CAFA 6 Protein Function Prediction Report

Trung Hau Tran, Trung Hieu Pham, Nam Khanh Pham
VNU University of Engineering and Technology
Hanoi, Vietnam

***Tóm tắt nội dung*—Bài báo cáo này trình bày phương pháp của chúng tôi cho CAFA 6 Protein Function Prediction. Cuộc thi này tập trung vào việc xây dựng mô hình học máy có khả năng dự đoán được chức năng sinh học của protein chỉ dựa trên trình tự axit amin của các protein đó. Dữ liệu gồm các chuỗi protein và trình tự axit amin tương ứng, cùng với các nhãn chức năng sinh học được gán với Gene Ontology (GO).**

Chúng tôi áp dụng phương pháp học máy và học sâu, kết hợp các kỹ thuật trích xuất đặc trưng , lựa chọn mô hình embedding phù hợp và kĩ thuật ensemble mô hình để xây dựng mô hình dự đoán chức năng protein.

Điểm cuối cùng đạt được là 0.358, top 56/460 đội (tính đến ngày 9/8/2025).

I. Giới thiệu

Protein được coi là những cỗ máy phân tử đảm nhiệm hầu hết các chức năng thiết yếu của sự sống, từ xúc tác các phản ứng sinh hóa, vận chuyển vật chất, truyền tín hiệu đến việc tạo nên cấu trúc tế bào. Trong những thập kỷ gần đây, sự bùng nổ của công nghệ giải trình thứ tự gen thế hệ mới (Next-Generation Sequencing - NGS) đã tạo ra một lượng dữ liệu khổng lộ về trình tự protein. Các cơ sở dữ liệu như UniProt hiện chứa hàng trăm triệu chuỗi protein từ đa dạng các loài sinh vật.

Tuy nhiên, tốc độ xác định chức năng của các protein này bằng thực nghiệm (như tinh thể học tia X hay cộng hưởng từ hạt nhân) lại chậm hơn rất nhiều do chi phí cao và quy trình phức tạp. Sự chênh lệch ngày càng lớn giữa lượng dữ liệu trình tự đã biết và lượng dữ liệu chức năng đã được xác thực tạo nên một thách thức lớn trong sinh học tính toán, được gọi là "Khoảng trống chú giải"(The Annotation Gap). Việc phát triển các thuật toán máy tính để dự đoán tự động chức năng protein (Automated Function Prediction - AFP) trở thành một nhu cầu cấp thiết để lấp đầy khoảng trống này.

Để thúc đẩy sự phát triển của các thuật toán AFP, cuộc thi CAFA (Critical Assessment of Functional Annotation) đã được tổ chức như một thử thách toàn cầu, nơi các nhà nghiên cứu tranh tài để dự đoán chức năng của một tập hợp các protein chưa được chú giải. CAFA 6 là phiên bản mới nhất của chuỗi thử thách này, đặt ra yêu cầu cao hơn về độ chính xác và khả năng tổng quát hóa của mô hình trên các loài sinh vật khác nhau.

Trong báo cáo này, chúng tôi trình bày các phương pháp đã áp dụng khi tham gia CAFA 6, bao gồm mô hình hóa bài toán, xây dựng khung quy trình tiền xử lý dữ liệu, tiếp cận bài toán theo ba hướng: Không học máy (non-ML) ,học máy (ML) và học sâu (DL).

II. Background

A. CAFA 6 Protein Function Prediction

B. Tập dữ liệu

Dữ liệu sử dụng trong nghiên cứu này được cung cấp bởi CAFA 6 gồm các tệp tin chính đóng vai trò đầu vào và nhãn mục tiêu cho quá trình huấn luyện:

- `train_sequences.fasta` (Dữ liệu thô):
  - Trình tự axit amin (primary sequences) của khoảng 140.000 protein.
  - Định dạng: FASTA. Mỗi mục gồm mã định danh (EntryID) và chuỗi ký tự đại diện cho các axit amin. (Ví dụ: M, K, T, L,...). Đây là dữ liệu đầu chính cho mô hình.
- `train_terms.tsv` (Dữ liệu nhãn):
  - Chứa thông tin gán nhãn chức năng cho các protein trong tập huấn luyện.
  - Cấu trúc: EntryID <-> GO Terms (Mã chức năng) <-> Aspect (Nhóm chức năng: BPO, MFO, CCO).
  - Đặc điểm: Một protein có thể tương ứng với nhiều nhãn GO khác nhau (Multi-lable).
- `go-basic.obo` (Cấu trúc Ontology):
  - File định nghĩa cấu trúc đồ thị của Gene Ontology, mô tả mối quan hệ cha-con giữa các thuật ngữ chức năng.
- `train_taxonomy.tsv` (Thông tin loài):
  - Cung cấp mã định danh loài (Taxon ID) cho từng protein. Dữ liệu này giúp mô hình phân biệt đặc điểm sinh học giữa các loài khác nhau (ví dụ: vi khuẩn vs. động vật có vú).

III. Phương pháp

A. Tiền xử lý dữ liệu

1) Tiền xử lý dữ liệu chuỗi (Sequence): Dữ liệu thô ban đầu là các chuỗi trình tự protein (dạng văn bản) trong tệp FASTA. Để đưa vào các mô hình học máy, chúng tôi đã thử nghiệm và triển khai hai phương pháp tiếp cận tiền xử lý dữ liệu riêng biệt nhằm khai thác tối đa thông tin từ trình tự protein:

- **Phương pháp 1: Trích xuất đặc trưng thủ công (Manual Feature Engineering)** — Dựa trên kiến thức sinh học thống kê.
- **Phương pháp 2: Học biểu diễn ngữ nghĩa (Semantic Representation Learning)** — Dựa trên mô hình ngôn ngữ lớn (ESM-2).

a) Phương pháp 1: Trích xuất đặc trưng thủ công: Phương pháp này coi chuỗi protein là một tập hợp các thành phần hóa học và sử dụng công thức thống kê để biến đổi chuỗi thành vector số học. Phương pháp tập trung vào đặc điểm thành phần (composition) thay vì thứ tự sắp xếp.

Xử lý dữ liệu thô:

- Đọc dữ liệu: sử dụng thư viện `Biopython` để phân tích cú pháp tệp FASTA.
- Làm sạch ID: Tách phần tiêu đề (Header) của FASTA để lấy mã định danh chuẩn (EntryID).

Kỹ thuật tạo đặc trưng (Feature Engineering): Chúng tôi xây dựng vector đặc trưng X_{manual} dựa trên các thuộc tính lý hóa của protein:

- `Thành phần Axit Amin (Amino Acid Composition -- AAC)`: tính toán tần suất xuất hiện của 20 loại axit amin chuẩn (A, R, N, D, C, Q, E, G, H, I, L, K, M, F, P, S, T, W, Y, V) trong mỗi chuỗi. Công thức:

$$AAC_i = \frac{\text{Count}(AA_i)}{\text{Length(Sequence)}} \tag{1}$$

- `Độ dài chuỗi (Sequence Length)`: Thêm đặc trưng về tổng số lượng axit amin. Độ dài chuỗi thường tương quan với độ phức tạp của chức năng protein.

Kết quả là mỗi protein được biểu diễn bằng một vector chiều thấp (Low-dimensional vector) với kích thước $D = 21$.

b) Phương pháp 2: Học biểu diễn ngữ nghĩa (Semantic Representation Learning): Phương pháp này coi chuỗi protein là một "ngôn ngữ" và sử dụng mô hình học sâu đã được huấn luyện trước (Pre-trained Model) để trích xuất các đặc trưng ngữ nghĩa tiềm ẩn. Phương pháp này nắm bắt được ngữ cảnh, thứ tự và cấu trúc 3D của protein.

Mô hình nền tảng: Chúng tôi sử dụng mô hình `ESM-2` (Evolutionary Scale Modeling) phiên bản 650 triệu tham số `(esm2_t33_650M_UR50D)`. Đây là mô hình Transformer tiên tiến nhất hiện nay cho dữ liệu protein.

Quy trình embedding:

- **Bước 1: Mã hóa (tokenization)** — Mỗi axit amin được chuyển đổi thành token số nguyên. Các token đặc biệt `<cls>` và `<eos>` được thêm vào để đánh dấu bắt đầu và kết thúc chuỗi.
- **Bước 2: Suy luận (inference)** — Chuỗi token được đưa qua 33 lớp Transformer của `ESM-2`. Tại đây, mô hình sử dụng cơ chế Self-Attention để học mối quan hệ phức tạp giữa các axit amin, bất kể khoảng cách của chúng trong chuỗi.
- **Bước 3: Gộp (pooling)** — Chúng tôi lấy vector trạng thái ẩn (hidden states) tại lớp cuối cùng và áp dụng kỹ thuật Mean Pooling (lấy trung bình) dọc theo chiều dài chuỗi để thu được một vector đại diện duy nhất cho toàn bộ protein.

Kết quả là mỗi protein được biểu diễn bằng một vector với kích thước $D = 1280$ (High-dimensional vector). Vector này chứa đựng thông tin phong phú về cấu trúc và chức năng của protein.

2) Tiền xử lý dữ liệu nhãn (Label): Đối với dữ liệu label, chúng tôi sử dụng phương pháp mã hóa và lọc dữ liệu để giải quyết thách thức của bài toán phân loại đa nhãn cực đoan (Extreme Multi-label Classification). Do không gian nhãn GO rất lớn (hơn 40.000 thuật ngữ) và sự phân bố dữ liệu mất cân bằng nghiêm trọng, quy trình xử lý được thực hiện qua hai bước chính:

a) Bước 1: Lọc nhãn phổ biến (Top-K Filtering): Phân tích thống kê cho thấy tần suất xuất hiện của các nhãn GO tuân theo quy luật "phân phối đuôi dài" (long-tail distribution). Phần lớn các nhãn chỉ xuất hiện ở một số lượng rất nhỏ các protein (nhãn hiếm — rare terms), trong khi một số ít nhãn lại xuất hiện rất thường xuyên. Việc cố gắng dự đoán các nhãn hiếm thường gây nhiễu cho mô hình, tốn kém tài nguyên tính toán và dẫn đến hiện tượng học vẹt (overfitting).

Do đó, chúng tôi áp dụng chiến lược chỉ giữ lại **Top K nhãn phổ biến nhất** cho mỗi nhóm chức năng (BPO, MFO, CCO) và loại bỏ các nhãn phần đuôi. Ví dụ, chúng tôi chọn $K \approx 1500$ cho nhóm BPO và $K \approx 1000$ cho các nhóm còn lại. Chiến lược này giúp giảm chiều dữ liệu đầu ra từ hàng chục nghìn xuống mức quản lý được, đồng thời tập trung mô hình vào các chức năng sinh học cốt lõi có độ tin cậy thống kê cao.

b) Bước 2: Mã hóa One-hot (One-hot Encoding): Sau khi lọc danh sách nhãn mục tiêu, chúng tôi chuyển đổi dữ liệu sang dạng số học bằng kỹ thuật *Multi-label Binarization*. Với mỗi protein i và tập hợp nhãn L_i của nó, ta tạo ra một vector nhãn $y_i \in \{0,1\}^K$. Giá trị của phần tử thứ j trong vector được xác định như sau:

$$y_{i,j} = \begin{cases} 1 & \text{nếu protein } i \text{ mang chức năng } j \\ 0 & \text{ngược lại} \end{cases} \tag{2}$$

Kết quả là chúng tôi thu được các ma trận nhãn thưa (Sparse Matrices). Ma trận này đóng vai trò là nhãn thực tế (Ground Truth) để tính toán hàm mất mát (Loss Function) trong quá trình huấn luyện mô hình.

B. Tiếp cận theo hướng không dùng học máy

1) Naive Prior Baseline: Frequency-based GO Term Prediction: Để thiết lập một mốc tham chiếu (baseline) đơn giản, chúng tôi xây dựng một phương pháp dự đoán dựa hoàn toàn trên thống kê tần suất nhãn trong tập huấn luyện, không sử dụng bất kỳ đặc trưng nào từ chuỗi protein hay embedding. Ý tưởng cốt lõi là khai thác *prior distribution* của các nhãn Gene Ontology (GO): các GO term xuất hiện càng thường xuyên trong dữ liệu huấn luyện được giả định có xác suất cao hơn để xuất hiện ở các protein chưa biết nhãn.

a) Ước lượng phân phối nhãn theo từng phân hệ (aspect).: Gọi $a \in \{P, C, F\}$ lần lượt tương ứng với *Biological Process*, *Cellular Component* và *Molecular Function*. Từ tập huấn luyện, ta đếm số lần xuất hiện của mỗi GO term t trong aspect a:

$$c_a(t) = \#\{(\cdot, t) \mid \text{aspect} = a\}, \tag{3}$$

và tổng số annotation trong aspect:

$$N_a = \sum_t c_a(t). \tag{4}$$

Xác suất tiên nghiệm của term t trong aspect a được ước lượng bằng tần suất chuẩn hoá:

$$p_a(t) = \frac{c_a(t)}{N_a}. \tag{5}$$

b) Chọn top-k GO term phổ biến nhất.: Với mỗi aspect a, ta chọn tập $\mathcal{T}_a^{(k)}$ gồm k term có $c_a(t)$ lớn nhất:

$$\mathcal{T}_a^{(k)} = \mathrm{TopK}_t\big(c_a(t)\big). \tag{6}$$

c) Inference: Với mỗi protein trong tập test x, baseline này không phân biệt giữa các protein mà xuất ra cùng một tập dự đoán cho mọi x. Cụ thể, với mỗi aspect a, ta dự đoán tất cả các term trong $\mathcal{T}_a^{(k)}$ với điểm tin cậy bằng xác suất tiên nghiệm $p_a(t)$:

$$\hat{s}(x,t) = \begin{cases} p_a(t), & \text{nếu } t \in \mathcal{T}_a^{(k)}, \\ \text{không xuất ra}, & \text{ngược lại.} \end{cases} \tag{7}$$

d) Vai trò và hạn chế.: Ưu điểm của baseline này là đơn giản, dễ tái lập và gần như không tốn chi phí tính toán. Tuy nhiên, do dự đoán không phụ thuộc vào đặc trưng protein, phương pháp không thể cá nhân hoá dự đoán theo chuỗi và chỉ phù hợp như một mốc tham chiếu để so sánh với các phương pháp học có điều kiện theo protein.

2) Phương pháp tiếp cận dựa trên sự tương đồng (Homology-based Approach):

Chúng tôi sử dụng phương pháp tin sinh học kinh điển: Gióng hàng trình tự (Sequence Alignment). Nguyên lý cốt lõi của phương pháp này dựa trên giả thuyết sinh học *"Guilt by Association"* (Tương đồng cấu trúc dẫn đến tương đồng chức năng).

Cụ thể, nếu protein A (trong tập Test) có trình tự axit amin tương đồng cao với protein B (đã biết chức năng trong tập Train), khả năng cao A sẽ thừa hưởng các chú giải chức năng của B. Do đó, bài toán dự đoán chức năng được chuyển về bài toán tìm kiếm láng giềng gần nhất trong không gian trình tự sinh học.

a) Thuật toán BLAST (Basic Local Alignment Search Tool): BLAST là tiêu chuẩn vàng trong tìm kiếm tương đồng trình tự cục bộ. Thay vì sử dụng quy hoạch động toàn cục (như Needleman-Wunsch) tốn kém chi phí tính toán $O(m \times n)$, BLAST sử dụng phương pháp heuristic dựa trên thống kê để tìm các vùng tương đồng cục bộ (Local Alignment) có ý nghĩa nhất.

Quy trình hoạt động và cơ sở toán học của BLAST bao gồm 3 thành phần chính:

`1. Ma trận chấm điểm (Scoring System):` Để định lượng mức độ giống nhau giữa hai axit amin i và j, BLAST sử dụng ma trận điểm thay thế (thường là BLOSUM62). Điểm số S_{ij} được tính dựa trên tỷ lệ *log-odds*:

$$S_{ij} = \frac{1}{\lambda} \log\left(\frac{p_{ij}}{q_i q_j}\right) \tag{8}$$

Trong đó:

- p_{ij}: Xác suất quan sát thấy axit amin i và j thay thế cho nhau trong các chuỗi tương đồng thực sự (do bảo tồn tiến hóa).
- q_i, q_j: Tần suất xuất hiện ngẫu nhiên (nền) của axit amin i và j.
- λ: Hệ số tỷ lệ.

Nếu $S_{ij} > 0$, sự thay thế được coi là tương đồng; ngược lại là không tương đồng.

`2. Cơ chế Seed-and-Extend:` Thuật toán hoạt động qua hai bước:

- **Gieo hạt (Seeding):** Chia chuỗi truy vấn thành các đoạn ngắn (k-mers, thường $k = 3$ cho protein). Tìm các vị trí trong cơ sở dữ liệu khớp với các k-mers này (hoặc có điểm số thay thế vượt ngưỡng T).
- **Mở rộng (Extension):** Từ các hạt giống, thuật toán mở rộng sang hai phía để hình thành Cặp đoạn điểm cao (HSP — High-scoring Segment Pair). Điểm số tích lũy S của đoạn liên kết được tính bằng tổng điểm các cặp axit amin trừ đi điểm phạt khoảng trống (gap penalty):

$$S = \sum_k S(x_k, y_k) - (G_{open} + L_{gap} \cdot G_{ext}) \tag{9}$$

  Quá trình mở rộng dừng lại khi điểm số suy giảm quá mức X so với giá trị đỉnh (drop-off heuristic).

`3. Đánh giá thống kê (E-value):` Để loại bỏ các kết quả ngẫu nhiên, BLAST sử dụng chỉ số E-value (Expectation value). E-value biểu thị số lượng các so khớp có điểm số $\geq S$ mà ta kỳ vọng tìm thấy *ngẫu nhiên* trong một cơ sở dữ liệu kích thước N:

$$E = K \cdot m \cdot n \cdot e^{-\lambda S} \tag{10}$$

Trong đó m, n là độ dài chuỗi truy vấn và cơ sở dữ liệu; K, λ là hằng số thống kê Gumbel. Một kết quả được coi là có ý nghĩa sinh học nếu E rất nhỏ (ví dụ $E < 10^{-5}$).

Hạn chế đối với CAFA 6 đó là mặc dù có độ chính xác cao, độ phức tạp tính toán của BLAST trở thành rào cản lớn đối với dữ liệu Big Data. Với tập dữ liệu hàng trăm nghìn chuỗi của CAFA 6, việc thực hiện so khớp tất cả với tất cả (All-vs-All) bằng BLAST có thể mất hàng tuần xử lý trên CPU thông thường, không khả thi trong giới hạn thời gian của cuộc thi.

b) Sử dụng DIAMOND làm giải pháp thay thế: Để khắc phục hạn chế về hiệu năng tính toán của BLAST trên dữ liệu lớn, chúng tôi sử dụng **DIAMOND** (Double Index Alignment of Next-generation Sequencing Data). Đây là thuật toán được tối ưu hóa đặc biệt cho dữ liệu giải trình tự thế hệ mới, cho phép tốc độ xử lý nhanh hơn từ **500 đến 20.000 lần** so với BLAST trong khi vẫn duy trì độ nhạy tương đương ở các thiết lập tiêu chuẩn.

Cơ sở toán học và thuật toán của DIAMOND dựa trên hai cải tiến cốt lõi so với mô hình *Seed-and-Extend* truyền thống:

`1. Bảng chữ cái rút gọn (Reduced Alphabet):` Trong khi BLAST so sánh chính xác trên không gian 20 axit amin chuẩn Σ_{std}, DIAMOND thực hiện tìm kiếm hạt giống trên một không gian rút gọn Σ_{red} (thường gồm 11 nhóm). Ánh xạ Φ được định nghĩa dựa trên tính chất hóa lý của axit amin:

$$\Phi : \Sigma_{std} \to \Sigma_{red} \tag{11}$$

Ví dụ: Các axit amin kỵ nước (Hydrophobic) hoặc tích điện dương (Positively Charged) được gom nhóm:

$$\Phi(\{L, V, I, M\}) \to \text{Hydrophobic}, \quad \Phi(\{K, R\}) \to \text{Positive}$$

Điều kiện khớp hạt giống (seed match) giữa đoạn chuỗi S_1 và S_2 tại vị trí k trở thành:

$$\text{Match}(S_1[k], S_2[k]) \iff \Phi(S_1[k]) = \Phi(S_2[k]) \tag{12}$$

Việc này làm tăng xác suất tìm thấy các hạt giống (seeds) ngay cả khi có đột biến thay thế axit amin, cho phép sử dụng các hạt giống dài hơn và có khoảng cách (spaced seeds) để tăng tốc độ lọc mà không làm giảm độ nhạy.

`2. Đánh chỉ mục kép (Double Indexing):` Khác với BLAST chỉ đánh chỉ mục cơ sở dữ liệu (Database), DIAMOND đánh chỉ mục cho cả Chuỗi truy vấn (Query) và Cơ sở dữ liệu cùng lúc. Thuật toán sắp xếp các hạt giống từ cả hai nguồn theo thứ tự từ điển. Quá trình tìm kiếm trở thành việc duyệt tuyến tính qua hai danh sách đã sắp xếp, giúp tối ưu hóa việc truy cập bộ nhớ cache (cache locality) và giảm thiểu chi phí truy cập ngẫu nhiên.

`3. Chuyển giao điểm số (Score Propagation):` Sau khi tìm được các protein tương đồng từ tập huấn luyện, điểm số dự đoán cho protein mục tiêu được tính:

$$Score(P_{test}, F) = \max_{P_{train} \in \text{Hits}} \left(\frac{\text{pident}(P_{test}, P_{train})}{100} \times y_{train,F} \right) \tag{13}$$

Trong đó pident là tỷ lệ phần trăm axit amin giống hệt, $y_{train,F} \in \{0, 1\}$ là nhãn của protein huấn luyện. Công thức này đảm bảo rằng các protein càng giống nhau về cấu trúc thì độ tin cậy của việc chuyển giao chức năng càng cao.

C. Tiếp cận theo hướng sử dụng học máy

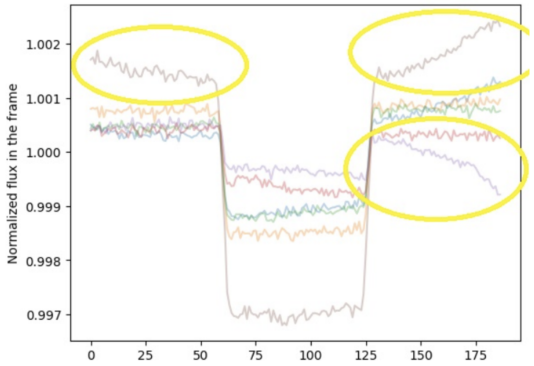
Chúng tôi đã thử một số phương pháp học máy khác nhau, gồm: K-Nearest Neighbors (KNN), Gaussian Process Regressor, phương pháp denoise bằng Auto Encoder. Tuy nhiên, kết quả thu được không khả quan, do dữ liệu có quá nhiều nhiễu và không đủ thông tin để mô hình hóa chính xác.

IV. Thực nghiệm

A. Tối ưu các hệ số mô hình non-ML

Ý tưởng của phương pháp non-ML đơn giản nhưng để thực nghiệm hiệu quả thì cần phải chọn các tham số phù hợp dựa vào các quan sát trên dữ liệu.

Như đã đề cập ở trên, chúng tôi nhân đoạn tín hiệu transit với $1 + s$ và kỳ vọng đoạn tín hiệu thu được sẽ mượt như khi không có hành tinh nào đi qua sao chủ. Tín hiệu này sẽ được xấp xỉ bằng một đa thức. Việc chọn bậc của đa thức này ảnh hưởng lớn đến độ chính xác của mô hình. Nhìn vào dữ liệu như hình 1, ta thấy rằng sau khi

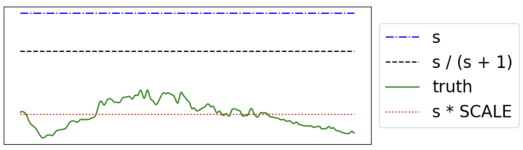


Hình 1. Hình dạng phổ biến của flux theo thời gian trong tập dữ liệu

nhân phần lõm ở giữa với $1 + s$ thì tín hiệu không phải lúc nào cũng là 1 đường tuyến tính. Hình dạng phổ biến là bậc 1, 2, 3. Chúng tôi thực nghiệm với các giá trị bậc đa thức khác nhau cho kết quả như bảng *A*..

Bậc đa thức	1	2	3	4	5	10
Điểm	0.314	0.315	0.322	0.310	0.304	0.115

Sau khi fit tín hiệu với một đa thức bậc 3, ta tìm s tối ưu (hiệu trị tuyệt đối nhỏ nhất), chúng tôi dự đoán transit depth dựa vào s.



Hình 2. Mối quan hệ giữa s, mục tiêu cần dự đoán

Nhận thấy rằng, nếu lấy giá trị dự đoán theo $\frac{s}{1+s}$ cho ra kết quả thấp hơn so với lấy $s * SCALE$. Hơn nữa, khi ta tính giá trị $SCALE = truth.mean/s$ với mỗi điểm dữ liệu, thì giá trị này sẽ gần bằng nhau cho tất cả các hành tinh. Chạy thực nghiệm cho thấy, sử dụng $SCALE = 0.9396$ cho điểm cao nhất.

Dựa vào công thức tính điểm, để tối ưu điểm không chỉ cần dự đoán chính xác mà còn cần dự đoán độ không chắc

chắn phù hợp. Giả định rằng, giá trị dự đoán của mô hình khớp với trung bình của quang phổ cần dự đoán, như vậy

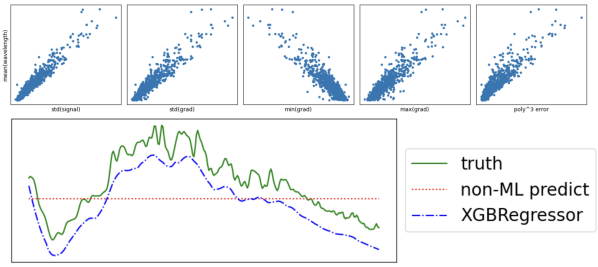
$$\sigma^* = argmax_\sigma \sum_{i=1}^{283} \left(log(\sigma^2) + \frac{(truth[i] - \mu_{truth})^2}{\sigma^2} \right) \tag{14}$$

Dễ thấy, hàm mục tiêu là tổng 2 hàm lồi,

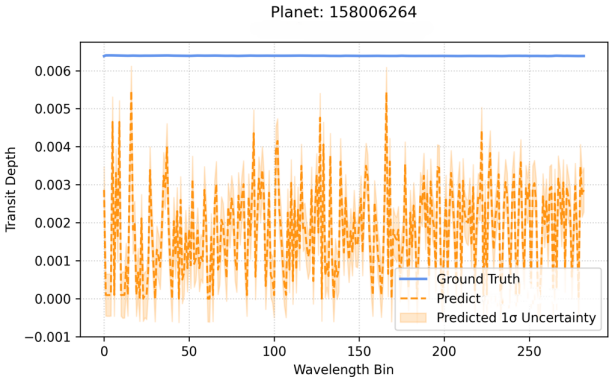
$$(\sigma^*)^2 = \sum_{i=1}^{283} (truth[i] - \mu_{truth})^2 = (\sigma_{truth})^2 \tag{15}$$

Nhận thấy rằng, giá trị σ tối ưu cho từng hành tinh là khác nhau và có liên quan đến tín hiệu theo trục thời gian. Chúng tôi sử dụng mô hình phụ LinearRegression với đầu vào là tín hiệu, đầu ra là σ_{truth}, và thực hiện huấn luyện mô hình này trên tập dữ liệu huấn luyện. Kết quả tăng 0.004 so với việc sử dụng giá trị σ cố định cho tất cả các hành tinh.

B. Kết quả



Hình 3. Kết quả của mô hình Regressor so với non ML

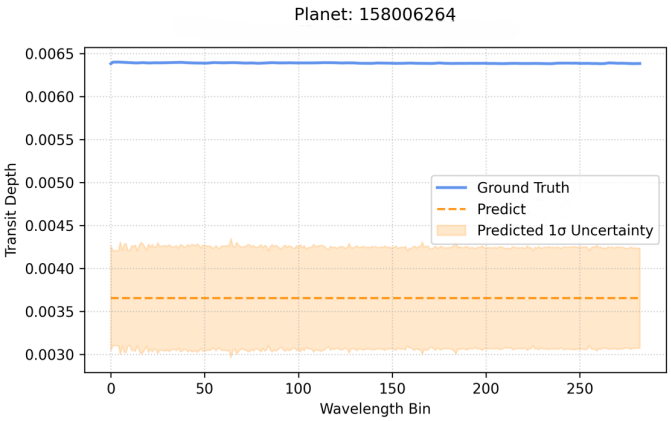


Hình 4. Các bước sóng chưa qua làm mịn

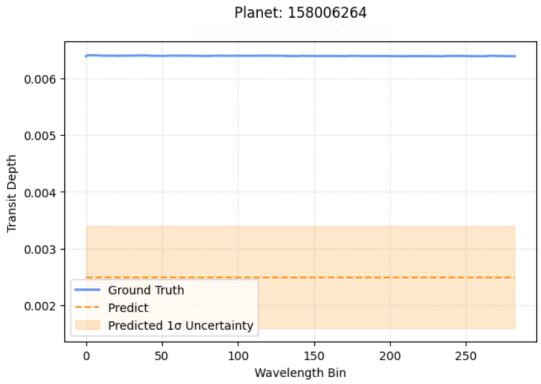
Phương pháp non-ML dự đoán quang phổ và Linear-Regression dự đoán độ không chắc chắn cho điểm trên tập valid là 0.75, điểm trên tập test là 0.326. Đạt hạng 56/460 đội (tính đến ngày 9/8/2025).

Với các phương pháp ML cho dự đoán quang phổ,

Chúng tôi thử với mô hình Regressor, lấy thuộc tính là giá trị flux và một số thuộc tính của gradient của flux có liên quan mạnh đến độ sâu transit như hình 3



Hình 5. Các bước sóng đã qua làm mịn



Hình 6. Các bước sóng dự đoán của mô hình non-ML

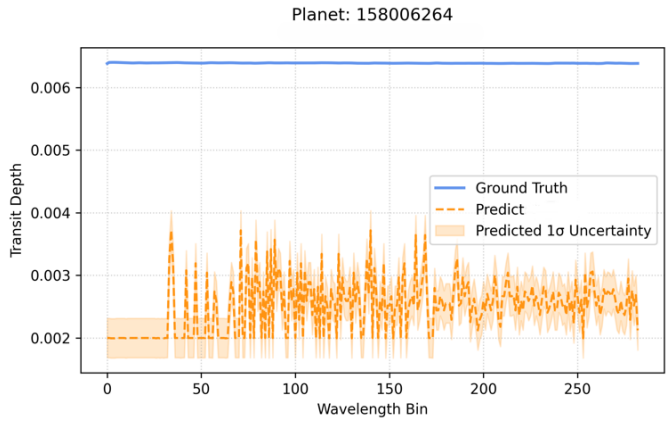
với điểm trên tập valid là 0.734, điểm trên tập test là 0.289, nguyên nhân có thể là do sự khác biệt trong phân phối dữ liệu giữa các tập và chọn thuộc tính chưa phù hợp.

Sự khác biệt về phân phối dữ liệu càng thể hiện rõ hơn khi chúng tôi thử với KNN, với khoảng cách trung bình các điểm dữ liệu trong tập train (với k = 2) là 50. Chúng tôi kết hợp KNN và phương pháp non-ML, với điểm dữ liệu trong tập test có khoảng cách với các điểm trong trong KNN dưới 40, chúng tôi nội suy từ tập train, những điểm lớn hơn thì chúng tôi sử dụng phương pháp non-ML. Kết quả test là 0.322, có hơn phương pháp non-ML nhưng độ chênh lệch quá nhỏ.

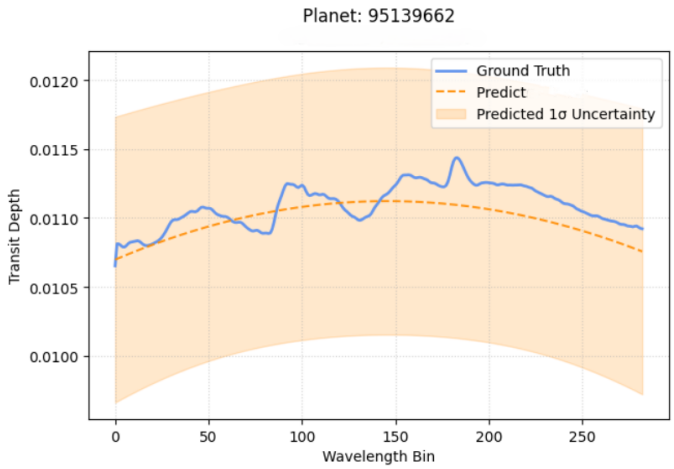
Chúng tôi cũng thử làm mịn kết quả dự đoán của 283 bước sóng bằng Gaussian Process (GP) và AutoEncoder (AE).

Như trong hình 4, các bước sóng chưa làm mịn có nhiều nhiễu và không đồng nhất. Để cải thiện, chúng tôi đã áp dụng Gaussian Process (GP) và AutoEncoder (AE) để làm mịn, khử nhiễu các bước sóng này. Kết quả là các bước sóng trở nên đồng nhất hơn, như trong hình 5.

Đồng thời, GP cũng sinh ra các ước lượng không chắc chắn cho các bước sóng, cho phép chúng tôi đánh giá độ tin cậy σ của các dự đoán. So với bước sóng được dự đoán bằng phương pháp non-ML , như trong hình 6, các

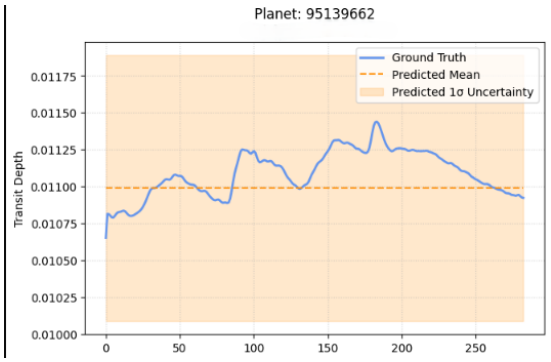


Hình 7. Các bước sóng đã qua làm mượt bằng PCA



Hình 8. Dự đoán của mô hình ML bám sát được hình dạng của ground truth

bước sóng này đã sát với ground truth hơn, 0.00356 so với 0.0024.



Hình 9. Dự đoán của mô hình non-ML không bám sát được hình dạng của ground truth

Một ý tưởng khác để làm mịn bước sóng là sử dụng PCA tìm ra những component chính của các bước sóng này. Kết quả làm mượt bằng PCA cho thấy những tiềm năng của phương pháp này trong việc cải thiện độ chính xác của dự đoán. Bằng việc sử dụng 5 component chính, chúng tôi đã đạt được những cải thiện đáng kể trong việc giảm thiểu nhiễu và tăng cường độ chính xác cho các bước sóng dự đoán.

Dù việc sử dụng PCA đã mang lại những cải thiện đáng kể trên valid, nhưng vẫn còn nhiều thách thức trong việc xử lý dữ liệu nhiễu và không đồng nhất. PCA với 5 component đem lại kết quả thấp trên test set so với valid set có thể do các tập trong test set phân tán hơn, dẫn đến việc mô hình không thể tổng quát tốt hơn cho các bước sóng này.

So với sử dụng mean của toàn bộ quang phổ như non-ML, việc dự đoán từng bước sóng cải thiện việc bắt được cấu trúc của các bước sóng quang phổ phức tạp , như hình 8. So với việc dự đoán từng bước sóng, phương pháp non-ML chỉ cho ra được một giá trị trung bình cho toàn bộ quang phổ, như hình 9. Tuy nhiên, việc dự đoán từng bước sóng cũng làm tăng độ nhiễu và độ không đồng nhất của các bước sóng này, dẫn đến việc khó khăn trong việc tổng quát khiến những phương pháp chúng tôi thử nghiệm không đạt được kết quả tốt hơn so với non-ML.

V. Kết luận

Trong bài báo này, chúng tôi đã trình bày phương pháp non-ML để dự đoán quang phổ và linear regression để dự đoán độ không chắc chắn. Phương pháp này sử dụng các kỹ thuật xử lý tín hiệu truyền thống và cho kết quả tốt hơn các phương pháp ML (KNN, GP, AE) chúng tôi đã thử.

Tuy vậy, các phương pháp học máy hiện đang dùng cho kết quả tốt hơn trên tập valid. Do một vài nguyên nhân khiến điểm test chưa tốt. Chúng tôi sẽ tiếp tục nghiên cứu để cải thiện hơn nữa các phương pháp này trong tương lai.

Phụ lục

A. *Tỉ lệ đóng góp*

Thành viên	Phan Bá Thọ	Nguyễn Quốc Huy	Trần Tuấn Anh
Tỉ lệ đóng góp	34%	33%	33%

Bảng I
Tỉ lệ đóng góp của các thành viên trong nhóm