Personalized Alzheimer's Disease Neural Network Model

Yilin Li
2113840@mail.nankai.edu.cn
School of Mathematical Sciences, Nankai University

2025-08-27

Abstract

In this paper, we introduce a neural network–based machine learning model for Alzheimer's Disease kinetics, focusing on illustrating the algorithm for constructing both population-level and personalized models, as well as uncertainty quantification.

Keywords Alzheimer's Disease · Dynamic System Modeling · Machine Learning · Personalized Medicine · Uncertainty Quantification

1 Introduction

Alzheimer's disease has long been one of the most threatening chronic illnesses affecting the lives of the elderly. Research on Alzheimer's disease has often focused on various biomarkers, such as the expression levels of A_β and τ proteins, and their relationship with cognitive decline. Clearly, there are inherent dynamics in this process; therefore, establishing mathematical models to accurately uncover and describe these dynamics has become a subject of both theoretical and practical importance. Zheng et al. constructed a polynomial model based on sparse identification, which consisted of quadratic polynomial equations, with the coefficients suggesting a transmission chain of $A_\beta \to \tau \to N \to C$ [1]. However, this sparse identification approach is limited to quadratic polynomials, since higher-order terms would introduce significant complexity and numerical instability. Yet, relying solely on quadratic polynomials may oversimplify the model, especially as its interpretability remains essentially at the level of logistic regression. To address this, we investigate various neural network modeling approaches, namely replacing the dynamic terms on the right-hand side of the equations with FNNs; FNNs combined with quadratic polynomials (with learnable coefficients, similarly hereafter) via residual learning; or FNNs multiplied by quadratic polynomials. By adopting ideas from PINNs, we aim to strike a balance among interpretability, model performance, and complexity. Finally, building upon the population-level model, we apply a similar approach to that of Zheng et al. and incorporate fine-tuning techniques to identify key parameters in the model. By adjusting these parameters, we achieve personalized modeling and prediction, thereby providing valuable intelligent support for the future prevention and treatment of Alzheimer's disease. In addition, we will provide uncertainty quantification of the model based on statistical methods, including estimates such as confidence intervals for the trajectories.

We study mainly 4 biomarkers related to AD, data are given by ADNI dataset: A_β, τ, N, C. The ODE model could be described as a 1-order ODE:

$$\frac{\mathrm{d}y}{\mathrm{d}s} = f(y), y = \left(A_\beta, \tau, N, C\right) \in R^4 \tag{1}$$

The polynomial (population) model proposed by Zheng et al. is as follows[1]:

$$\begin{cases} \frac{\mathrm{d}A_\beta}{\mathrm{d}s} = 0.917A_\beta - 0.873A_\beta^2 \\ \frac{\mathrm{d}\tau}{\mathrm{d}s} = 0.788\tau - 0.246\tau^2 + 0.002A_\beta + 3.066A_\beta^2 - 3.650A_\beta\tau \\ \frac{\mathrm{d}N}{\mathrm{d}s} = 1.627N - 1.253N^2 + 0.018\tau + 2.342\tau^2 - 4.015\tau N \\ \frac{\mathrm{d}C}{\mathrm{d}s} = 0.159C + 0.202C^2 + 0.010N + 0.019N^2 - 0.176NC \end{cases} \tag{2}$$

each biomarker was first normalized, i.e.

$$y = \frac{y - y_{\text{mean}}}{\sigma} \tag{3}$$

where $y_{\text{mean}} \in R^4$ denotes mean value of y and σ denotes deviation.

However, in order to place all patients on a unified temporal scale for assessing disease progression—that is, to determine the actual stage of severity—we cannot make direct comparisons on the original age scale (most individuals being between 60 and 90 years old). Instead, age must first be linearly transformed into the Disease Progression Score (DPS) scale (hereafter referred to as the DPS transform, with the resulting score denoted as DPS or simply s), upon which the model is then trained.

2 DPS Transformation

2.1 Introduction

For each patient i, we apply DPS transformation as follows:

$$s_i = a_i t_i + b_i \tag{4}$$

where a_i, b_i are patient-specific parameters and t_i denotes patient i's age series.

Unlike the approach in Zheng et al[1]., where the model parameters and the DPS transformation parameters are updated simultaneously, in this work we pre-train the DPS transformation parameters and subsequently keep them fixed during the main training process. This design choice is based on empirical observations: jointly optimizing the DPS parameters with the model often leads to numerical instability, particularly with the DPS score s frequently jumping outside the range of –60 or +600, exhibiting large and unpredictable scales, leaving a gap in interpretability. After comparing the two algorithms, we believe this difference may arise because Zheng et al.'s model is relatively simple and thus employs the more sophisticated and robust Levenberg–Marquardt algorithm, whereas our implementation relies on PyTorch with the Adam optimizer, which results in reduced numerical stability.

Here we first display the original scatter plot in which the horizontal axis is the original age axis in Figure 1.

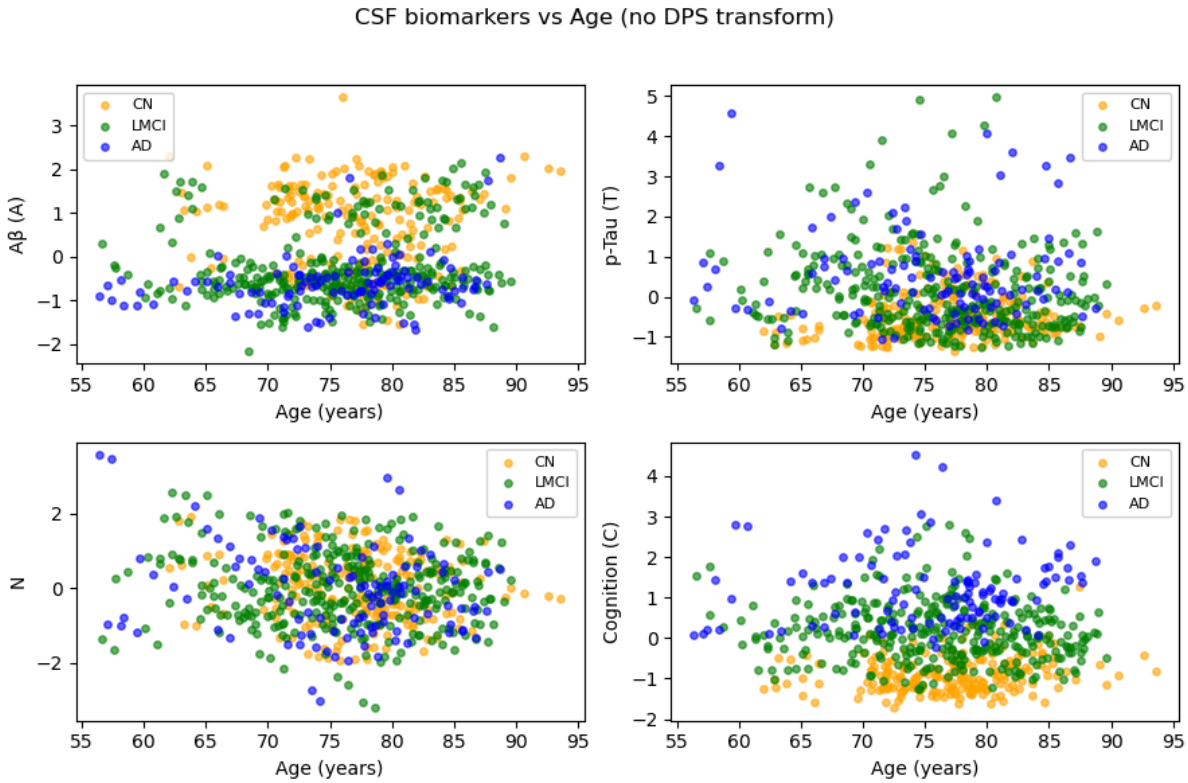

Figure 1: raw data without DPS transformation

2.2 Pretrain Stage

Our objective is to avoid reliance on existing diagnostic categories (which are limited to the three discrete labels CN, LMCI, and AD, and often cause transformed data points to cluster together). Instead, we aim to induce a continuous transformation: the lower a patient's A_β and N values and the higher their τ and C values, the more severe the disease stage, which could been seen from scatter plot. Accordingly, we define the score as follows:

$$z_i = \text{mean}\left(-A_{\beta_i} + \tau_i - N_i + C_i\right) \tag{5}$$

where mean denotes average over time, as for biomarkers here are time series.

Our next objective is to ensure that $a_i + b_i$ has the same sign as z_i and is as large as possible. Besides, we would make $0 < a_i < 4$ so as to avoid excessively unreasonable growth that would compromise interpretability, as well as avoid $a_i < 0$, which would imply an implausible (reversible) disease course.

$$\text{loss}_i = -z_i(a_i + b_i) + \text{boundary penalty i} \tag{6}$$

The boundary penalty is defined as a quadratic penalty applied to the portions of s that fall outside the interval $[-10,20]$:

$$\text{boundary penalty i} = \sum_i \left[(s_i - 20)_+^2 + (-10 - s_i)_+^2\right] \tag{7}$$

where $(x)_+ := \max(0, x)$.

$$a = 4\ \text{sigmoid}(\theta) = \frac{4}{1 + e^{-\theta}} \in (0, 4] \tag{8}$$

In the training process, we would update θ so that by Equation 8 the a_i would always be controlled in $(0, 4]$.

The final pre-training results are shown in the figure below Figure 2, where we have filtered out the top 5% and bottom 5% of outlier values of s.

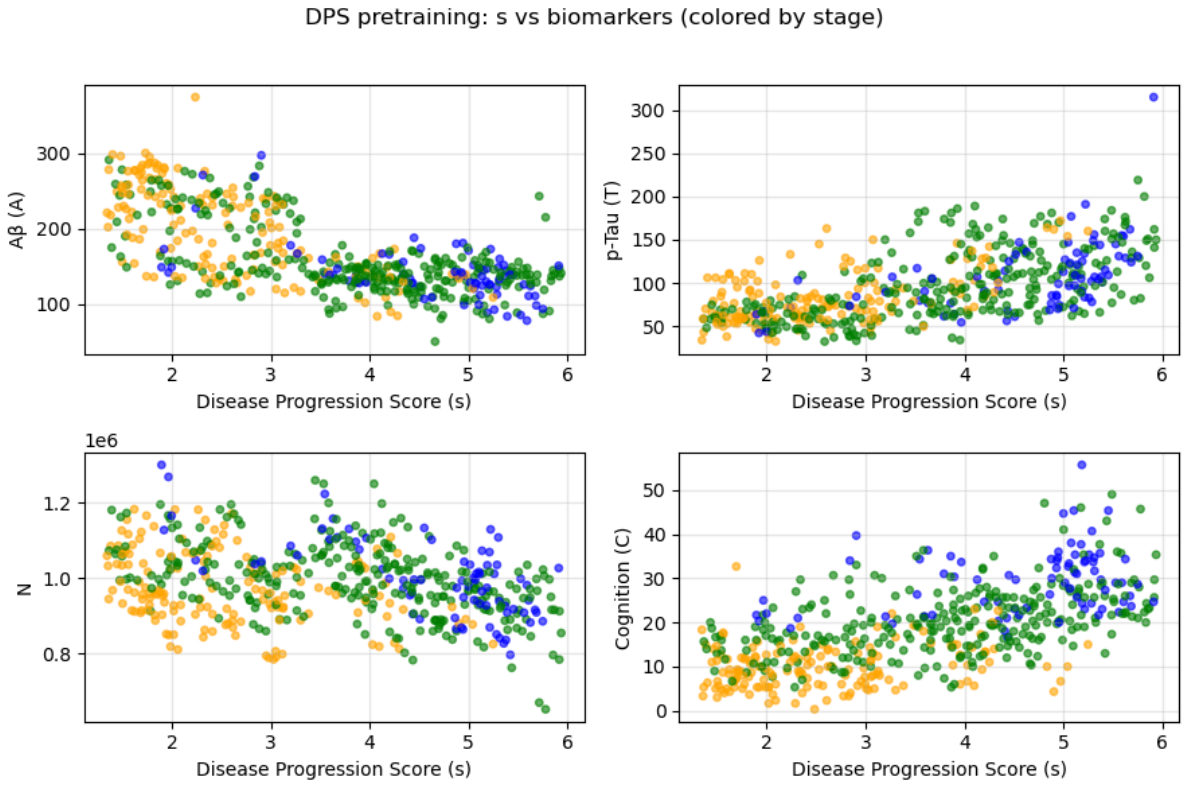

Figure 2: data after DPS transformation

3 Population Model

Now we start introduce the main model and main algorithm.

3.1 Training Algorithm

For the time being, we make no assumptions regarding the specific form of the right-hand side in the equations, and denote it simply as f. Starting from initial condition $s = -10, y_0 = (0.1, 0, 0, 0)$, i.e., assuming that the disease has not yet begun, and assigning a small value of 0.1 to A_β as the initiating factor, we apply the 4-order Runge–Kutta method to iteratively compute the values of y at each grid point of s. We then calculate the mean squared error (MSE) between these results and the population data points (after filtering out outliers). Here s grid is all s values of our filtered data. The 4-order Runge-Kutta method, RK4, could be described as follows:

$$
\begin{aligned}
k_1 &= f(y_i, s_i) \\
k_2 &= f\left(y_i + \frac{h}{2}k_1, s_i + \frac{h}{2}\right) \\
k_3 &= f\left(y_i + \frac{h}{2}k_2, s_i + \frac{h}{2}\right) \\
k_4 &= f(y_i + hk_3, s_i + h) \\
y_{i+1} &= y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}
\tag{9}
$$

The specific mathematical expression of the MSE loss is as follows:

$$
L = \frac{1}{4}\sum_{k=1}^{4} \frac{1}{\sum_{i=1}^{I} J_i} \sum_{i=1}^{I} \sum_{j_i=1}^{J_i} \left[y_{i,j,k} - \tilde{y}_{i,j,k}\right]^2 \tag{10}
$$

here I denotes the number of valid patients and J_i denotes number of time points of patient i's data. $\tilde{y}_{i,j,k}$ denotes the prediction of patient i's k-th biomarker at j-th time point.

In addition, we favor high-quality data, which is reflected in patients/samples that provide more data points (typically more than three). We implement this by simply replicating samples according to the number of their time points; for example, if a patient provides data at three time points, that sample is duplicated twice in the dataset. The parameters used in the training process are summarized in the table below.

Item	Setting
Number of Epochs	300
Batch Size	128
Learning Rate	1e-3

3.2 FNN and Polynomial Only

We first attempt two approaches separately: a simple FNN and a polynomial model. The FNN architecture is defined as follows:

```
Linear(4, 64),
ReLU(),
Linear(64, 64),
ReLU(),
Linear(64, 4),
Tanh()
```

It is worth noting that, due to the numerical stability issues of RK4 integration, even slightly larger outputs from the neural network can cause the trajectories to diverge explosively. To mitigate this, we adopt the following initialization strategy: weights are initialized from a normal distribution with mean 0 and variance 0.005, while biases are initialized to 0. This may result in trajectories remaining close to linear, but it is still preferable to numerical explosion.

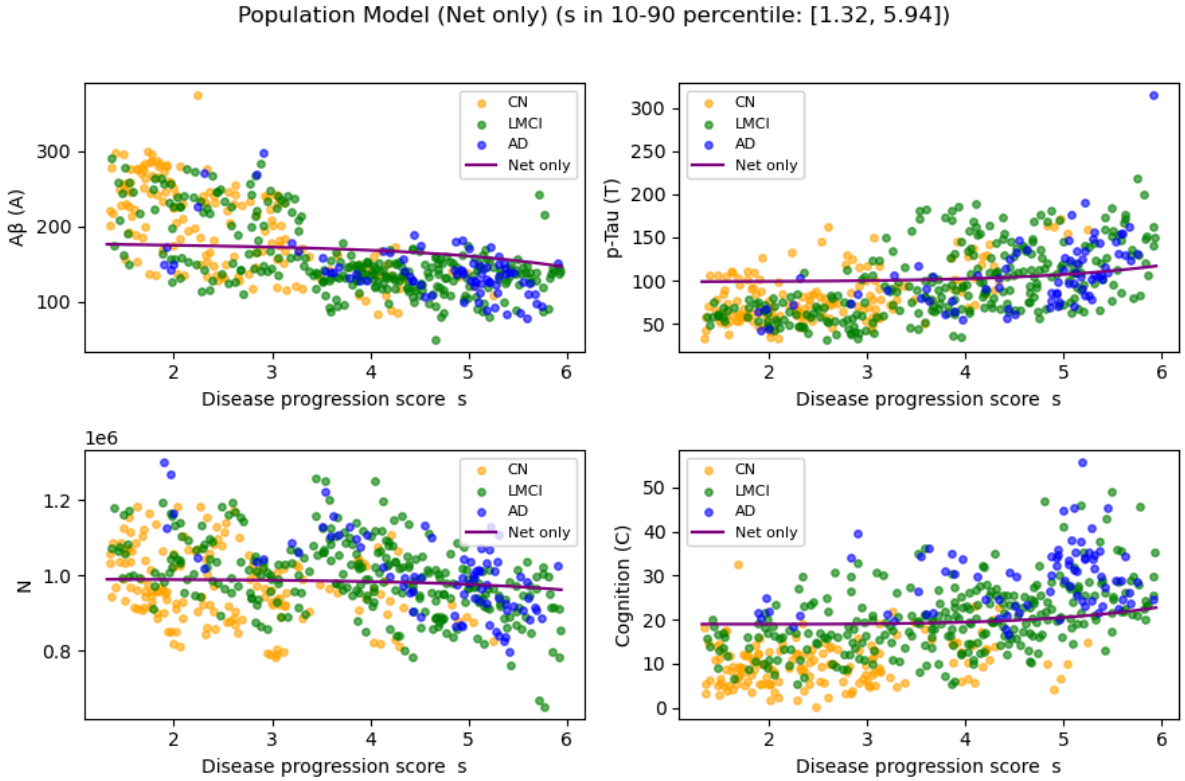

Figure 3: Single FNN Population Model Evaluation

For the polynomial model, we simplify by following the conclusions of Zheng et al., employing a quadratic polynomial model where the couplings adhere to the transmission chain $A_\beta \to \tau \to N \to C$, thereby discarding non-adjacent couplings such as $A_\beta - N$ and $\tau - C$. However, due to differences in the specific DPS transformation parameters, the values of s, and the training algorithms, we cannot directly adopt the polynomial coefficients from Zheng et al., as this would lead to numerical explosion. Instead, we initialize the polynomial coefficients with values of either 0.01 or –0.01 (with the sign determined according to prior assumptions on whether each term contributes positively or negatively to the dynamics). This avoids the risk of numerical instability caused by large initial values, while also preventing the trivial case of initializing at zero, which would otherwise result in trajectories remaining linear throughout training.

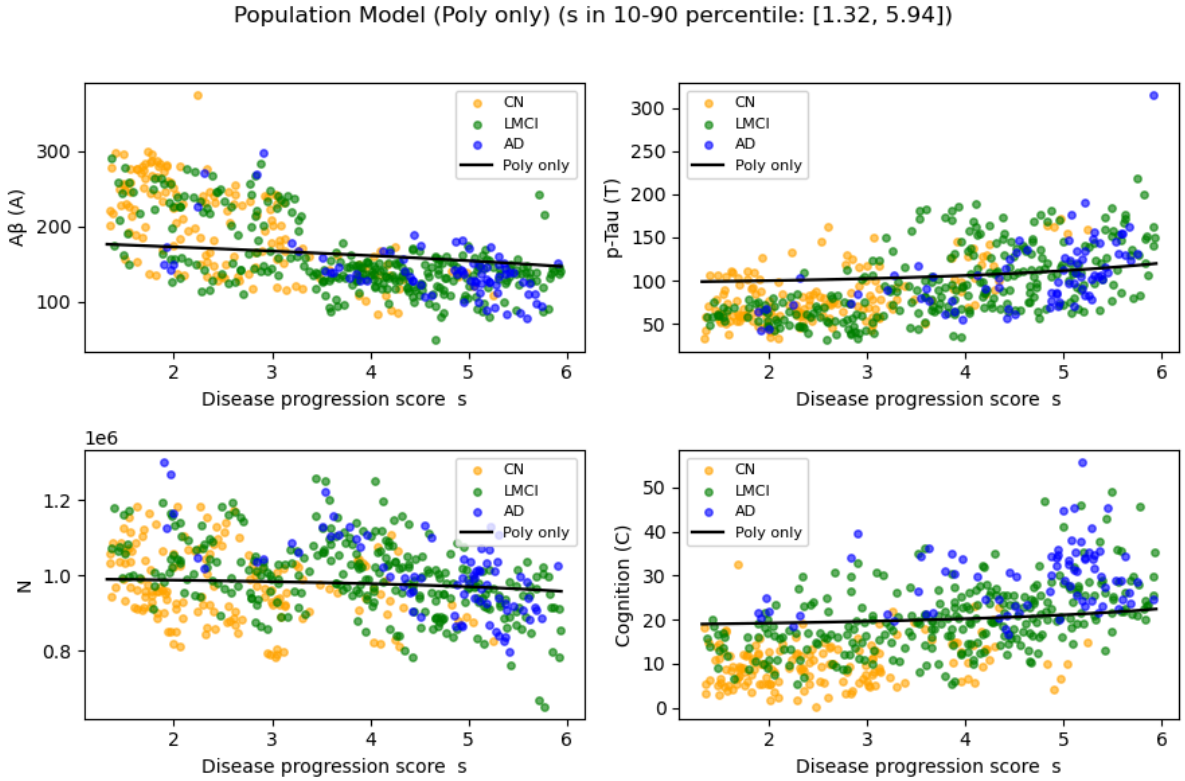

Figure 4: Single Polynomial Population Model Evaluation

3.3 Hybrid Model between FNN and Polynomial

Since standalone FNN and polynomial model are both limited in terms of accuracy and interpretability, our primary focus is on integrating them. We evaluate two fusion strategies: addition and multiplication. We will use net denote FNN and P denote polynomial model.

1. Addition: This can be interpreted as a form of residual learning, where the quadratic polynomial serves as the interpretable backbone and the FNN acts as a residual component that supplements the dynamic terms. Accordingly, in the trajectory–scatter plots, we additionally plot the trajectories predicted by each of the two models individually as references.

$$\begin{aligned} f(y) &= \mathrm{net}(y) + P(y) \in R^4 \\ \frac{\mathrm{d}y}{\mathrm{d}s} &= f(y) \end{aligned} \tag{11}$$

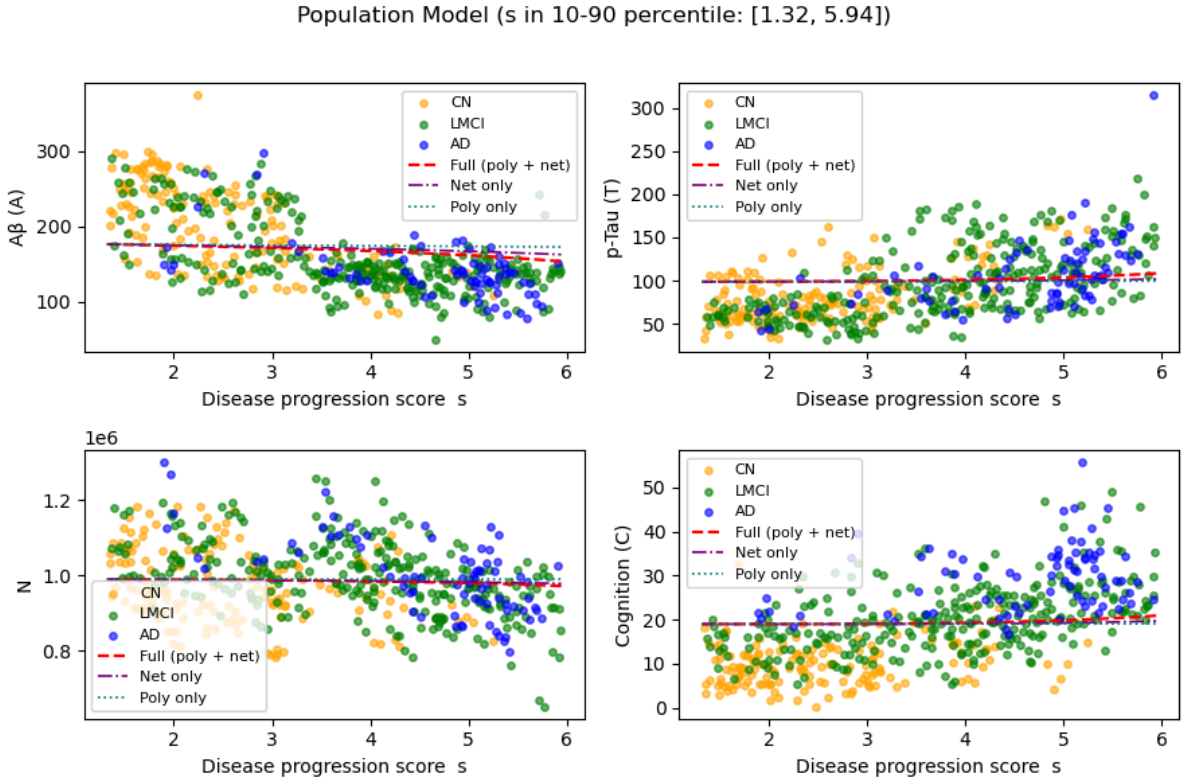

Figure 5: FNN + P Model Population Model Evaluation

1. Multiple: This represents a more complex model, in which the FNN is multiplied with the polynomial component P. Here, the FNN captures the multiplicative residual of P, which can also be understood as a complexity supplement introduced through multiplication. However, in this setting, the trajectories predicted by the FNN alone tend to diverge numerically, whereas the polynomial component P, being close to zero, counteracts this divergence and pulls the trajectories back. In this setting, in order to control numerical divergence, we modify the initialization of the neural network: the weights are drawn from a normal distribution with mean 0 and variance 0.001.

$$\begin{aligned} f(y) &= \mathrm{net}(y) \cdot P(y) \in R^4 \\ \frac{\mathrm{d}y}{\mathrm{d}s} &= f(y) \end{aligned} \tag{12}$$

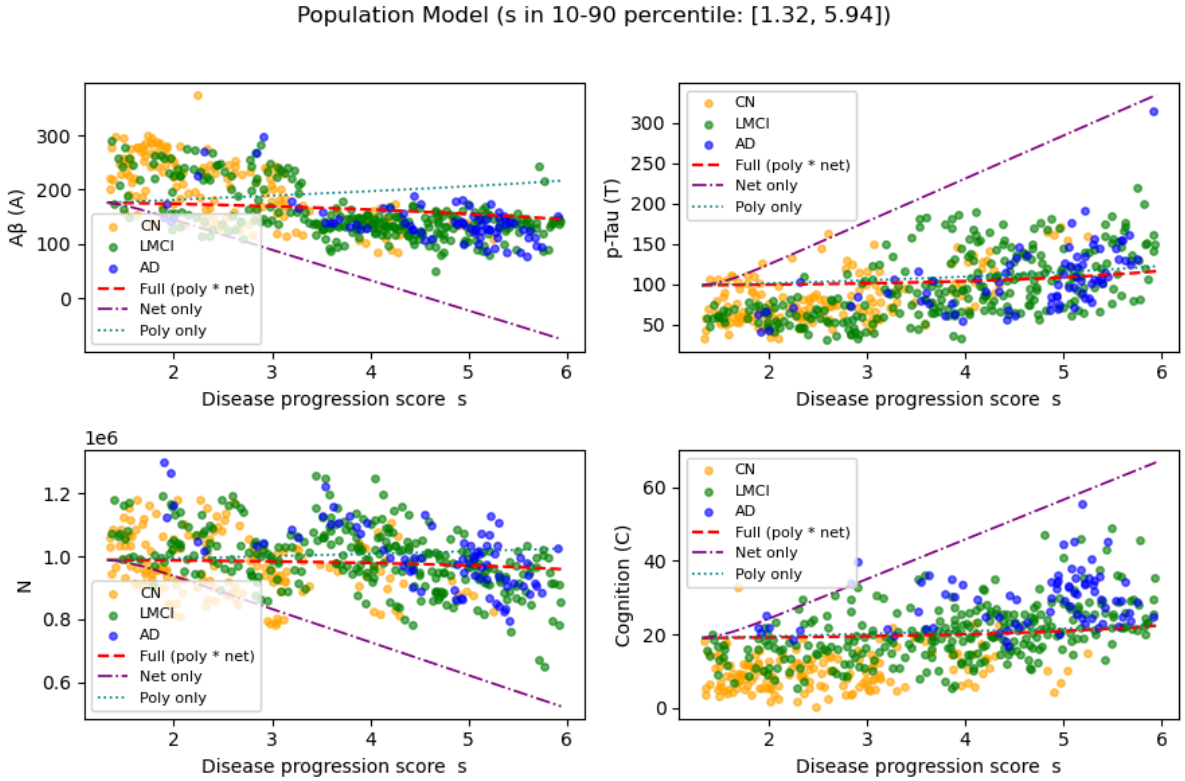

Figure 6: FNN * P Model Population Model Evaluation

Bibliography

[1] H. Zheng, J. R. Petrella, P. M. Doraiswamy, and others, "Data-driven causal model discovery and personalized prediction in Alzheimer's disease," *npj Digital Medicine*, vol. 5, no. 1, p. 137, 2022, doi: 10.1038/s41746-022-00632-7.