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

USA AI Olympiad (USAAIO)

April 5, 2026

Problem 3. Non Open-ended Problem: Diffusion Models (50 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

   `Diffusion_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**
   ```

In this problem, we study the diffusion model, which is widely used in modern generative AI systems. For simplicity, we consider one-dimensional data, so all variables take values in $\mathbb{R}$.

Notation and Preliminaries For a random variable x, we write

$$x \sim \mathcal{N}(\mu, \sigma^2)$$

to denote that x follows a normal distribution with mean μ and variance σ^2.
The probability density function of a one-dimensional Gaussian distribution is

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

When conditioning on another variable y, we write

$$x \mid y \sim \mathcal{N}(\mu, \sigma^2)$$

to mean that conditional on y, the random variable x follows a normal distribution with mean μ and variance σ^2.
Define

$$\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s.$$

Part 3.1. Properties of the Forward Process

Let $0 < \alpha_t < 1$ be a sequence of parameters. The forward diffusion process is defined by

$$x_t = \sqrt{\alpha_t} x_{t-1} + \sqrt{1-\alpha_t}\, \epsilon_t,$$

where

$$\epsilon_t \sim \mathcal{N}(0, 1)$$

and the noise variables $\epsilon_1, \epsilon_2, \ldots$ are independent.
The sequence

$$x_0, x_1, x_2, \ldots, x_T$$

forms a Markov chain describing the gradual addition of Gaussian noise.

Part 3.1.1. (5 Points)

Mathematically prove that x_2 can be written in the form

$$x_2 = A_2 x_0 + B_2 \bar{\epsilon}_2,$$

where

$$\bar{\epsilon}_2 \sim \mathcal{N}(0, 1).$$

Find explicit expressions for A_2 and B_2.
Then, using this expression, conclude that the conditional distribution of x_2 given x_0 is Gaussian, and compute its mean and variance.
A complete derivation is required.

Part 3.1.2. (5 Points)

Mathematically prove that for any $t \geq 1$, x_t can be written in the form

$$x_t = A_t x_0 + B_t \bar{\epsilon}_t,$$

where

$$\bar{\epsilon}_t \sim \mathcal{N}(0, 1).$$

Find explicit expressions for A_t and B_t.
Then, using this expression, conclude that the conditional distribution of x_t given x_0 is Gaussian, and compute its mean and variance.
Your final answers should be written in terms of $\bar{\alpha}_t$ and x_0.
A complete derivation is required.

Part 3.1.3. (5 Points)

We have

$$x_t \mid x_0 \sim \mathcal{N}(\mu_{t|0}, \sigma_{t|0}^2).$$

Compute

$$\lim_{t\to\infty} \mu_{t|0} \quad \text{and} \quad \lim_{t\to\infty} \sigma_{t|0}^2.$$

Your final answers should be simplified as much as possible. Reasoning is required.

Part 3.2. Kullback–Leibler Divergence

Part 3.2.1. (5 Points)

Let q and p be two one-dimensional normal distributions defined as

$$q \sim \mathcal{N}(\mu_q, \sigma_q^2), \qquad p \sim \mathcal{N}(\mu_p, \sigma_p^2).$$

Compute the Kullback–Leibler divergence

$$D_{KL}(q \,\|\, p).$$

Your answer should be expressed in terms of $\mu_q, \mu_p, \sigma_q^2, \sigma_p^2$.
A complete derivation is required.

Part 3.2.2. (5 Points)

In the diffusion model, the true posterior distribution of the forward process is

$$q(x_{t-1} \mid x_t, x_0) \sim \mathcal{N}(\tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t),$$

while the reverse generative model is

$$p_\theta(x_{t-1} \mid x_t) \sim \mathcal{N}(\mu_\theta(x_t, t), \sigma_t^2).$$

Compute

$$D_{KL}\Big(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)\Big).$$

Your answer should be expressed in terms of

$$\tilde{\mu}_t(x_t, x_0),\ \mu_\theta(x_t, t),\ \tilde{\beta}_t,\ \sigma_t^2.$$

A complete derivation is required.

Part 3.2.3. (10 Points)

Compute $\tilde{\mu}_t(x_t, x_0)$ and $\tilde{\beta}_t$ in

$$q(x_{t-1} \mid x_t, x_0) \sim \mathcal{N}(\tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t).$$

Your final answers should be expressed in terms of x_t, x_0, α_t, and $\bar{\alpha}_t$.
A complete derivation is required.

Part 3.2.4. (5 Points)

Recall from Part 3.1.2 that x_t can be written in terms of x_0 and $\bar{\epsilon}_t$.
First express x_0 in terms of x_t and $\bar{\epsilon}_t$.
Then substitute this expression into the formula for $\tilde{\mu}_t(x_t, x_0)$ obtained in the previous part, and rewrite it in terms of $x_t, \bar{\epsilon}_t, \alpha_t$, and $\bar{\alpha}_t$.
A complete derivation is required.

Part 3.2.5. (5 Points)

Assume that

$$\sigma_t^2 = \tilde{\beta}_t.$$

Suppose that at time step t, the model outputs a quantity $\epsilon_\theta(x_t, t)$, and that the reverse-process mean $\mu_\theta(x_t, t)$ is written in terms of x_t and $\epsilon_\theta(x_t, t)$ in the same form as the expression obtained for $\tilde{\mu}_t(x_t, x_0)$ in the previous part.

Using this parameterization, compute

$$D_{KL}\Big(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)\Big).$$

Your final answer should be simplified as much as possible.
A complete derivation is required.

Part 3.3. Neural Network Parameterization (5 Points)

In the reverse generative process, we use a deep neural network to predict the noise term

$$\epsilon_\theta(x_t, t).$$

Suppose the input x_t is a tensor with shape

$$(B, C, L_0, L_1, L_2, \ldots, L_{M-1}),$$

where B denotes the batch size, C denotes the number of channels, and $L_0, L_1, \ldots, L_{M-1}$ denote the spatial or data dimensions.
The output $\epsilon_\theta(x_t, t)$ has the same tensor shape as x_t.
Because the function $\epsilon_\theta(x_t, t)$ depends on the time step t, explain how to design the architecture of the deep neural network so that the information of t is incorporated into the model.
Your answer should describe how the time step t is represented and how this representation is incorporated into the neural network.
No programming or implementation is required. Provide a conceptual explanation.

Problem 4. Open-Ended Problem: Classifying Geometric Shapes from RGB Images (40 Points)

In this problem, you will solve an image classification task involving simple geometric shapes.
Each sample consists of a single RGB image containing exactly one geometric object.
Your goal is to correctly classify the type of shape present in the image.

Image Model Each image is a color image of size

$$112 \times 112$$

with three color channels (RGB).
The background of the image is white.
Each image contains exactly one geometric shape, which can be one of the following:

- a circle,
- a triangle,
- a rectangle.

The shapes are generated with variability in the following aspects:

- **Position:** The shape may appear anywhere in the image.
- **Scale:** The size of the shape varies across samples.
- **Rotation:** Triangles and rectangles may be rotated by arbitrary angles.
- **Shape geometry:** Triangles are not necessarily equilateral.
- **Color:** Each shape is rendered in a randomly selected RGB color.

Dataset Description **Training dataset**
The training dataset contains 1,000 samples.
Each sample indexed from 0 to 199 includes

- an RGB image,
- the corresponding shape label.

Each sample indexed from 200 to 999 has an RGB image only (no ground-truth label).
Test dataset
The test dataset contains 300 samples.
For each test sample you are given only the RGB image.

Prediction Targets For each test sample s, you must predict a single label

$$y_s \in \{0, 1, 2\}.$$

The label definitions are:

- 0: circle,
- 1: triangle,
- 2: rectangle.

Evaluation Metric The evaluation score is macro $F1$. Higher scores indicate better performance.

Invalid Predictions For a missing or invalid prediction:

- If the predicted label is missing or not in $\{0, 1, 2\}$, it is treated as an incorrect prediction.

Prohibited Models **All pretrained models (regardless of whether you use their pretrained parameters) are prohibited, such as those imported from `torchvision.models` or HuggingFace. That is, you are required to build your own model and train it from scratch. If you violate this rule, you will get 0 point in this problem.**

Prohibited Label Imputation/Prediction Method For training data samples with missing labels, participants are not allowed to inspect the images and manually determine their labels using human vision. Participants may either exclude these samples from training or impute the missing labels using a coding-based method. Manual labeling based on visual inspection is strictly prohibited.
For the test dataset, participants must not predict labels by visually inspecting the images. All predictions must be generated using code.

Baseline Model The baseline solution is to make no predictions at all. Under this approach, the resulting score is zero.
We adopt this trivial baseline to provide a clear reference point. Any meaningful model should achieve a score above zero, demonstrating that it has learned useful information from the data.

Final Competition Score Let

- X be your score,
- X_{baseline} be the baseline score,
- X_{best} be the best score among all contestants.

Your final competition score is computed as

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

If

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

then the final score is 0.

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).

File naming format
All submitted files must follow the format

`Shape_Classification_LastName_FirstName_SchoolName`

Submission format
A sample submission CSV file is provided. Your submission must follow exactly the same structure as the sample submission file.

Starter Code Please use the following URL:

```
https://drive.google.com/file/d/15ISQTWZSSuVzuCRyNibaNAQVsuXH2ChB/view?usp=drive_link
```

Problem 5. Open-Ended Problem: Mixed Functions Parameter Regression (50 Points)

Problem Overview In this problem, data are organized into groups. Each group contains 400 observed data points (x, y).
The observations in each group are generated from a mixture of four mathematical functions:

- an exponential function
- a logarithmic function
- a trigonometric function
- a linear function

Each function contributes 100 data points, and Gaussian noise is added to the observations.
Thus, each group corresponds to one set of parameters describing the four functions. Your task is to use the 400 observed points in each group to estimate the parameters of all four functions.

Mathematical Models Each group contains data generated from the following four equations.

Exponential function

$$y = a_0 e^{a_1 x} + a_2$$

Logarithmic function

$$y = b_0 \ln(x) + b_1$$

Trigonometric function

$$y = c_0 \sin(c_1 x + c_2) + c_3$$

Linear function

$$y = d_0 x + d_1$$

Each group therefore contains 11 parameters:

$$a_0, a_1, a_2,\ b_0, b_1,\ c_0, c_1, c_2, c_3,\ d_0, d_1$$

Dataset Description **Training data**
The training dataset contains 400 groups.
The file `train.csv` contains observed data points with columns

$$\texttt{group\_id},\ x,\ y.$$

Each group contains 400 observations.
The file `train_params.csv` contains the true parameters for the training groups with columns

$$\texttt{group\_id}, a_0, a_1, a_2, b_0, b_1, c_0, c_1, c_2, c_3, d_0, d_1.$$

Test data
The test dataset contains 200 groups.
The file `test.csv` contains observed data points with columns

$$\texttt{group\_id},\ x,\ y.$$

Each group also contains 400 observations.

Task Using the training data, build a model that maps

$$400 \text{ observations } (x, y) \to 11 \text{ parameters.}$$

Then apply your model to the test dataset. For each group in `test.csv`, predict

$$a_0, a_1, a_2,\ b_0, b_1,\ c_0, c_1, c_2, c_3,\ d_0, d_1.$$

Evaluation Metric Submissions are evaluated using the Root Mean Squared Error (RMSE). The RMSE is computed across all predicted parameters for all groups:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{p}_i - p_i)^2}.$$

Here p_i denotes the true parameter value, $\hat{p}_i$ denotes the predicted parameter value, and N is the total number of predicted parameters across all groups. Lower RMSE indicates better performance.

Baseline Solution A baseline approach proceeds as follows.
For each group, the 400 observed points (x, y) are first partitioned into four clusters using a clustering method such as K-means with $k = 4$.
Since the data in each group are generated from four different functions, each cluster is expected to correspond approximately to one of the four function types.
Next, for each cluster, four candidate models are fitted: exponential, logarithmic, trigonometric, and linear. For each function type, the model that best fits one of the clusters is selected, and the corresponding parameters are used as the predicted parameters for that function.
Combining the parameters from the four fitted functions produces the 11 predicted parameters for that group.
The performance of this baseline solution defines the baseline RMSE, denoted by X_{baseline}.

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

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

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

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).

All submitted files must follow the format

`mix_functions_LastName_FirstName_SchoolName`

Hint If you knew which function each data point belongs to, you might consider the following to help you estimate parameters that modulate the function:

`scipy.optimize.curve_fit`

This can be found here:

`https://docs.scipy.org/doc/scipy/index.html`

Please use the following URL:

```
https://drive.google.com/file/d/134CU_q4kTEYa2umo71ybniEAq5sfHkf
X/view?usp=drive_link
```