2026 USA-North America AI Olympiad (USA-NA-AIO) Round 2 Problems Day 1

USA AI Olympiad (USAAIO)

April 4, 2026

Problem 1. Non Open-ended Problem: Linear Attention Is All You Need (90 Points)

Submission Requirements

1. Each contestant must submit a Jupiter Notebook (.ipynb). You should put all your solutions on this file.
2. Your submitted file must follow the format

   `Attn_LastName_FirstName_SchoolName`

3. Each part's name/number must appear in a text cell in bold section format, such as

   ```
   # **Part 1.3.4**
   ```

The attention mechanism is a central component of the transformer architecture introduced in *Attention Is All You Need*. In this problem, we study the computational properties of attention, explore **linear attention**, introduce **relative positional encoding**, and finally apply these techniques to build a transformer model for time-series classification.
Throughout this problem, we consider **self-attention only** and **ignore masking** for simplicity.

Part 1.1. Linear Attention

Assume the input sequence contains T tokens. For any attention head, let

$$Q = \begin{bmatrix} q_0^\top \\ q_1^\top \\ \vdots \\ q_{T-1}^\top \end{bmatrix}, \quad K = \begin{bmatrix} k_0^\top \\ k_1^\top \\ \vdots \\ k_{T-1}^\top \end{bmatrix}, \quad V = \begin{bmatrix} v_0^\top \\ v_1^\top \\ \vdots \\ v_{T-1}^\top \end{bmatrix} \in \mathbb{R}^{T \times d}.$$

For each token position $i \in \{0, 1, \ldots, T-1\}$,

$$q_i, k_i, v_i \in \mathbb{R}^{d \times 1}$$

are column vectors corresponding to the query, key, and value of the i-th token, respectively.
The self-attention output matrix is

$$O = \begin{bmatrix} o_0^\top \\ o_1^\top \\ \vdots \\ o_{T-1}^\top \end{bmatrix} = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right) V \in \mathbb{R}^{T \times d}$$

where

$$o_i \in \mathbb{R}^{d \times 1}$$

is a column vector corresponding to the output of the i-th token.

Part 1.1.1. (5 Points)

Analyze the time complexity of computing the self-attention output O. Express the complexity in terms of T and d using Big-O notation. Reasoning is required.

Part 1.1.2. (5 Points)

Suppose

$$\exp\left(\frac{q_i^\top k_j}{\sqrt{d}}\right) = \phi(q_i)^\top \phi(k_j),$$

where $\phi(\cdot)$ is a feature map function. Write the attention output vector o_t using the feature map representation. Reasoning is required.

Part 1.1.3. (5 Points)

Define

$$\phi(Q) = \begin{bmatrix} \phi(q_0)^\top \\ \phi(q_1)^\top \\ \vdots \\ \phi(q_{T-1})^\top \end{bmatrix}, \quad \phi(K) = \begin{bmatrix} \phi(k_0)^\top \\ \phi(k_1)^\top \\ \vdots \\ \phi(k_{T-1})^\top \end{bmatrix}.$$

Express the matrix O using $\phi(Q)$, $\phi(K)$ and V. Reasoning is required.

Part 1.1.4. (5 Points)

Suppose the feature map

$$\phi : \mathbb{R}^d \to \mathbb{R}^\ell.$$

Assume $\phi(q_i)$ and $\phi(k_j)$ have already been computed. Analyze the time complexity of computing O in terms of T, d, and ℓ. Reasoning is required.

Part 1.1.5. (5 Points)

Explain why the representation

$$\exp\left(\frac{q_i^\top k_j}{\sqrt{d}}\right) = \phi(q_i)^\top \phi(k_j)$$

requires $\ell = \infty$.
Hint:

$$\exp(x) = \sum_{r=0}^{\infty} \frac{x^r}{r!}.$$

Part 1.1.6. (5 Points)

Consider the truncated Taylor's expansion

$$\exp(x) \approx \sum_{r=0}^{2} \frac{x^r}{r!}.$$

Construct an explicit feature map $\phi(x)$ for $x \in \mathbb{R}^d$ and compute the feature dimension ℓ. Reasoning is required.

Part 1.2. Relative Positional Encoding

In many time-series datasets, the relationship between two time steps depends on their **relative distance** rather than their absolute positions.

Part 1.2.1. (5 Points)

Suppose the attention score is modified as

$$\frac{q_i^\top (k_j + r_{j-i})}{\sqrt{d}}$$

where the learnable parameter $r_{j-i} \in \mathbb{R}^{d\times 1}$ represents the **relative directional positional encoding** from token i to token j.
Explain when this encoding is more appropriate than absolute positional encoding.

Part 1.2.2. (5 Points)

Assume the sequence contains T tokens indexed as

$$0, 1, \ldots, T-1.$$

Determine the possible range of $j - i$. Reasoning is required.

Part 1.2.3. (5 Points)

For each token j, suppose each component p satisfies

$$\mathrm{Var}(k_{j,p}) = \sigma^2.$$

Suppose component p in r_{j-i} is initialized according to

$$r_{j-i,p} \sim \mathcal{N}(0, \tau^2).$$

Determine the appropriate value of τ^2. Reasoning is required.

Part 1.2.4. (5 Points)

Explain how relative positional encoding can be incorporated into the linear-attention framework developed earlier. Reasoning is required.

Part 1.3. Implementation of Linear Attention with Relative Positional Encoding

In this part, you are asked to build a PyTorch module called `MyLinearAttentionRelativePos` for multi-head self-attention with linear attention and relative positional encoding.

Starter Code Run the following starter code

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import matplotlib.pyplot as plt
```

You may use basic modules such as `nn.Linear`. You are not allowed to use high-level modules such as `nn.MultiheadAttention` or `nn.TransformerEncoder`. You are not allowed to import anything else beyond those provided in the starter code. You are not allowed to import any model from HuggingFace.

Hyperparameters Let d_{model} denote the length of each input token, T denote the number of tokens (a hyperparameter that needs to be instantiated), H denote the number of attention heads and d denote the length of each value vector for each token and each head. You should **not** assume that $Hd = d_{model}$.
Hint: In this problem, we treat T as a hyperparameter. This is because we will use this model to study a dataset that all samples are with equal length T.

Feature Map Function In this task, please use a feature map function defined with the ELU function. Recall that the ELU function is defined as

$$\text{ELU}(x) = \begin{cases} x & \text{if } x \geq 0 \\ e^x - 1 & \text{if } x < 0 \end{cases}.$$

Therefore, the feature map function is defined as

$$\phi(x) = 1 + \text{ELU}(x).$$

You can instantiate an ELU transformation by using `nn.ELU()`.

Part 1.3.1. (5 Points)

Write down the total number of learnable parameters of this module. Express your answer in terms of d_{model}, d, H, and T.
For simplicity, please ignore the bias term in any affine/linear transformation. Reasoning is required.

Part 1.3.2. (5 Points)

Plot $\text{ELU}(x)$ and $\phi(x)$ for $x \in [-3, 3]$. Put two plots in two separate figures.

Part 1.3.3. (10 Points)

Build the PyTorch class `MyLinearAttentionRelativePos` from scratch as a subclass of `nn.Module`.

Part 1.4. Transformer Model for Time-Series Classification (20 Points)

In this part you will build an **encoder-only transformer** by using the linear attention with relative positional encoding module that you build in the previous part. You then apply your transformer to the **FordA dataset**.
The FordA dataset is a time-series dataset derived from automotive sensor signals. Each sample consists of a time series of length **500**. Each sample belongs to one of **two classes**.
Your task is to train a transformer model to perform binary classification.

Requirements

- The transformer must use linear attention with relative positional encoding that you build in the previous part.
- No additional positional encoding is allowed at the input.

- You should build an encoder-only transformer from scratch. Any attempt to load a well-built attention block or encoder module (except those that you build above) is prohibited.

Feel free to import any needed libraries/modules/functions, as long as they are not against the above requirements.

Starter Code Please use the following URL:

```
https://drive.google.com/file/d/1ZpO63c-LJAQL-n9po9HXdrQl57DDbI
4K/view?usp=drive_link
```

Submission Beyond the Jupyter notebook, for this part, you should also submit a CSV file containing predictions for the test dataset.
All submitted files must follow the format

`FordA_LastName_FirstName_SchoolName`

GPU You must do training and testing with L4 GPU.

CSV Format The format of the CSV file must be the same as the sample submission file. It must contain two columns (in the sample submission file, the label column is empty):

test_sample_id label

Each row corresponds to one test sample.

Evaluation Your model will be evaluated using the **macro F1 score**. A random guessing baseline achieves approximately 0.5. Your goal is to achieve a higher score.

Final Competition Score Let X be the macro F1 score of your submission, X_{baseline} be the baseline score, and X_{best} be the best score among all submissions.
The final competition score is computed as

$$Score_{final} = \frac{X - X_{\text{baseline}}}{X_{\text{best}} - X_{\text{baseline}}} \times 100\%.$$

If $X < X_{\text{baseline}}$, then the final score is 0. The best submission receives 100% of the total score in this part.
In addition,

1. In your submitted CSV file, if your prediction in a sample has a missing value or a value that is not one of those two labels, your solution in this sample will be graded to be a wrong prediction.

2. If we find that you manually label the test dataset, you will receive 0 point in this part.

3. If your Jupiter Notebook does not run properly on L4 GPU, you will receive 0 point in this part.

Problem 2. Open-Ended Problem: Recovering a Single Source Central Field from Vector Field Observations (70 Points)

In this problem, you will solve an inverse physics problem involving a central force field generated by a single source.
We consider a two-dimensional continuous domain

$$[-1, 1] \times [-1, 1].$$

Within this domain there is one unknown source location

$$(x_s, y_s).$$

Your goal is to recover both the coordinates of the single source and the magnitude function of the force field as a function of distance from the source.

Central Field Model The force field is central, meaning that the force always points in the radial direction from the source.
For sample s, let

$$\vec{r}_s = (x_s, y_s)$$

denote the source location.
For any point

$$\vec{r} = (x, y),$$

the vector field has the form

$$\vec{F}_s(\vec{r}) = f_s(\|\vec{r} - \vec{r}_s\|) \frac{\vec{r} - \vec{r}_s}{\|\vec{r} - \vec{r}_s\|}.$$

Here

$$\|\vec{r} - \vec{r}_s\|$$

is the Euclidean (norm-2) distance from the source, and $f_s(r)$ is an unknown scalar function describing how the field magnitude depends on distance.

Unlike classical electric or gravitational fields, the explicit functional form of $f_s(r)$ is not provided. That is, you should not simply assume $f_s(r) \propto \frac{1}{r^2}$.
Please also note that f_s may take different forms for different samples.

Field Observations The vector field is measured on a regular 64×64 grid over the domain

$$[-1, 1] \times [-1, 1].$$

At each grid point we observe a vector

$$(F_x, F_y).$$

Thus each sample is represented by a tensor of shape

$$(64, 64, 2).$$

Dataset Description **Training dataset**
The training dataset contains 10,000 samples.
Each sample includes

- the vector field measurements,
- the true source location,
- the true values of the magnitude function at predefined radial checkpoints.

Validation dataset
A validation dataset containing 2,000 samples is also provided. It has the same format as the training dataset and can be used for model selection.
Test dataset
The test dataset contains 3,000 samples.
For each test sample you are given only the vector field observations.

Prediction Targets For each test sample s, you must predict the coordinates of the source

$$(x_s, y_s)$$

and the field magnitude

$$f_s(r_i)$$

evaluated at 100 predefined radii

$$r_i = 0.02 + 0.02i \qquad \text{for } i = 0, 1, \ldots, 99.$$

Thus the radii are

$$0.02, 0.04, 0.06, \ldots, 2.00.$$

Each prediction therefore contains

$$2 + 100 = 102$$

values.

Evaluation Metric The evaluation score consists of two components.

Source location error

For each sample s, for the source coordinates, we compute the mean absolute (norm-1) error

$$error_{s,\text{source}} = |x_s^{pred} - x_s^{true}| + |y_s^{pred} - y_s^{true}|.$$

Field magnitude error

For the magnitude function, we compute the root mean squared error

$$error_{s,\text{field}} = \sqrt{\frac{1}{100}\sum_{i=0}^{99}\left(f_s^{pred}(r_i) - f_s^{true}(r_i)\right)^2}.$$

Sample error

For each sample s,

$$error_s = 10^3 \cdot error_{s,\text{source}} + 3 \cdot 10^{-3} \cdot error_{s,\text{field}}.$$

Final evaluation score
Let N denote the number of test samples. The final evaluation score is

$$Score = \frac{1}{N} \sum_{s=0}^{N-1} error_s.$$

Lower scores indicate better performance.

Invalid Predictions For a missing or invalid predicted value:

1. If the prediction is about the source location, then for this sample s, $error_{s,\text{source}} = 10$.

   For instance, in sample s, if the prediction about `source_x` is valid (say, 0.312) but the prediction about `source_y` is missing, then $error_{s,\text{source}} = 10$ for this sample.

2. If the prediction is about the field magnitude, then for this sample s, $error_{s,\text{field}} = \frac{10^7}{3}$.

   For instance, in sample s, suppose the prediction about the field magnitude has 99 valid values and one missing value. Then $error_{s,\text{source}} = \frac{10^7}{3}$ for this sample.

Baseline Model A baseline solution treats the vector field as an image-like tensor. A convolutional neural network (CNN) is applied to the input tensor

$$(64, 64, 2)$$

and directly outputs 102 numbers:

- 2 numbers for the source location,
- 100 numbers for the magnitude values.

Final Competition Score Let

- X be your evaluation score,
- X_{baseline} be the baseline score,
- X_{expert} be the expert model score that has been developed by the jury and achieves a score significantly better the baseline score,
- $X_{\text{contestants}}$ be the best score among all contestants.

Please note that the expert model is for the jury's internal use for grading your solution. Contestants are not supposed to know this model.
Define

$$X_{\text{best}} = \min(X_{\text{expert}}, X_{\text{contestants}}).$$

Your final competition score is computed as

$$Score_{\text{final}} = \frac{X_{\text{baseline}} - X}{X_{\text{baseline}} - X_{\text{best}}} \times 100\%.$$

If

$$X > X_{\text{baseline}},$$

then the final score is 0.

Submission Requirements Each contestant must submit three files:

- a prediction file (.csv),
- a Jupyter notebook (.ipynb),
- a report (.docx).

File naming format
All submitted files must follow the format

`Central_Field_LastName_FirstName_SchoolName`

Submission Requirements Each contestant must submit three files:

- a prediction file (.csv),
- a Jupyter notebook with all your code (.ipynb),
- A report that describes the model you submitted in your .ipynb file, any other models you tried but did not adopt, and your thought process for this problem (.docx).

Starter Code Please use the following URL:

```
https://drive.google.com/file/d/1pD0upn_OBIn_iyvMtyqtyyc87j-6kdVP/view?usp=drive_link
```