	0.642	0.694	0.625	0.636

Fig. 5. Classification Phylogenetic Split Comparison

the paper as much when compared to the random split results. However, for these runs, my results did not out-perform any of the paper's.

Source	Archaea		Bacteria		Archaea + Bacteria	
	RMSE	r^2	RMSE	r^2	RMSE	r^2
My Results	39.36	-1.834	11.46	0.503	13.67	0.585
Paper Results	11.06	0.772	12.67	0.370	13.23	0.590

Fig. 6. Regression Phylogenetic Split Comparison

Another interesting observation to note is the regression output for archaea seems to be indicative of the model being an extremely poor fit for the data.

Improvement Results Comparison and Analysis

In this section I will go into detail describing the steps I took to try to improve on the results I obtained in order to get closer to the results achieved by the original authors.

Hyperparameter Fine-tuning

Hyperparameters describe the parameters that are used to tweak the behavior of the layers that make a Neural Network.

```
'hidden_layer1': [4, 8, 16, 32, 64, 128],
'drop_out1':  [0, 0.05, 0.1, 0.2, 0.3],
'filter_size1': [4, 8, 16, 32, 64, 128],
'kernel_size1': [1, 3, 5, 8, 16, 32],
'pool_size1': [3, 5, 8, 16, 32],
'strides1':  [1, 3, 5, 8],
'hidden_layer2':[4, 8, 16, 32, 64, 128],
'drop_out2': [0, 0.05, 0.1, 0.2, 0.3],
'filter_size2':  [4, 8, 16, 32, 64, 128],
'kernel_size2':  [1, 3, 5, 8, 16, 32],
'pool_size2':  [3, 5, 8, 16, 32],
'strides2':  [1, 3, 5, 8],
'hidden_layer3':  [4, 8, 16, 32, 64, 128],
'drop_out3':  [0, 0.05, 0.1, 0.2, 0.3],
'filter_size3':  [4, 8, 16, 32, 64, 128],
'kernel_size3':  [1, 3, 5, 8, 16, 32],
'pool_size3':  [3, 5, 8, 16, 32],
'strides3':  [1, 3, 5, 8],
'hidden_layer4':  [4, 8, 16, 32, 64, 128],
'drop_out4': [0, 0.05, 0.1, 0.2, 0.3],
'batch_size':  [16, 32, 64, 128],
'learning_rate':  [0.1, 0.01, 0.001, 0.0001]
```

Fig. 7. List of hyperparameters and their values to iterate through.

Hyperparameter fine-tuning describes an extensive process of iterating thorough different combinations of hyperparameters in order to find an arrangement that achieves the best performance. This process is usually done using existing libraries that automate the process of finding the best combination of parameters. This automated process is used to save the user much time, as manually trying out different combinations is a grossly inefficient process. In this case, the *Hyperopt* python library was used to perform the hyper-parameter fine-tuning. *Figure 7* contains the list of hyperparameters and their values specified to iterate through by *Hyperopt*.

As an experiment, I first ran hyperparameter fine-tuning on the temperature classifier model for the archaea file to gauge the performance of the model running with the parameters determined to be optimal by *Hyperopt*. The results can be seen in *figure 8*. As can be seen, the hyperparameter fine-tuning

Temperature Difference	Archaea		
	Original Results	"Best" Parameters	Paper Results
>0°C	0.522	0.516	0.814
>5°C	0.629	0.63	0.889
>10°C	0.726	0.637	0.93
>20°C	0.888	0.837	0.902
>30°C	0.991	0.978	0.954

Fig. 8. Classification Comparison with Fine-tuned hyperparameters.

did not improve our accuracies. On top of this, the run for just this one experiment took 26 hours to complete. Due to the unsatisfactory performance and the demanding run-time,

I decided to move on to using transformer models to try and improve the accuracy of my implementation of the models.

Transformers

Transformers have a self-attention mechanism that allow them to capture long-range dependencies and model complex relationships in data effectively. It is for this reason that I decided to use a transformer model to attempt to improve the results of the original authors. Specifically, I used TabNet, a transformer that specializes in the reasoning form tabular data, that is, data that is organized in rows and columns, like our input CSV's.

TabNet

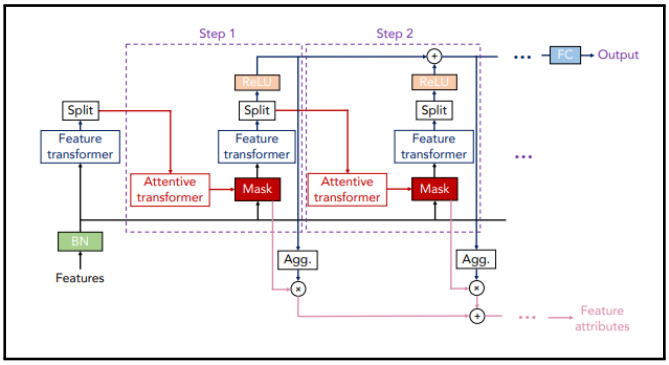

Fig. 9. General TabNet Architecture (3).

TabNet was developed by researchers at Google Cloud AI in 2020, for this paper, I used the *PyTorch* implementation of TabNet to perform both regression and classification. A high-level view of the TabNet architecture can be seen in *figure 9*. Without going into too much detail, TabNet inlcudes a *feature transformer*, consisting of four consecutive Gated Linear Units (GLU's). This output is fed into an *attentive transformer*, which in theory grants better learning and interpretability as it performs sparse feature selection. This is followed by a *mask*, which is used to develop what are called decision parameters that are then fed into the next step (3).

Results

Temperature Difference	Archaea		Bacteria		Archaea + Bacteria	
	TabNet Results	Paper Results	TabNet Results	Paper Results	TabNet Results	Paper Results
>0° C	0.404	0.814	0.326	0.756	0.329	0.728
>5° C	0.691	0.889	0.575	0.824	0.574	0.788
>10° C	0.802	0.93	0.6	0.83	0.671	0.853
>20° C	0.915	0.902	0.58	0.882	0.798	0.914
>30° C	0.974	0.954	0.614	0.921	0.899	0.95

Fig. 10. Classification Random Split TabNet Comparison.

As can be seen in *figure 10*, the performance of the transformer model was fairly similar to the archaea run, even outperforming the original paper results for the greater than 30 temperature difference. However, for the bacteria and the archaea & bacteria runs, the transformer did not do as well as the temperature classifier.

Regarding the regression results, as we mentioned earlier, the lower the RMSE and the higher the r^2, the better fit the model is. Similar to the results seen in *figure 4*, the transformer

Source	Archaea		Bacteria		Archaea + Bacteria	
	RMSE	r2	RMSE	r2	RMSE	r2
TabNet Results	6.04	0.966	5.64	0.546	8.09	0.780
Paper Results	8.06	0.862	6.76	0.818	7.31	0.875

Fig. 11. Regression Random Split TabNet Comparison.

model did better than the paper for the archaea run. However, it was not able to best the paper's results for the other two runs, performing about the same as my run of the logistic regression model.

Future Work

In the future, I hope to find better results by implementing TabNet from scratch based on the research paper rather than using a pre-built PyTorch architecture. I believe that building it from scratch will allow me to make the necessary modifications to make TabNet more applicable to the tRNA data that I am working with in this project. Additionally, I hope to discover what went wrong in my local run of the archaea phylogenetic distance split, I suspect that it could be an issue with the NCBI taxonomy tree that populates the distance matrix. I am also planning on collecting more organism data so that my files can get closer to the size of the files used by the authors. I plan on doing this autonomously via web-scraping tools, but this is a decent amount of more work that needs to be done in order for this to work. Since I did not do any improvements on the models that run off the phylogenetic split data, I plan on doing some edits to that code as well. Specifically, I plan on tweaking the clustering method used, to see if my results get closer to what the paper reported.

Conclusion

After my attempts, it seems that I was not able to replicate or best the accuracy presented by the source paper. However, after some analysis of my data and work, I am confident that given more time and experimentation, I would have been able to best the results of the authors. At the beginning, my limited knowledge of biology hindered me, but now that I have newfound knowledge, I believe that my future decisions of improving this paper will be better informed. In conclusion, this work taught me valuable lessons about working in bioinformatics, as exposed me to various new ideas both in the field of biology, artificial intelligence, and data analytics.

References

1. DCimen, Emre and Jensen, Sarah E and Buckler, Edward S Building a tRNA thermometer to estimate microbial adaptation to temperature *Nucleic Acids Research*, 2020.
2. Sauer, David B and Wang, Da-Neng Predicting the optimal growth temperatures of prokaryotes using only genome derived features *Bioinformatics*, 2019.
3. Sercan O. Arik and Tomas Pfister TabNet: Attentive Interpretable Tabular Learning *arXiv*, 2020