BACHELOR'S THESIS COMPUTING SCIENCE

RADBOUD UNIVERSITY NIJMEGEN

# Predicting DNA sequence pair similarity with HyenaDNA

*Leveraging foundational models for biological downstream tasks.*

*Author:*
Samuel Padron
s1059584

*First supervisor/assessor:*
Asst. Prof. Twan van Laarhoven

*Second assessor:*
Prof. Dr. Ir. Dick de Ridder

August 18, 2024

## Abstract

The rise of large language models (LLMs) has revolutionized the field of genomics by providing new tools for research and analysis. For example, traditional methods for detecting regions of similarity between genomes, which rely on local genome alignments, are often computationally expensive. This is particularly the case when dealing with whole genomes. To address this challenge, we propose the use of HyenaDNA, a foundation model pre-trained on the human genome, for the task of predicting DNA sequence similarity. By leveraging the embeddings generated by HyenaDNA within a Siamese neural network architecture, we aim to bypass the need for traditional alignment tools to directly predict DNA sequence pair similarity.

This study investigates the effectiveness of using HyenaDNA as a backbone for such a prediction task. The model demonstrates the ability to detect similar pairs of DNA sequences with an average test accuracy of 0.841 across various species, with a notably higher accuracy of 0.927 in chimpanzees (panTro6). Additionally, a regression variant of the model was tested to predict a similarity score for a pair of sequences, though it yielded unsatisfactory results. The promising outcomes of our multi-species evaluation, however, suggest that HyenaDNA has significant potential in genomic similarity studies, offering a more efficient alternative to conventional methods.

# Contents

# Chapter 1

# Introduction

The rise of large language models (LLMs) has led to advancements in the field of genomics. The development of DNA LLMs is aiding researchers in the study of homology by providing more sophisticated tools for analyzing genomic sequences. These LLMs and so called foundation models, which are capable of understanding the underlying "language" and patterns within genomic data, have the potential to evaluate the similarity of regions through evolution more effectively than traditional methods. These traditional methods for detecting similar sequences rely on local genome alignments, which involve aligning small regions of genomes to identify homologous sequences. However, when whole genomes are involved, these methods become computationally intensive and time-consuming. To make this process more efficient, we propose the use of a foundation model called HyenaDNA, an extensively pre-trained LLM which can be fine-tuned for a range of specific tasks with relatively minor adjustments, to help predict whether two sequences are similar or not. This approach aims to reduce the dependency on traditional alignment tools to evaluate similar regions between genomes through evolution.

We investigate the potential of HyenaDNA for the downstream task of predicting sequence similarity by using embeddings from HyenaDNA as inputs into a Siamese neural network. By using both aligning and non-aligning sequence pairs during training, we develop a model that essentially screens for sequence similarity, potentially serving as a preliminary step before employing more computationally intensive methods. This not only eliminates reliance on traditional tools, but also demonstrates HyenaDNA's ability to be used in genomics studies related to sequence similarity opens up new possibilities for larger scale studies.

Our contributions in this paper are the following:

- We provide a brief introduction to HyenaDNA, explaining its architecture and what makes it more suitable for the context of predicting local genome similarity than other existing models (Chapter 2).

- In Chapter 3 we provide the implementation of our deep learning model that uses a Siamese neural network architecture with HyenaDNA as a backbone layer to predict DNA sequence pair similarity.
- We experiment on the ability of the model to generalize on different species, showcasing HyenaDNA's ability to perform in a multi-species context.
- Turn our classification model into a regression model for predicting a DNA sequence similarity score by making minimal adjustments to its architecture.

# Chapter 2

# Preliminaries

Before moving on to the details of HyenaDNA, we give a short introduction to some key topics that the reader should have some understanding of before continuing. The topics covered in this section are DNA sequence alignments, Natural Language Processing (NLP), and foundation models. Lastly, we explain Siamese neural networks and how they fit within the context of the paper.

## 2.1 Genome Similarity

A genome consists of one or more chromosomes i.e. long DNA sequences[8], so the task of predicting genome similarity is that of determining the similarity between two DNA sequences. The similarity of two sequences is a measure determined by sequence alignment tools [7]. The term 'local' refers to a similarity measure calculated by an algorithm that performs local sequence alignment. A local alignment of two DNA sequences consists of considering segments of various lengths (regions) of the sequences and finding the one with the highest similarity measure. These alignments are made with sequence alignment programs which are described in more detail in Section 3.1.1.

## 2.2 Large Language Models and Foundation Models

Large Language Models (LLMs) are machine learning models based on deep learning techniques. This means they are neural networks with many layers which allow them to learn complex patterns in large datasets. The development of LLMs has been a significant milestone in NLP, enabling a wide range of applications from text generation to translation, sentiment analysis, and most relevant to this research, genomics.

The use of LLMs has greatly increased in the past years in the field of genomics. The creation of the Transformer architecture has allowed researchers to leverage the power of these new models for use in genomics applications. This has led to the creation of task-agnostic "foundation models" in genomics. The goal of such models is to learn generalizable features from unlabeled genome data so a researcher can later fine-tune it to their experiment's needs.

In the field of genomics, foundation models are increasingly being used to analyze and interpret genomic data [3, 4, 5]. By treating DNA sequences as a form of language, these models are able to generate embeddings that capture the functions and interactions of nucleotide sequences. These embeddings can then be used for various genomics tasks such as identifying genetic variants or understanding the functional structure of genes. The use of LLMs in this context leverages their ability to recognize complex patterns in large datasets, making them powerful tools for advancing research in genomics.

## 2.3 HyenaDNA

HyenaDNA is a genomic foundation model developed to address the limitations of previous genomics models [3, 4] that struggled with long-range dependencies and high computational costs [6]. HyenaDNA leverages the capabilities of Hyena, a model based on implicit convolutions, which allows HyenaDNA to handle long sequences efficiently (such as genomes which can be billions of nucleotides in length).

HyenaDNA can process sequences up to 1 million tokens long at single nucleotide resolution, significantly surpassing the capabilities of previous models that handled only up to 4,000 tokens [6]. This extended context length enables better modeling of long-range interactions within DNA sequences.

Unlike models that rely on tokenization methods, HyenaDNA maintains single nucleotide resolution, which allows it to capture more subtle genetic variations that can have significant biological implications. In addition, HyenaDNA's architecture scales sub-quadratically with sequence length, making it much faster and more efficient than traditional Transformer-based models. For instance, it can train up to 160 times faster than Transformers on long sequences [6]. These key factors are why HyenaDNA has been selected as the backbone model we build our prediction model with.

Another feature of HyenaDNA is that it can adapt to new tasks without requiring updates to its pre-trained weights, a significant advantage for rapidly evolving fields like genomics where new data and tasks frequently emerge. This feature of HyenaDNA is what allows us to utilize it for predicting DNA sequence similarity.

## 2.4 Siamese Neural Networks

A Siamese Neural Network (SNN) is a type of neural network which has two inputs and one output value which indicates the similarity of the two inputs [2]. The architecture of SNNs consist of two or more sub-networks which each process one of the inputs to the model. The outputs of these sub-networks are then usually combined using a distance metric to produce a similarity score. In our model we use instances of HyenaDNA as our sub-networks which give us embeddings that we then process to get an output value. Since we are determining whether two sequences align or not, we replace the usual distance metric with a value which indicates the probability that the two sequences align.

# Chapter 3

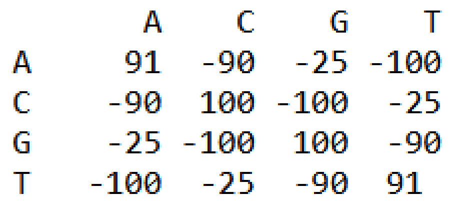
# Methodology

## 3.1 Data

### 3.1.1 Data Collection

To train the model to predict whether two sequences are similar we make use of pairwise sequence alignments. These alignments come from the University of California Santa Cruz (UCSC) Genome Browser, in which one can find a multitude of genomes belonging to different species. For the purpose of this paper, we use the pairwise alignments from the hg38 (human) and galGal6 (chicken) genomes which are aligned using the LASTZ alignment program. This program pre-processes the target sequences from hg38 and aligns the query sequences from galGal6 to them using the scoring matrix seen below in Figure 3.1. We select the hg38 and galGal6 genomes to ensure the sequences in a pair come from two species that are not too close in terms of evolutionary distance. This is important to avoid the risk of our model learning to distinguish only between closely related species (such as a human and gorilla) which might limit our model's ability to generalize to pairs of sequences from more distantly related species. In Section 4.2 we explore how the model performs when trained on more species to evaluate its performance when trained with species of increasing evolutionary distance from humans.

The pairwise alignments come in a Multiple Alignment Format (MAF) file which consists of a series of alignment blocks. Each block consists of an alignment block line which indicates the start of the block and includes a score variable which indicates the score that was assigned to the pair alignment. The alignment block line is then followed by two sequence lines (one for each sequence in the pair in the alignment).



|   | A | C | G | T |
|---|---|---|---|---|
| A | 91 | -90 | -25 | -100 |
| C | -90 | 100 | -100 | -25 |
| G | -25 | -100 | 100 | -90 |
| T | -100 | -25 | -90 | 91 |

Figure 3.1: Scoring matrix used by LASTZ for aligning galGal6 sequences with hg38 sequences [9]

### 3.1.2 Data Processing

Before using the hg38 and galGal6 pairwise alignments, some processing has to be done in order to train the model. The MAF file has to first be converted to a CSV file. This allows us to be able to write a script using the Pandas library to parse the alignment blocks in the MAF file into tensors that our model can use for training. The alignment blocks are parsed in such a way that each pair of sequences is represented in one line with four values: an ID for the pair, the blastz score, the first sequence of the pair, and the second sequence. The alignment blocks that are longer than 5,000 nucleotides are split into smaller blocks of 5,000. This allows us to to train the model using a single GPU and remove the necessity of setting up a distributed training strategy.

Another important step in data collection is to get non-aligning pairs of sequences to serve as negative samples when training. This is accomplished by taking the aforementioned pairwise sequence alignments and making a new CSV file in which the second sequence of a pair (the sequence from the galGal6 genome) gets swapped with the second sequence of the next pair. Although there is a theoretical chance that the resulting pairs contain similar sequences after the swap, this is highly unlikely in practice. As evidenced by our results, this simple approach proves to be effective since the model, trained with these negative samples, successfully generalizes and predicts sequence similarity for other species' genomes. This indicates that this method of creating negative samples allows the model to distinguish between aligning and non-aligning sequence pairs accurately.

After making the two CSV files from the MAF file obtained in UCSC Genome Browser, they are then concatenated to create a single CSV file containing both positive and negative samples. This final CSV file is the file we use to create the dataset that we train the model on.

## 3.2 Model Architecture

### 3.2.1 Overview

we use a Siamese neural network architecture with HyenaDNA as a backbone model. When a pair of sequences is retrieved from the dataset, we individually run each of them through the HyenaDNA model to get an embedding for each of them. We then pass these embeddings through a head module which turns these embeddings into a prediction. The final layer has an output of size one, a number that represents the probability that the sequences align. Our model is built using PyTorch v2.2.1 and PyTorch Lightning v1.8.6. A diagram of the model is provided in Appendix A.

### 3.2.2 Backbone

We use HyenaDNA as a backbone model since it allows us to extract meaningful features from the input sequences. The backbone is configured with the following default hyperparameters:

- `d_model`: The dimensionality of the model (hidden size) : 256.
- `n_layer`: The number of layers in the model : 4.
- `d_inner`: The dimensionality of the inner feed-forward layers : 1024.
- `vocab_size`: The size of the vocabulary, adjusted to be a multiple of a specified padding size : 12.
- `resid_dropout`: Dropout probability for the residual connections : 0.0.
- `embed_dropout`: Dropout probability for the embedding layers : 0.1.
- `layer_norm_epsilon`: Epsilon value for layer normalization :

$$1.00 \times 10^{-5}$$

During training, the backbone's parameters are frozen since we only want the embeddings of the input sequences from the pre-trained HyenaDNA model without additional fine-tuning. This results in a faster training process since the model does not need to train on the 3.3M parameters of the HyenaDNA backbone model.

### 3.2.3 Prediction Head

The second component of our model is the `PredictionHead` component. This component processes the output from the backbone and predicts the similarity between input sequences. The architecture of the `PredictionHead` is as follows:

- **Initialization:**
  - `input_size`: The number of input features.
  - `hidden_size`: The size of the hidden layer.
  - `dropout_prob`: The dropout probability (set to 0.5 by default).
- **Layers:**
  - `self.conv1d`: A 1D convolutional layer that processes the input sequences.
  - `self.fc1`: A fully connected layer that takes the concatenated input features and reduces them to the hidden size.
  - `self.fc2`: A fully connected layer that further processes the hidden representation.
  - `self.fc_out`: The output layer that reduces the hidden representation to a single value.
  - `self.dropout`: A dropout layer added between the fully connected layers to prevent overfitting.
- **Forward Pass:**

  We describe the the forward pass of a pair of sequences through the model to illustrate how a pair of DNA sequences get reduced to the single value indicating the probability that the sequences align. Note that for the description of the tensor shapes we have batch size $B = 16$, sequence length $L = 5,000$, hidden size of HyenaDNA $H = 256$, and output dimensions of the fully connected layers $F1, F2 = 256$.

| **Step** | **Tensor Shape** |
|---|---|
| Initial Tensors `seq1`, `seq2` | $(B, L, H)$ |
| Pass through 1D convolutional layer `self.conv1d` | $(B, L, H) \rightarrow (B, H, L)$ |
| Mean of the convolutional output across sequence length dimension | $(B, H, L) \rightarrow (B, H)$ |
| Compute absolute difference and element-wise product of `seq1` and `seq2` | $(B, H), (B, H)$ |
| Concatenate `seq1`, `seq2`, absolute difference, and element-wise product | $4 \cdot (B, H) \rightarrow (B, 4H)$ |
| Pass through fully connected layers (`self.fc1` and `self.fc2`) with activations and dropout | $(B, 4H) \rightarrow (B, F1) \rightarrow (B, F2)$ |
| Final output through `self.fc_out` | $(B, F2) \rightarrow (B, 1)$ |

## 3.3 Training and Validation

To train the model, the CSV file described in Section 3.1.2 is split into a training, validation, and test sets with a 70-15-15 ratio. We then perform random shuffling on the data as the original dataset contains all positive samples first followed by the negative ones.

Since the output of our model is a tensor containing the probability of each pair of sequences in the batch, we calculate the loss by passing these probabilities through the sigmoid function to get a prediction for each pair and compare it to the pair's label. We use binary cross entropy as our loss function as well as the Adam optimizer to update the weights of the network after each epoch.

## 3.4 Regression Model

In order to further see how suitable HyenaDNA is for predicting DNA sequence pair similarity, we make a regression model based on our classification model in order to predict a similarity score. To make the model we simply include the blastz score from each pair of sequences in the MAF file as a new feature to our dataset. This score is then normalized using the length of the largest sequence in the pair. Since we are now predicting a score, this means that the last output layer of our model does not output the probability for whether a pair aligns or not, but instead directly outputs the similarity score for the pair. For this reason, we use the Means Squared Error (MSE) as our loss function to replace Binary Cross Entropy. The results of this version of our model are shown in Section 4.3.

## 3.5 Training Resources

Both the classification and regression models were trained using a single RTX 2080ti GPU with 16 GB of VRAM of a computer cluster node running on a Xeon 4212 processor and 10 GB of memory.

# Chapter 4

# Results

To predict whether two sequences are similar, we use HyenaDNA as the backbone of a Siamese neural network. In our model, the embeddings generated by HyenaDNA serve as features to predict the probability that two sequences are similar. Here, we present results for the model trained with the hg38 (human) and galGal6 (chicken) genomes, here onwards referred to as the "baseline model", to get an idea of how well the model performs. In Section 4.2 we investigate how well the model performs when it is trained on other species. Lastly, in Section 4.3 the results of the regression model from Section 3.4 are shown. All collected training and evaluation metrics are shown in Appendix B.

## 4.1 Baseline Model Performance Evaluation

### 4.1.1 Training and Validation

We train the Siamese network and keep track of the validation set loss and accuracy for ten epochs in order to get a baseline performance for comparison.

We obtain an average training loss of 0.576. As for validation, the average loss obtained is 0.432 while achieving an accuracy of 0.842 with a standard deviation of 0.0197. Figure 4.1 shows the model's validation accuracy over training epochs in one run. We can see that the model flattens out after the third epoch, indicating that increasing the number of epochs will not help the model learn better.

### 4.1.2 Testing

We evaluate the model using the accuracy, precision, recall and F1-score metrics. The model was tested ten times in order to get a mean for each metric score. The mean test accuracy achieved by the model is 0.841. The

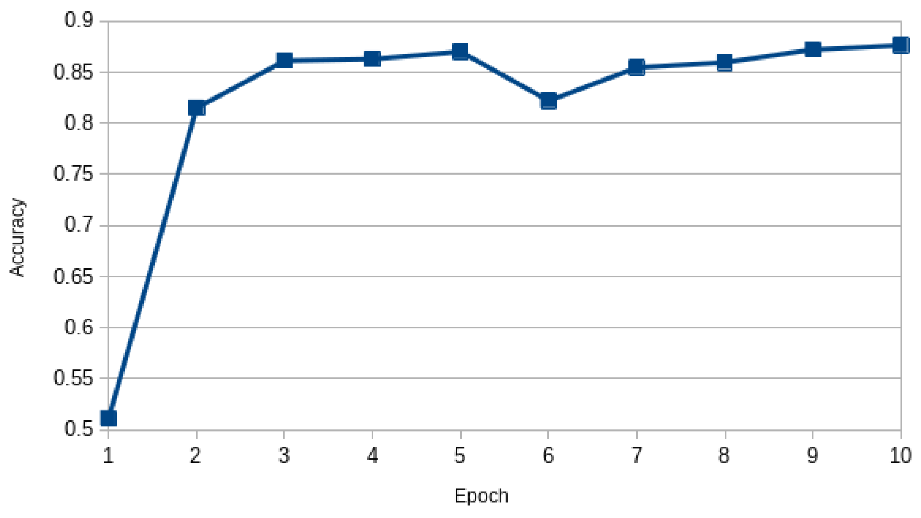
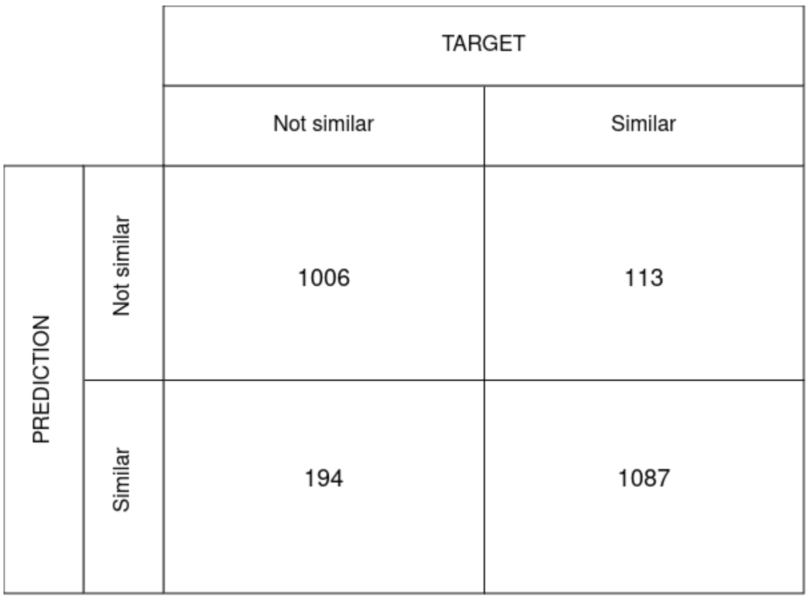


Figure 4.1: Validation accuracy as a function of the number of training epochs.

| | | TARGET | |
|---|---|---|---|
| | | Not similar | Similar |
| PREDICTION | Not similar | 1016 | 113 |
| | Similar | 184 | 1087 |

Figure 4.2: Confusion matrix of the baseline model on the training set.

value for the other metrics are 0.856 for precision, 0.823 for recall, and 0.837 for the F1-score.



| | | TARGET | |
|---|---|---|---|
| | | Not similar | Similar |
| PREDICTION | Not similar | 1006 | 113 |
| | Similar | 194 | 1087 |

Figure 4.3: Confusion matrix of the baseline model on a test run.

### 4.1.3 Hyperparameter Tuning

Next, we attempt to optimize the hyperparameters of our baseline model to see to what extent performance could be improved. To this end, we use the Optuna framework. By performing hyperparameter optimization, we aim to eliminate any human biases when choosing the original hyperparameters. Optuna employs a Random Search strategy to find the optimal hyperparameters, which has been demonstrated to be highly effective due to its ability to explore the hyperparameter space without the biases associated with grid search methods [1].

The hyperparameter search space was created using the following ranges:

- **Learning Rate:** $[1 \times 10^{-5}, 1 \times 10^{-3}]$, using a logarithmic scale
- **Weight Decay:** $[1 \times 10^{-6}, 1 \times 10^{-2}]$, also on a logarithmic scale
- **Dropout:** $[0.2, 0.5]$
- **Number of Layers:** $[1, 4]$
- **Number of Units per Hidden Layer:** $[256, 1024]$

These hyperparameters were selected due to their likely high impact on model performance and generalization.

After conducting twenty trials, we observe the highest validation accuracy of 0.868 in Trial 3 and the lowest in Trial 5 with a value of 0.815. All other trials after Trial 5 are pruned by the optimization framework. The optimal hyperparameters found during Trial 3 are as follows:

- **Learning Rate:** $6.1 \cdot 10^{-4}$
- **Weight Decay:** $9.1 \cdot 10^{-5}$
- **Dropout:** 0.33878600984991813
- **Number of Layers:** 1
- **Number of Units per Hidden Layer:** 516

After running a study on the hyperparameters, the best hyperparameters found do not improve our model's performance but actually decrease it. This means that our initial values for our chosen hyperparameters of our baseline model are more effective in achieving higher validation accuracy.

### 4.1.4 Training Set Size

To investigate the importance of the training set size to determine whether more data might improve performance, we train the model using subsets of the original training dataset:

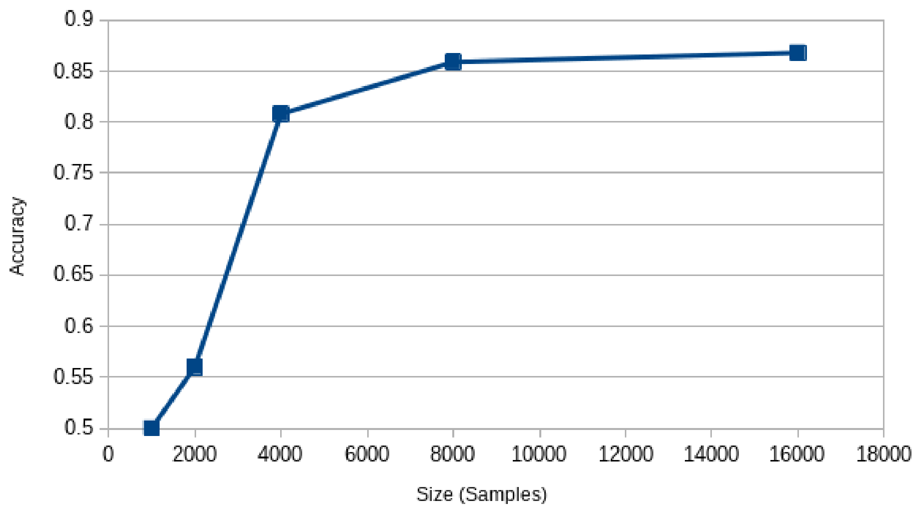


Figure 4.4: Average validation accuracy as a function of the number of training samples.

The model's accuracy improves significantly as the size of the training dataset increases, especially when the dataset size is relatively small. However, the rate of improvement slows down as the dataset size continues to grow. After 4,000 samples, the gains in accuracy begin to lower. This is shown in the improvements between 4,000 to 8,000 samples and 8,000 to 16,000 samples. Based on this trend, while training on larger training datasets might yield an improvement in the model's performance, it will likely be marginal. For practical purposes we conclude that 16,000 samples is indeed sufficient and that increasing the number of samples will not improve our model's performance.

## 4.2 Multi-Species Evaluation

With our evaluation of the baseline model complete, we train and test the model on datasets from different species apart from galGal6 to show that the model's architecture generalizes to other species. This is important to determine that the model's performance is not reliant on a single training dataset. The chosen species are panTro6 (chimpanzee), susScr11 (pig), xenTro10 (tropical clawed frog), and danRer10 (zebrafish). These species are chosen based on their increasing evolutionary distance from the human species. The model is trained and tested ten times to get an average of each evaluation metric as well as the validation accuracy for comparison with the baseline model.

| **Species** | **Accuracy** | **Precision** | **Recall** | **F1-score** |
|---|---|---|---|---|
| panTro6 | 0.927 | 0.948 | 0.912 | 0.928 |
| susScr11 | 0.841 | 0.889 | 0.789 | 0.831 |
| galGal6 | 0.841 | 0.856 | 0.823 | 0.837 |
| xenTro10 | 0.853 | 0.863 | 0.843 | 0.850 |
| danRer10 | 0.838 | 0.864 | 0.807 | 0.832 |

Table 4.1: Test results for chosen species

The results in Table 4.1 indicate that the model performs well across all tested species, demonstrating its ability to generalize beyond the initial training dataset. We can see the model achieves the highest accuracy and F1-score with panTro6 (chimpanzee), with an accuracy of 0.927 and an F1-score of 0.928. This performance is expected, given the close evolutionary relationship between humans and chimpanzees, which likely leads to more similar sequence patterns.

For the other species, the accuracy ranges from 0.838 (danRer10) to 0.853 (xenTro10), with corresponding F1-scores ranging from 0.832 to 0.850. Although these results are slightly lower than those observed with panTro6, they still indicate strong performance, suggesting that the model effectively captures relevant features across a diverse set of species.

Overall, these results confirm that the model's architecture is robust and capable of generalizing across species with varying evolutionary proximity to the human species. The relatively consistent performance across different species highlights the model's potential applicability in a broad range of biological contexts, making it a valuable tool for research in the field of genomics.

## 4.3 Regression Model

We show the results of the adaption our classification model into a regression model in order to predict similarity scores for sequence pairs.

### 4.3.1 Evaluation Metrics

Like the previous versions, we run the regression model ten times, using the Mean Squared Error (MSE) as our evaluation metric. In these ten runs the model achieves an average MSE of 294.0684 with a standard deviation of 28.971 which indicates some variability in the model's performance.

In a random batch sample containing target we observe that the similarity scores land in the range $(-5, 75)$. Given this range, an MSE value of 294.0684 is quite considerable, suggesting that this model's predictions are significantly inaccurate. Since a high MSE indicates that the predicted similarity scores deviate considerably from the true values. This result highlights the need for improvements to reduce the model's error.

# Chapter 5

# Conclusion

We have created a model that uses HyenaDNA as a backbone to classify whether a pair of sequences are similar or not. Despite its relatively simple architecture, the model consistently demonstrates adequate performance, achieving an average test accuracy of 0.841. In our multi-species experiments we similarly obtain average test accuracies around 0.850 with the panTro6 model being an exception with an accuracy of 0.927.

Furthermore, we extended our classification model into a regression framework to predict similarity scores between DNA sequences. Although the regression model did not produce satisfactory results due to the high average MSE value obtained, our effort highlights the need for more extensive research in how HyenaADNA may be leveraged for such a task.

The consistent performance of our baseline model, particularly its ability to generalize across species, suggests that HyenaDNA has significant potential as a tool for predicting local genome similarity. These findings suggests that further investigation into its application in comparative genomics and other related fields is warranted. Future work could focus on improving the models using more complex architectures and more computational resources for training to see how HyenaDNA would perform in state-of-the-art research. Furthermore, expanding the scope of species evaluated to plants and fungi could help us further explore the full potential of HyenaDNA in genomic analysis.

# Bibliography

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631, 2019.

[2] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. In J. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6. Morgan-Kaufmann, 1993.

[3] Hugo Dalla-Torre, Liam Gonzalez, Javier Mendoza Revilla, Nicolas Lopez Carranza, Adam Henryk Grzywaczewski, Francesco Oteri, Christian Dallago, Evan Trop, Hassan Sirelkhatim, Guillaume Richard, Marcin Skwark, Karim Beguir, Marie Lopez, and Thomas Pierrot. The nucleotide transformer: Building and evaluating robust foundation models for human genomics. *bioRxiv*, 2023.

[4] Yanrong Ji, Zhihan Zhou, Han Liu, and Ramana V Davuluri. DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome. *Bioinformatics*, 37(15):2112–2120, 02 2021.

[5] Alexander Karollus, Johannes Hingerl, Dennis Gankin, Martin Grosshauser, Kristian Klemon, and Julien Gagneur. Species-aware dna language models capture regulatory elements and their evolution. *Genome Biology*, 25(1):83, Apr 2024.

[6] Eric Nguyen, Michael Poli, Marjan Faizi, Armin Thomas, Callum Birch-Sykes, Michael Wornow, Aman Patel, Clayton Rabideau, Stefano Massaroli, Yoshua Bengio, Stefano Ermon, Stephen A. Baccus, and Chris Ré. Hyenadna: Long-range genomic sequence modeling at single nucleotide resolution, 2023.

[7] William R Pearson. An introduction to sequence similarity ("homology") searching. *Curr. Protoc. Bioinformatics*, Chapter 3(1):3.1.1–3.1.8, June 2013.

[8] Stephanie Clare Roth. What is genomic medicine? *J. Med. Libr. Assoc.*, 107(3):442–448, July 2019.

[9] Scott Schwartz, W James Kent, Arian Smit, Zheng Zhang, Robert Baertsch, Ross C Hardison, David Haussler, and Webb Miller. Human-mouse alignments with BLASTZ. *Genome Res.*, 13(1):103–107, January 2003.

# Appendix A

# Model Diagram

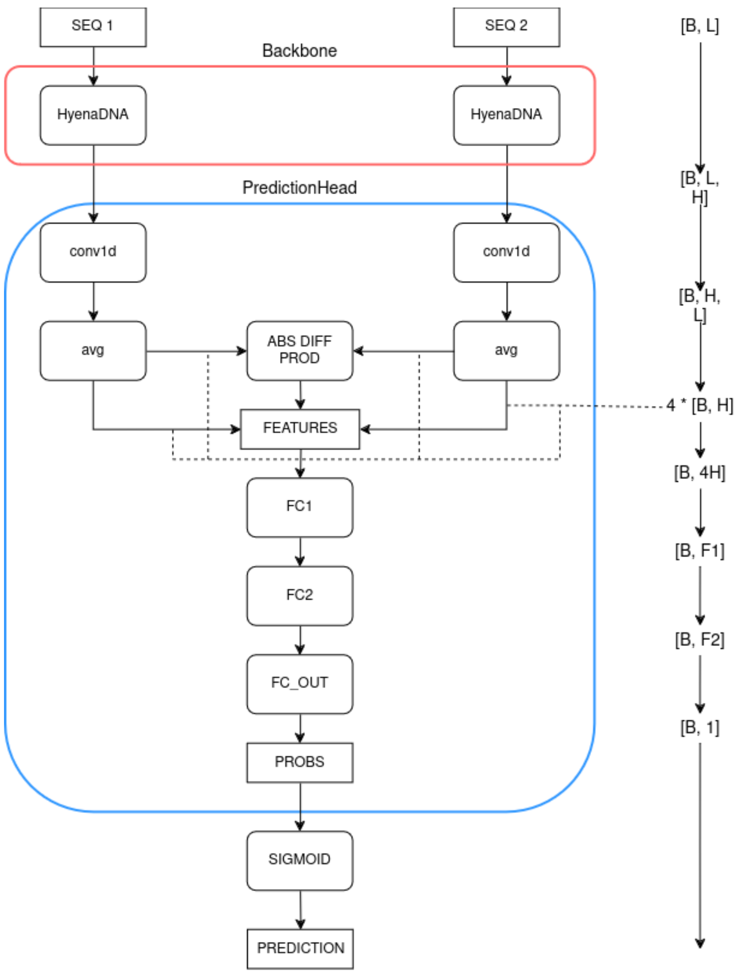


Figure A.1: Model diagram of our classification Siamese neural network

# Appendix B

# Training and Evaluation Metrics

We provide the various training and evaluation metrics for all runs of the different models in the paper. Section 1 shows the training and validation metrics for the multi-species models. Section 2 and 3 show the evaluation metrics for the multi-species and regression versions respectively. Note that the baseline model is included in the multi-species version under galGal6.

## B.1 Multi-Species Training and Validation Results

| Run | Validation Accuracy | Validation Loss | Training Loss |
|---|---|---|---|
| 1 | 0.842 | 0.242 | 0.466 |
| 2 | 0.931 | 0.216 | 0.467 |
| 3 | 0.938 | 0.183 | 0.446 |
| 4 | 0.937 | 0.198 | 0.420 |
| 5 | 0.923 | 0.232 | 0.470 |
| 6 | 0.933 | 0.215 | 0.460 |
| 7 | 0.943 | 0.186 | 0.311 |
| 8 | 0.934 | 0.185 | 0.403 |
| 9 | 0.935 | 0.226 | 0.409 |
| 10 | 0.933 | 0.272 | 0.524 |

Table B.1: Training and validation metrics for panTro6

| **Run** | **Validation Accuracy** | **Validation Loss** | **Training Loss** |
|---|---|---|---|
| 1 | 0.822 | 0.405 | 0.585 |
| 2 | 0.833 | 0.374 | 0.490 |
| 3 | 0.793 | 0.432 | 0.556 |
| 4 | 0.844 | 0.378 | 0.604 |
| 5 | 0.853 | 0.362 | 0.538 |
| 6 | 0.846 | 0.390 | 0.532 |
| 7 | 0.863 | 0.360 | 0.509 |
| 8 | 0.869 | 0.351 | 0.494 |
| 9 | 0.877 | 0.347 | 0.561 |
| 10 | 0.830 | 0.399 | 0.458 |

Table B.2: Training and validation metrics for susScr11

| **Run** | **Validation Accuracy** | **Validation Loss** | **Training Loss** |
|---|---|---|---|
| 1 | 0.855 | 0.398 | 0.418 |
| 2 | 0.833 | 0.410 | 0.590 |
| 3 | 0.849 | 0.725 | 0.725 |
| 4 | 0.855 | 0.373 | 0.558 |
| 5 | 0.858 | 0.418 | 0.858 |
| 6 | 0.852 | 0.394 | 0.587 |
| 7 | 0.812 | 0.388 | 0.485 |
| 8 | 0.850 | 0.409 | 0.533 |
| 9 | 0.855 | 0.402 | 0.479 |
| 10 | 0.803 | 0.403 | 0.523 |

Table B.3: Training and validation metrics for galGal6

| **Run** | **Validation Accuracy** | **Validation Loss** | **Training Loss** |
|---|---|---|---|
| 1 | 0.861 | 0.399 | 0.452 |
| 2 | 0.852 | 0.396 | 0.529 |
| 3 | 0.857 | 0.394 | 0.417 |
| 4 | 0.860 | 0.401 | 0.524 |
| 5 | 0.832 | 0.423 | 0.489 |
| 6 | 0.837 | 0.419 | 0.484 |
| 7 | 0.863 | 0.398 | 0.568 |
| 8 | 0.854 | 0.434 | 0.602 |
| 9 | 0.849 | 0.392 | 0.696 |
| 10 | 0.866 | 0.413 | 0.527 |

Table B.4: Training and validation metrics for xenTro10 training runs

| **Run** | **Validation Accuracy** | **Validation Loss** | **Training Loss** |
|---|---|---|---|
| 1 | 0.840 | 0.419 | 0.592 |
| 2 | 0.861 | 0.411 | 0.474 |
| 3 | 0.839 | 0.398 | 0.564 |
| 4 | 0.844 | 0.447 | 0.545 |
| 5 | 0.844 | 0.423 | 0.470 |
| 6 | 0.849 | 0.389 | 0.489 |
| 7 | 0.804 | 0.425 | 0.493 |
| 8 | 0.842 | 0.435 | 0.553 |
| 9 | 0.818 | 0.431 | 0.570 |
| 10 | 0.844 | 0.399 | 0.418 |

Table B.5: Training and validation metrics for danRer10

## B.2 Multi-Species Evaluation Results

| Run | Test Accuracy | Precision | Recall | F1-score |
|---|---|---|---|---|
| 1 | 0.850 | 0.989 | 0.774 | 0.868 |
| 2 | 0.930 | 0.945 | 0.913 | 0.929 |
| 3 | 0.939 | 0.936 | 0.942 | 0.939 |
| 4 | 0.941 | 0.950 | 0.931 | 0.940 |
| 5 | 0.936 | 0.950 | 0.921 | 0.935 |
| 6 | 0.938 | 0.941 | 0.935 | 0.938 |
| 7 | 0.942 | 0.940 | 0.943 | 0.942 |
| 8 | 0.932 | 0.951 | 0.910 | 0.930 |
| 9 | 0.930 | 0.929 | 0.930 | 0.930 |
| 10 | 0.933 | 0.945 | 0.919 | 0.932 |

Table B.6: Evaluation metric results for panTro6 testing runs

| Run | Test Accuracy | Precision | Recall | F1-score |
|---|---|---|---|---|
| 1 | 0.814 | 0.922 | 0.733 | 0.812 |
| 2 | 0.830 | 0.896 | 0.748 | 0.815 |
| 3 | 0.802 | 0.932 | 0.651 | 0.766 |
| 4 | 0.847 | 0.895 | 0.785 | 0.837 |
| 5 | 0.859 | 0.882 | 0.828 | 0.854 |
| 6 | 0.835 | 0.887 | 0.768 | 0.824 |
| 7 | 0.873 | 0.817 | 0.907 | 0.859 |
| 8 | 0.852 | 0.854 | 0.884 | 0.869 |
| 9 | 0.876 | 0.886 | 0.863 | 0.874 |
| 10 | 0.825 | 0.915 | 0.718 | 0.804 |

Table B.7: Evaluation metric results for susScr11 testing runs

| **Run** | **Test Accuracy** | **Precision** | **Recall** | **F1-score** |
|---|---|---|---|---|
| 1 | 0.839 | 0.848 | 0.823 | 0.837 |
| 2 | 0.843 | 0.878 | 0.797 | 0.835 |
| 3 | 0.861 | 0.849 | 0.877 | 0.863 |
| 4 | 0.862 | 0.863 | 0.798 | 0.829 |
| 5 | 0.854 | 0.860 | 0.858 | 0.859 |
| 6 | 0.842 | 0.873 | 0.706 | 0.781 |
| 7 | 0.812 | 0.869 | 0.853 | 0.861 |
| 8 | 0.836 | 0.827 | 0.895 | 0.860 |
| 9 | 0.860 | 0.811 | 0.891 | 0.849 |
| 10 | 0.802 | 0.886 | 0.730 | 0.800 |

Table B.8: Evaluation metric results for galGal6 testing runs

| **Run** | **Test Accuracy** | **Precision** | **Recall** | **F1-score** |
|---|---|---|---|---|
| 1 | 0.869 | 0.874 | 0.863 | 0.868 |
| 2 | 0.858 | 0.834 | 0.894 | 0.863 |
| 3 | 0.870 | 0.853 | 0.895 | 0.874 |
| 4 | 0.870 | 0.839 | 0.915 | 0.876 |
| 5 | 0.825 | 0.904 | 0.728 | 0.806 |
| 6 | 0.822 | 0.882 | 0.743 | 0.807 |
| 7 | 0.846 | 0.886 | 0.794 | 0.837 |
| 8 | 0.860 | 0.817 | 0.928 | 0.869 |
| 9 | 0.851 | 0.871 | 0.824 | 0.847 |
| 10 | 0.858 | 0.870 | 0.843 | 0.856 |

Table B.9: Evaluation metric results for xenTro10 testing runs

| **Run** | **Test Accuracy** | **Precision** | **Recall** | **F1-score** |
|---|---|---|---|---|
| 1 | 0.834 | 0.852 | 0.809 | 0.830 |
| 2 | 0.856 | 0.841 | 0.878 | 0.859 |
| 3 | 0.844 | 0.869 | 0.810 | 0.838 |
| 4 | 0.835 | 0.802 | 0.890 | 0.845 |
| 5 | 0.853 | 0.900 | 0.795 | 0.844 |
| 6 | 0.835 | 0.885 | 0.769 | 0.823 |
| 7 | 0.803 | 0.903 | 0.678 | 0.775 |
| 8 | 0.839 | 0.807 | 0.891 | 0.847 |
| 9 | 0.829 | 0.901 | 0.739 | 0.812 |
| 10 | 0.848 | 0.877 | 0.810 | 0.842 |

Table B.10: Evaluation metric results for danRer10 testing runs

## B.3 Regression Model Results

| **Run** | **MSE** |
|---|---|
| 1 | 262.466 |
| 2 | 311.496 |
| 3 | 338.037 |
| 4 | 332.906 |
| 5 | 304.686 |
| 6 | 268.113 |
| 7 | 267.090 |
| 8 | 278.551 |
| 9 | 266.347 |
| 10 | 310.992 |

Table B.11: MSE for regression model runs