Problem Statement 1- Computer Vision

Title: AI Art Restoration - Reviving Cultural Heritage

Motivation: Preserving cultural heritage is of paramount importance. While history has preserved countless masterpieces, the ravages of time have left many artworks faded, damaged, or on the brink of disappearance. Traditional restoration methods are often time-consuming and require extensive expertise. By resurrecting damaged or obscured artworks, we breathe life back into these forgotten stories, reviving the narratives that have shaped our collective consciousness. The participants are required to develop an innovative model that can automate the restoration process and ensure the longevity of art pieces for future generations.

Problem Statement: In the domain of art restoration, develop an innovative computer vision model capable of effectively restoring deteriorated images of art pieces. The goal is to create an automated solution that can enhance and reconstruct degraded artworks, improving their visual quality while preserving their original characteristics. The model should handle a diverse range of deterioration types, including but not limited to noise, blur, scratches, fading, and other common forms of degradation.

Objective: The objective of the hackathon is to design and implement an advanced computer vision model tailored for the restoration of deteriorated art images. Participants are encouraged to explore various techniques, architectures, and training methodologies to develop a robust and efficient solution. The primary focus should be on achieving accurate restoration results while maintaining the original aesthetics, colors, and details of the artwork. The resulting model should be practical, capable of handling diverse deterioration types, and applicable to a wide range of art styles, colors, and textures.

Training Dataset: No data will be provided in order to train the participants' models,they are expected to obtain this data by themselves. The training data should include a diverse range of art pieces with various deterioration types, such as noise, blur, scratches, fading, and other common forms of degradation. Each deterioration should have thousands of images. Potential sources for obtaining dataset could be scraping from the web using DuckDuckGo scraper, Yahoo API or BeautifulSoup library.

Validation and Test Datasets: The participants will be provided with a test dataset containing 50-100 images and a validation dataset of 10 images. The validation dataset is provided to help the participants get an understanding of the test data. In the validation dataset, each image will have a corresponding ground truth image representing the undamaged and ideal version of the artwork. The images provided in the dataset are 512x512 RGB images in JPG format.

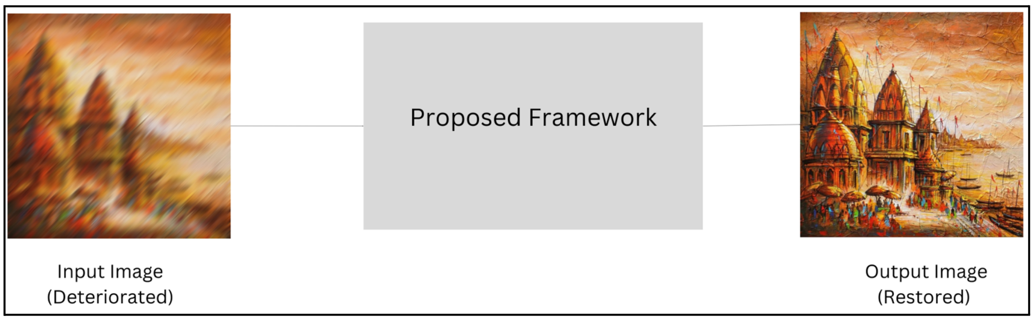
The validation and testing datasets will cover a range of deterioration types commonly found in art pieces, such as noise, blur, scratches, fading, water damage, discoloration, and tears. The degradation levels will vary, providing a challenge for the participants' models.

Tools and Libraries:

- Participants can use deep learning frameworks and libraries such as TensorFlow [1], PyTorch [2], or Keras [3] for implementing the model.
- Additionally, libraries such as OpenCV [4] or PIL [5] can be used for image manipulation and preprocessing tasks.

Model Architectures:

- Participants are encouraged to utilize a diverse range of models, including but not limited to Generative Adversarial Networks (GANs) [6] , U-Net [7] , Variational Autoencoders (VAEs) [8] , and other innovative approaches.
- Creative models will be highly valued as they foster innovation and contribute to the advancement of the field.



Evaluation Criteria:The models will be evaluated based on the following criteria:

- **Restoration Quality**: The restored images should accurately depict the original undamaged artwork, effectively reducing the impact of various deterioration types. The models will be assessed based on the quality of the restoration, preserving the aesthetics, colors, and details.
- **Generalization**: The models should demonstrate the ability to handle a wide range of deterioration types and art styles. Generalization across different types of degradation will be a significant factor in evaluating the models.
- **Innovation**: Participants are encouraged to explore innovative approaches, architectures, or training methodologies that go beyond existing models and techniques for art restoration. Novel ideas and techniques will be awarded higher scores after analysis by the panel of judges.
- **Model Performance**: The efficiency of the models will be analyzed using quantitative metrics, such as Peak Signal-to-Noise Ratio (PSNR) [9] and Structural Similarity Index (SSIM) [10].

Instructions:

- Participants are required to submit their trained model along with a report that includes all the steps they have followed to achieve their final predictions.
- The models can be of the formats .pth, .pt, .pb, or .h5.
- The report should include information about the model architecture, references taken, and any innovations introduced.

References:

[1] Tensorflow: https://www.tensorflow.org/api_docs
[2] PyTorch: https://pytorch.org/docs/stable/index.html
[3] Keras:https://keras.io/
[4] OpenCV:https://docs.opencv.org/4.x/
[5] PIL: https://pillow.readthedocs.io/en/stable/
[6] GANs : https://arxiv.org/pdf/1406.2661.pdf
[7] U-Net: https://arxiv.org/pdf/1505.04597.pdf
[8] VAEs: https://arxiv.org/pdf/1906.02691.pdf#chapter.2
[9] PSNR: https://in.mathworks.com/help/images/ref/psnr.html
[10] SSIM: https://en.wikipedia.org/wiki/Structural_similarity

Problem statement 2 - GPU

Title: Accurately modeling particle flow with Discrete Element Method using GPU Acceleration

Motivation: Discrete Element Method is a group of numerical methods that deals with the motion and effect of a large number of small particles. This is a simpler alternative to the Finite Element Method, which has largely been used to model the problem to follow. Discrete Element Method is preferred over Finite Element Method due to its lower computational requirements.This kind of problem finds its place wherever there is a need to model granular flow, like soil mechanics (Civil Engineering), powder mechanics (Mining/Manufacturing), heat transfer (Physics), chemical reactions (Chemistry) etc.

Problem Statement: The main challenge is to accurately model particle flow in the fastest possible time using GPU-accelerated CUDA parallel code. The setup is as follows:

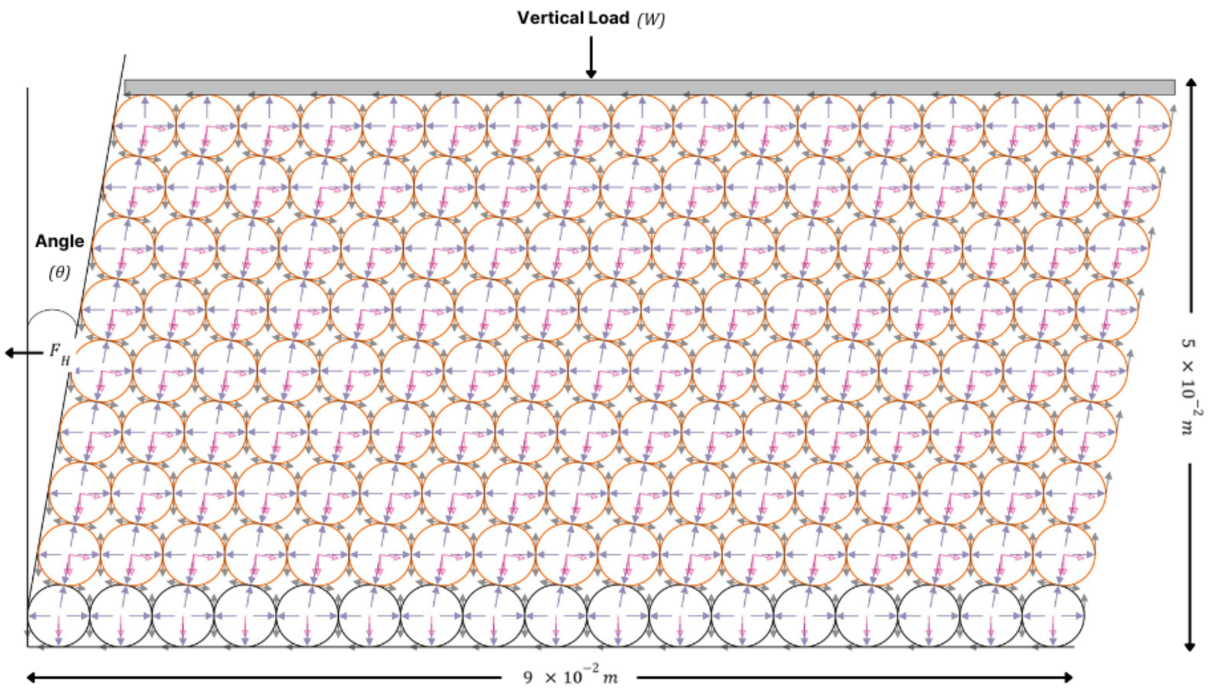


Fig. 1: Model setup with labeled inputs

This model contains 153 rollers, of which the ones outlined in orange need to be modeled. There is a load applied from top, a uniform vertical load. The task is to find the resultant horizontal force required to displace the model (F_H)[1] as the rollers slide over each other at a certain angle (θ). This can be calculated from the given strain rate (horizontal displacement per unit time)[2] while taking horizontal distance at a certain time and model height as follows:

$$\theta = tan^{-1}(horizontal\ displacement/model\ height)$$

Bottom row of the model is fixed (assumed to be undisturbed during the test). Rest of the rows will slide due to horizontal displacement. This results in the horizontal force mentioned.

Inputs will be the following:

1. 153 rollers distributed across 9 rows and 17 columns.
2. Vertical load applied (per sphere):
   $W = 8.7$ N
3. Initial dimensions of the model:
   a. Height = $5 \times 10^{-2}\, m$
   b. Length = $9 \times 10^{-2}\, m$
4. Roller properties:
   a. Radius (r) = $2.6 \times 10^{-3}\, m$
   b. Weight (W_r) = 3.5×10^{-2} N
   c. Strain rate: $1.25 \times 10^{-3}\, m/min$
5. Horizontal displacement ΔH (in mm) to be calculated from Strain rate.

Output expected: Horizontal force (F_H)

The mathematical formulation for the layers are as follows:

For all top layer, just below the vertical load (which is movable)

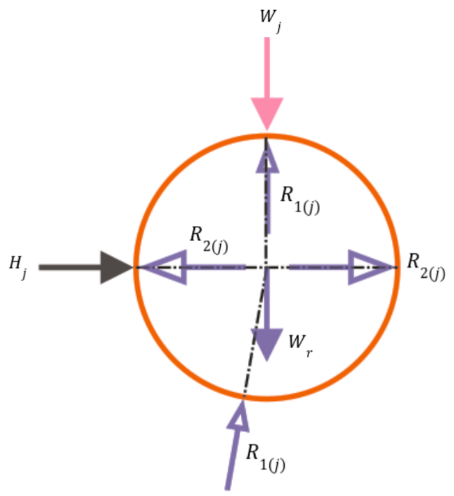


Fig. 2: Model setup for rollers in the top layer

Iterating over columns, denoted by j (from 1 to 17).

1. $W_r - R_{1(j)}(1 + cos\theta) + W = 0$

2. $H_j - 2R_{2(j)} + R_{1(j)}sin\theta = 0$

3. $17r(H_j - 2R_{2(j)}) + (W - R_{1(j)})[(2j - 1)r + 16rsin\theta] + W_r[(2j - 1)r + 16rsin\theta] - R_{1(j)}cos\theta[(2j - 1)r + 15rsin\theta] + R_{1(j)}sin\theta \times 16r = 0$

where $R_{1(j)}$, $R_{2(j)}$: Reaction forces as shown in Fig. 2

H_j: Unbalanced horizontal force of the roller in the j^{th} column

For all intermediate layers (which are movable)

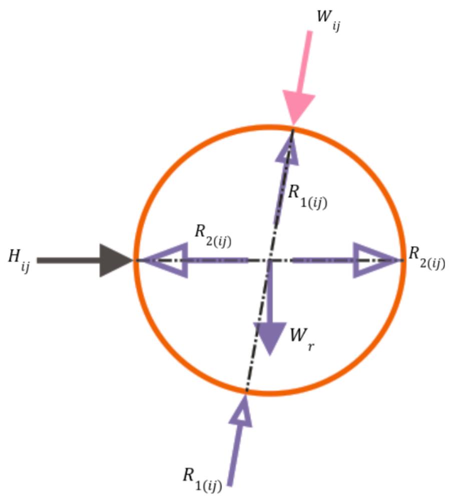


Fig. 3: Model setup for rollers in the intermediate layers

Iterating over rows, denoted by i (from 1 to 6). Column meanings (denoted by j) remain as usual.
Note: Rows starting at 1 ignores the bottom, fixed row for the generalized formula.

1. $W_r - 2R_{1(ij)}cos\theta + W_{ij}\, cos\theta = 0$

2. $H_{ij} + 2R_{1(ij)}sin\theta - 2R_2 - W_{ij}sin\theta = 0$

$$3.\quad H_{ij}(2i + 1)r - 2R_{2(ij)}(2i + 1)r + 4irR_{1(ij)}sin\theta + R_{1(ij)}sin\theta r - 2R_{1(ij)}cos\theta[(2j - 1)r + 2irsin\theta] + R_{1(ij)}cos\theta \times rsin\theta - 2(i + 1)rW_{ij}sin\theta + W_{ij}cos\theta[(2j - 1)r + (2i + 1)rsin\theta] + W_r[(2j - 1)r + 2irsin\theta] = 0$$

where

$W_{ij} = W + \sum_{i=1}^{7}(8 - i)W_r$

$R_{1(ij)}, R_{2(ij)}$: Reaction forces as shown in Fig. 3

H_{ij}: Unbalanced horizontal force of roller in the i^{th} row and j^{th} column

For bottom layer which is fixed (undisturbed during the test)

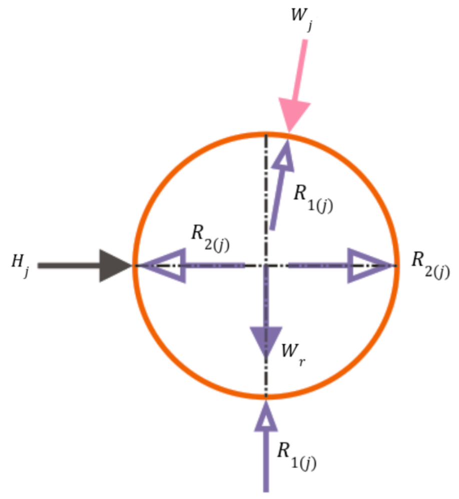


Fig. 4: Model setup for rollers in the bottom layer

Iterating over columns, denoted by j (from 1 to 17), we have the following equations the individual rollers in the bottom layer:

$$1.\quad W_r - R_{1(j)}(1 + cos\theta) + W_j cos\theta = 0$$

$$2.\quad H_j - 2R_{2(j)} + (R_{1(j)} - W_j)sin\theta = 0$$

3. $H_j r - 2R_{2(j)} r - W_j sin\theta r + W_j cos\theta[(2j - 1)r + rsin\theta] - R_{1(j)}(2j - 1)r[1 + cos\theta] + R_{1(j)} sin\theta r + W_r r = 0$

where

$W_j = W + 8W_r$

$R_{1(j)}, R_{2(j)}$: Reaction forces as shown in Fig. 4

H_j: Unbalanced horizontal force of the roller in the j^{th} column

To calculate the Horizontal force (F_H)

$$F_H = \sum_{j=1}^{17} H_j \text{ (top layer)} + \sum_{i=1}^{6}\sum_{j=1}^{17} H_{ij} \text{ (intermediate layers)} + \sum_{j=1}^{17} H_j \text{ (bottom layer)}$$

Objective: Despite this simplification, as particle numbers increase, so does computation, which leads to increasing times on consumer processors. The objective of this problem statement is to reduce or speed-up some of that computation and effectively model the given scenario. The given scenario is novel and the computation has not been parallelized yet[3].

Instructions:

1. Sequential code to be parallelized is made available in this GitHub repository: https://github.com/dipyamanroy/ICECTI-Hackathon-GPU/. The equations solved sequentially are those presented in the preceding sections. Comments have been provided in the program that demarcates the three different cases, viz. top layer, intermediate layers and the bottom layer.
2. Given problem/sequential program is to be parallelized on a GPU accelerator using CUDA C.
3. Program must be executed using the GPU accelerator on Google Colab (Free version). Accessed via Runtime → Change Runtime Type → Hardware accelerator → GPU. This program must be single-precision.
4. Access to 1xNVIDIA A100 Tensor Core GPU will be provided by Mahindra University to 5-8 shortlisted entrants. The code to be run here must be double-precision. This will be communicated in due time.

Evaluation criteria:

1. Accuracy versus experimental results will be tested.
2. Parallelism demonstrated will be evaluated using established routines. Examples of such routines can be Amdahl's law and Gustafson's law.[4]

3. Speed-up versus sequential code will be tested. This is the runtime required for the parallel program to run versus the sequential program on Google Colab (Free version).

Submission:
Source code and a report mentioning the approach must be privately shared via Google Forms. Source code must be shared in the form of a Google Colab notebook link, and report must be shared as a .pdf file uploaded to Google Drive, whose link will be taken.

Problem Statement 3 – Natural language processing

Title: AI-Assisted Learning for NVIDIA SDKs and Toolkits

Problem Statement: Develop an AI-powered language model (LLM) that assists users in understanding and effectively using various NVIDIA SDKs (Software Development Kits) and toolkits. The objective of this hackathon is to create an interactive and user-friendly platform that provides comprehensive information, examples, and guidance on NVIDIA's SDKs and toolkits. By leveraging the power of language models and NVIDIA toolkits, participants aim to simplify the learning curve for developers and empower them to utilize NVIDIA's technologies more efficiently.
Note that you can either use a pre-trained model and modify it according to the task or you can develop one from scratch.

To provide you with an idea, here are some illustrative inquiries that can be directed towards the LLM. When posed with these questions or queries, the LLM should respond with answers in and around the following context:

Question	Expected Response
1. What is the NVIDIA CUDA Toolkit?	The CUDA Toolkit from NVIDIA provides everything you need to develop GPU-accelerated applications. The CUDA Toolkit includes GPU-accelerated libraries, a compiler, development tools and the CUDA runtime.
2. How can I install the NVIDIA CUDA Toolkit on Windows?	The setup of CUDA development tools on a system running the appropriate version of Windows consists of a few simple steps: 1. Verify the system has a CUDA-capable GPU. 2. Download the NVIDIA CUDA Toolkit. 3. Install the NVIDIA CUDA Toolkit. 4. Test that the installed software runs correctly and communicates with the hardware.

3. What are the key features of the NVIDIA TensorRT?	NVIDIA® TensorRT™, an SDK for high-performance deep learning inference, includes a deep learning inference optimizer and runtime that delivers low latency and high throughput for inference applications.Some key features of TensorRT include: - NVIDIA TensorRT-based applications perform up to 36X faster than CPU-only platforms during inference, enabling you to optimize neural network models - TensorRT, built on the NVIDIA CUDA® parallel programming model, enables you to optimize inference using techniques such as quantization, layer and tensor fusion, kernel tuning, and others on NVIDIA GPUs. - For more information one can check out https://developer.nvidia.com/tensorrt
4 What is the difference between NVIDIA's BioMegatron and Megatron 530B LLM?	BioMegatron focuses specifically on biomedical NLP tasks and has been trained on relevant biomedical data, Megatron 530B LLM is a more general-purpose language model trained on a wide variety of text from different domains. The choice between the two models depends on the specific requirements and domain of your NLP task..

Dataset for the Challenge:

To build an AI-powered language model (LLM) for assisting users in understanding and effectively using various NVIDIA SDKs and toolkits, the dataset can be composed of the following sources:

1. **NVIDIA Documentation:** The primary data source would be the extensive documentation provided by NVIDIA for their SDKs and toolkits. This documentation contains comprehensive information, usage guidelines, examples, and troubleshooting details for each SDK and toolkit. It covers topics such as installation, API references, sample code, and best practices. Link: https://docs.nvidia.com/

2. **Internet Articles and Tutorials:** Alongside the official documentation, incorporating relevant articles and tutorials from the internet can enhance the dataset. Blog posts, tutorials, and guides authored by developers and technology enthusiasts provide real-world insights, use cases, and tips for effectively utilizing NVIDIA SDKs and toolkits.

3. **Community Forums and Q&A Platforms:** Community forums like NVIDIA Developer Forums, question-and-answer platforms such as Stack Overflow, and discussions on GitHub can serve as valuable sources of information. These platforms host discussions, provide solutions to common issues, and address user queries related to NVIDIA SDKs and toolkits.
4. Incorporating data from these sources enables the language model to offer practical

guidance and address specific user concerns. Link: https://forums.developer.nvidia.com

By combining the NVIDIA documentation with curated internet articles and insights from community forums, the language model can be trained to effectively respond to a wide range of user queries related to NVIDIA SDKs and toolkits. This approach ensures the model is up-to-date, incorporating information from official sources as well as the broader developer community

Evaluation Criteria: These below factors/criteria will be evaluated by a review panel**.**

1. **Authenticity:** Assessing the correctness of the information provided by the language model in response to user queries related to NVIDIA SDKs and toolkits.

2. **Comprehensiveness:** Evaluating the extent to which the language model provides complete and thorough information on NVIDIA's SDKs and toolkits, covering key features, APIs, usage guidelines, and best practices.

3. **Contextual Understanding:** Evaluating the model's comprehension of user queries and its ability to provide relevant responses and explanations tailored to the specific SDK or toolkit being discussed.

4. **Consistency:** Same or similar prompts should generate identical or almost identical responses related to NVIDIA SDKs and toolkits.

5. **Speed:** The speed at which the model can produce results is important, especially when it needs to be deployed for critical use cases.

6. **Grammar and Readability:** The model must generate language in a readable format. Ensuring proper grammar and sentence structure is essential.

7. **Interactive Assistance:** Assessing the model's interactivity and responsiveness in assisting users with their queries, offering step-by-step guidance, code snippets, and troubleshooting tips.

By considering these aspects, the language model can be evaluated holistically, ensuring they meet the needs of users and developers utilizing NVIDIA's SDKs and toolkits.

Submission instructions:

1. **Submission Components:** Submissions must include the following components:
   a. Trained Model: The language model trained for providing information on

NVIDIA SDKs and toolkits.

b. Training Process Details: A detailed document (A pdf file preferably) outlining the steps taken in data collection, data preprocessing, model architecture design, model training, hyperparameter tuning and training time/duration.

c. Testing Notebook: Include a notebook (.ipynb) with clear instructions on how to load the model and how to ask a query to test our hold-out test dataset. Ensure the notebook is executable and produces accurate results when run. We will execute them to verify their functionality. Failure to run these notebooks successfully will lead to disqualification of the submission.

d. Example Responses: A collection of example responses generated by your model, showcasing its capabilities in providing informative and accurate information related to NVIDIA SDKs and toolkits. Provide necessary screenshots.

e. Report: One page report/summary of the work should be submitted in the provided template.