Zama's Bounty: Implement an FHE-based Biological Age and Aging Pace Estimation ML Model

By Iria Quintero García (iria.quintero01@estudiant.upf.edu)

1. Introduction

There are multiple biological clocks that have been developed. However, they differ in important ways, such as the type of tissue they used for training, the health status of the individuals and the variable used as ground truth. There are also multiple publicly available datasets that you can use for testing. I downloaded GSE40279 and GSE55763, which consist of DNA methylation data obtained from whole blood samples.

While it might seem beneficial to build a better clock than existing ones, these models are already highly optimized. They were trained on large datasets using strong statistical methods, and many researchers have tried to improve them with only small gains. In most cases, using fewer CpGs or a different algorithm means losing some accuracy.

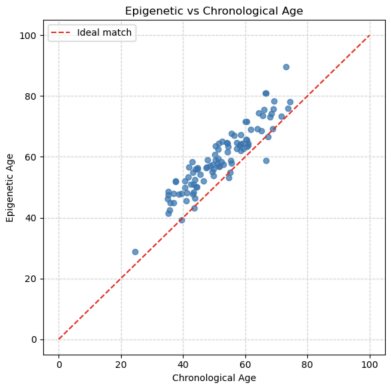
While Hannum's model was originally trained on whole blood from the general adult population, Horvath was trained with samples from 51 different tissues and cell types, which makes it more diverse but also introduces inter-tissue variability. Both used chronological age as ground truth. Previous studies have shown that both clocks perform similarly when predicting chronological age in blood-derived DNA, making Hannum's model a valid and effective choice in this context.

2. FHE Integration and Model Evaluation

First, I tried plaintext inference with both datasets to assess if the results I was getting were in the expected range, and since GSE55763 includes chronological age metadata, I was able to compare the predictions against the chronological age.

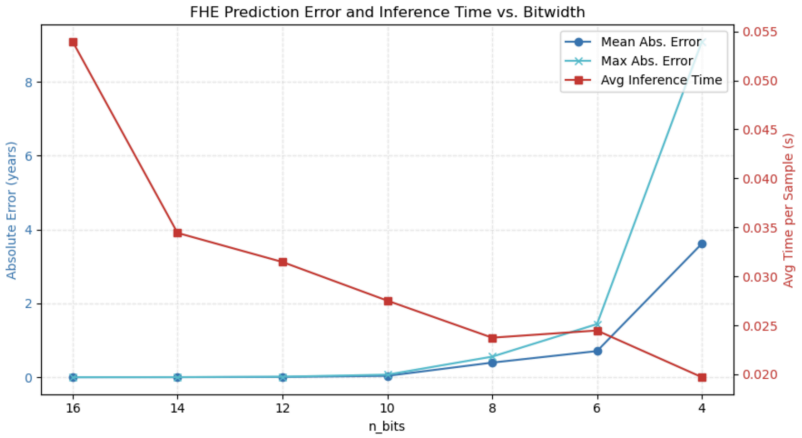
Then, I manually scaled the inputs and model parameters by a factor of 1,000 and initially used 16 bits for representation. Since fitting is required prior to compilation, I simulated the target variable y_train by applying Hannum's original formula, a linear combination of the scaled CpG beta values with the known coefficients and intercept. This ensured that the fitted model would reproduce Hannum's exact weights. Once fitted, the model was compiled and saved using FHEModelDev(...).save() to prepare it for encrypted inference.

After this, I tried encrypted inference with data from GSE40279 to check if the outputs were in the expected range. Once I checked this, I wanted to compare the outputs with the chronological age, so I did inference over 100 samples of GSE55763 while measuring the time and plotted the output with the chronological age, obtaining the following results.



Step	Average Time (s) ± Std
Cleartext (s)	0.0000 ± 0.0000
Encrypt (s)	0.0098 ± 0.0018
FHE Inference (s)	0.0102 ± 0.0018
Decrypt (s)	0.0008 ± 0.0000

Then, I benchmarked how the choice of quantization precision (n_bits) affects the performance of the encrypted model in terms of both accuracy and inference time. For each bitwidth (from 16 down to 4), I retrained and compiled a new Concrete ML model, exported it, and ran encrypted inference on the test samples using the FHE client-server interface. For each bit setting, I measured the average and maximum absolute errors between the encrypted and reference predictions, as well as the time required to process each sample. The results are summarized below.



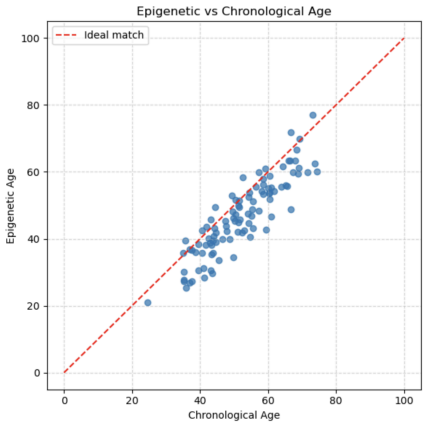
To further investigate inference performance, I compared predictions obtained through three different approaches: standard plaintext inference, FHE inference using the Concrete ML client-server model, and FHE inference using circuits compiled manually with Concrete Numpy. For this, I ran all three methods on the same 20 input samples and measured both the prediction errors wrt to the plaintext output and the inference times. The manually compiled Concrete Numpy circuit obtained more accurate (lower mean and maximum absolute errors) and faster results than the Concrete ML approach. However, while Concrete Numpy shows promising results in local benchmarks, I was unable to find a straightforward method to export or deploy the circuit. Therefore, despite its lower performance, the Concrete ML model was used in the final application deployed on Hugging Face.

```
Inference Comparison Summary with 20 samples
--------------------------------------------------
Mean Absolute Error (Concrete ML):     0.1102 years
Max Absolute Error  (Concrete ML):     0.1702 years
Mean Absolute Error (Concrete-Numpy):  0.0771 years
Max Absolute Error  (Concrete-Numpy):  0.1264 years
--------------------------------------------------
Avg Time/sample (Plaintext):           0.000009 seconds
Avg Time/sample (Concrete ML FHE):     0.034410 seconds
Avg Time/sample (Concrete-Numpy FHE):  0.019186 seconds
```

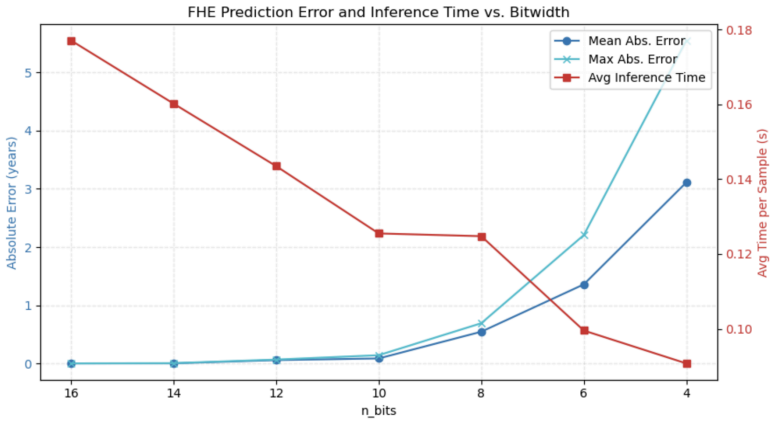
All corresponding steps and results can be found in the notebook hannum_fhe.ipynb.

3. Extending to Phenoage and Interpreting Results

I then repeated the entire process for the PhenoAge model. The steps and results are documented in phenoage_fhe.ipynb.



Step	Average Time (s) ± Std
Cleartext (s)	0.0001 ± 0.0000
Encrypt (s)	0.0602 ± 0.0025
FHE Inference (s)	0.0647 ± 0.0029
Decrypt (s)	0.0010 ± 0.0002



In this case, n_bits 12 provided the best trade-off between precision and inference speed.

Then, I compared it with inference directly using Concrete Numpy, and the results are shown below.

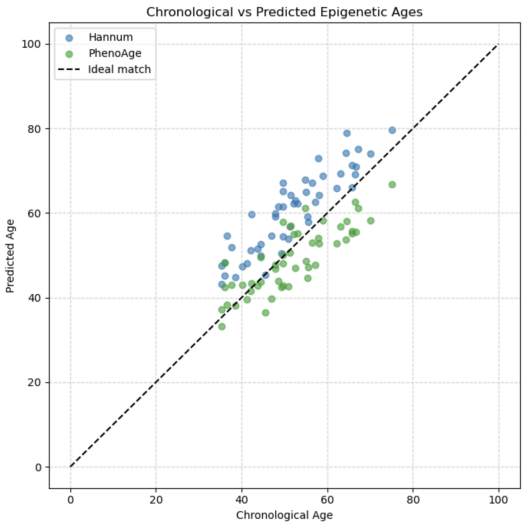
```
Inference Comparison Summary with 20 samples
-------------------------------------------------
Mean Absolute Error (Concrete ML):     0.0765 years
Max Absolute Error  (Concrete ML):     0.1466 years
Mean Absolute Error (Concrete-Numpy):  0.0551 years
Max Absolute Error  (Concrete-Numpy):  0.1489 years
-------------------------------------------------
Avg Time/sample (Plaintext):           0.000019 seconds
Avg Time/sample (Concrete ML FHE):     0.135125 seconds
Avg Time/sample (Concrete-Numpy FHE):  0.117095 seconds
```

Although both the Hannum and PhenoAge clocks are based on linear regression and use DNA methylation data from whole blood, they capture different aspects of aging. Hannum's model was trained to predict chronological age and is designed to closely match a person's actual age in years. PhenoAge, on the other hand, was trained on a composite measure of "phenotypic age" built from clinical biomarkers associated with mortality risk, making it more reflective of biological aging and health status. In practice, Hannum offers a more direct and interpretable estimate of age, while PhenoAge can give deeper insight into an individual's physiological condition. Including both models

in this project allows for a broader view of aging, combining time-based and health-based perspectives.

To evaluate how both clocks behave on the same data, I selected a sample of 50 individuals from the GSE55763 dataset, which includes methylation profiles and known chronological ages from healthy subjects. I aligned the input features to cover all CpGs used by both models (578 in total) and ran private inference using Concrete ML. This sample is available on the repository.

When applied to this healthy cohort, the models showed different behaviors. Hannum's predictions tend to systematically overestimate age but stay within a narrow range, suggesting a consistent upward bias. PhenoAge, on the other hand, yields more variable results, with both under- and overestimations, which may reflect its sensitivity to physiological differences. The general tendency of PhenoAge to predict lower ages could indicate that these individuals are biologically younger than their chronological age. This supports the idea that PhenoAge captures health-related aspects of aging, while Hannum remains a more stable but less nuanced estimator.



The inference times for the Hannum and PhenoAge models change noticeably depending on the hardware they run on. When the models were tested on a supercomputer with 64 GB of RAM and 2 CPU cores, the Hannum model was significantly faster, taking about 0.035 seconds per sample compared to 0.145 seconds for PhenoAge. But when the same models were run on Hugging Face Spaces using the "CPU Basic" tier (with just 2 virtual CPUs and 16 GB of RAM), inference times increased to around 0.19 seconds for Hannum and 0.16 seconds for PhenoAge.

This change in relative speed is mostly due to how FHE interacts with different hardware setups, as it is very sensitive to CPU performance, vectorization, and memory bandwidth. Hannum is a smaller model, it uses 71 CpG sites as input, while PhenoAge is more complex, using 513. On powerful machines, this difference in complexity makes PhenoAge slower, as expected. But on lower-resource environments like Hugging Face Spaces, general overhead (like the cost of encrypting and decrypting data) becomes a bigger part of the total time. Since this overhead affects both models, it ends up shrinking the performance gap, and Hannum loses much of its speed advantage.