INFO 4000
Informatics III

Data Science Specialization - Advanced

Week9 SVD Application, Interpretability of NNs, PINN, Digital Twins

Course Instructor

Jagannath Rao

raoj@uga.edu

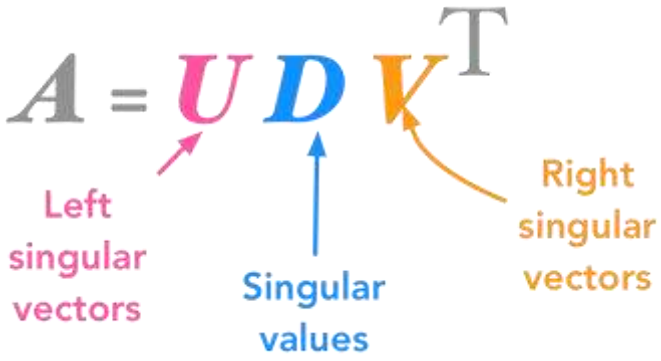
MP2

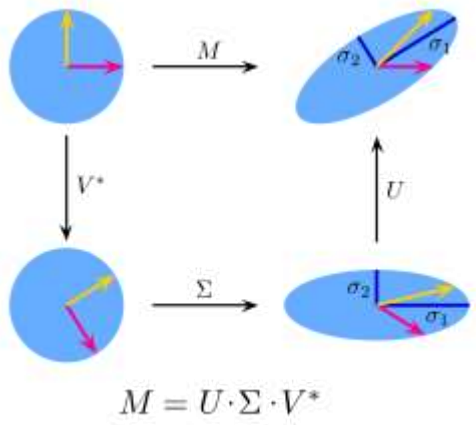
MP2 Briefing

- The MP2 is a 2-part mini project
  - Part 1: Is a RL DQN problem of training an Inventory Management agent to learn to optimize inventory, cashflow and profitability of a business.
  - Part 2: Is an Audio Analytics problem of detecting faults in faulty valves purely from the sound the produce.
- For part 1, you will use a Python Class that is provided to you in "**Inventory_env_class.py**", which behaves exactly like the Gymnasium environments except that it is independent of that. Instruction about the class and how to use it is provided. Test it by import the class and giving it some random actions to make sure you have got that going. (Demo)
- For part 2, a data set of valve sounds is provided the way they were collected, and they were collected at different points of time and hence are in different folders and need to be consolidated programmatically.
- The weightage for grading for the two parts are 50-50.
- Please read instructions very carefully for both parts before you start solving the assignment.
- MP2 will be due on October 28th by midnight.

Recap

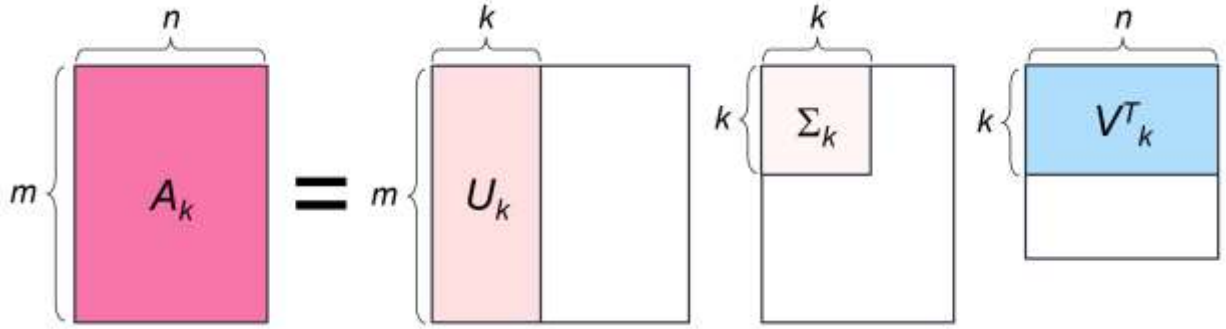
Singular Value Decomposition (SVD)





- (SVD) is a factorization method in linear algebra that decomposes a matrix into three other matrices, providing a way to represent data in terms of its singular values.
- The singular values indicate the importance of the corresponding dimensions (eigenvectors) in the data, with larger values representing more significant components.
- They define a new coordinate system where the data's most important variations are captured along the first few axes.
- The SVD technique used for dimensionality reduction, data compression, and noise reduction.
- **Natural Language Processing (NLP):** Extracting semantic relationships between words and documents.
- **Image Compression:** Reducing the size of an image by discarding less important singular values.
- **Noise Reduction:** Separating meaningful signals from noise by ignoring the singular values associated with noise

What does SVD do?



$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \ldots.$$

- **Left Singular Matrix U:** Captures patterns across the rows of the matrix A. Each column of U corresponds to a direction in the row space of A, sorted by significance.
- **Right Singular Matrix V^T:** Captures patterns and structures related to the **columns** of A. Each column of V^T corresponds to a direction in the column space of A.
- Σ represent singular values which say which gives us a ranking of the most important features in the data

What SVD means to Image data?

- Image data
  - **Original Data Matrix A:** The matrix where each element is the intensity value of a pixel.
  - **Columns of U:** These vectors can be viewed as "basis images" that, when combined, can reconstruct the original image. The first few vectors represent the most significant visual patterns, such as the general shape and overall lighting.
  - **Columns of V:** These vectors capture the dominant features and structure of the image in a different dimension.
  - **Singular Values Sigma:** The largest singular values correspond to the singular vectors that capture the most prominent features. Smaller singular values correspond to noise or finer details
- For image compression, you can perform a truncated SVD by keeping only the largest singular values and their corresponding vectors. When you reconstruct the image using this truncated information, you get a recognizable, though lower-quality, image that uses much less data.
- This helps Deep Learning models to focus on the main features, does not overfit on noise and converges much faster.

Let us look at some applications of SVD applications

Neural Network Interpretability
Get into the “black-box”

Visualizing NN activations in a layer to refine model: Example 4

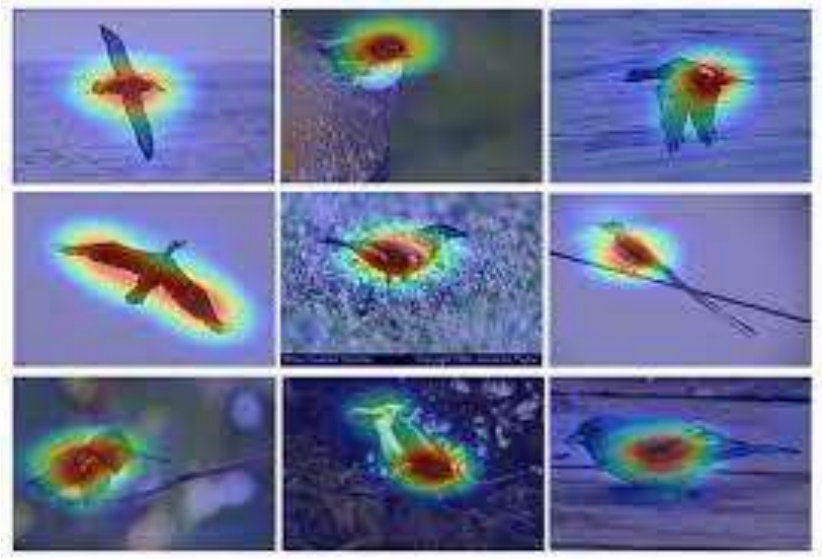
- A layer activations heatmap, often called a "Class Activation Map" (CAM), visually shows which parts of an image are most important for a convolutional neural network (CNN) to make a prediction.
- It is essentially highlighting the regions in an image that most strongly activate the neurons in a specific layer of the model, thus providing insights into how the network is "looking" at the image to make its decision;
- Useful for understanding the reasoning behind a model's predictions, identifying potential biases, and debugging issues with the network architecture.
- By overlaying the heatmap on the original image, you can see which areas are considered most relevant by the model, allowing for easier interpretation of the prediction.
- If the heatmap highlights irrelevant areas of an image, it could indicate that the model is focusing on incorrect features, allowing developers to identify and fix issues with the network.
- Analyzing heatmaps can guide adjustments to the network architecture or training process to improve its performance by focusing on the most relevant features.
- The interpretation of a heatmap depends on the specific model architecture and training data used.

Activation Map for Class Siberian Husky

Saliency Maps: Example 5

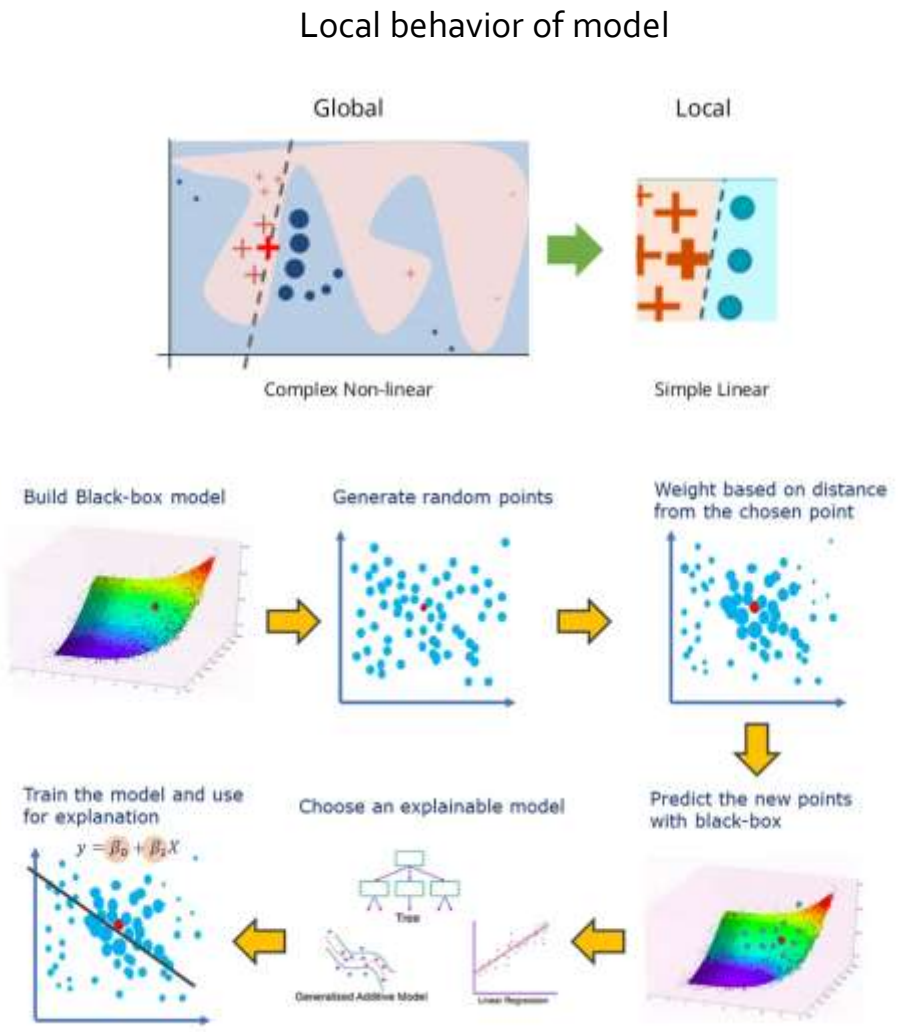
- **Saliency map:** is an image that highlights the most relevant regions for machine learning models.
- In a Saliency Map, the brightness of a pixel represents how salient the pixel is i.e brightness of a pixel is directly proportional to its saliency.
- **Improve interpretability**: Saliency maps help people understand the decision-making process of AI models, which can improve trust and confidence.
- **Help identify biases** : Saliency maps can help identify biases in datasets and help prepare models to be more robust against adversarial attacks.

Saliency Map



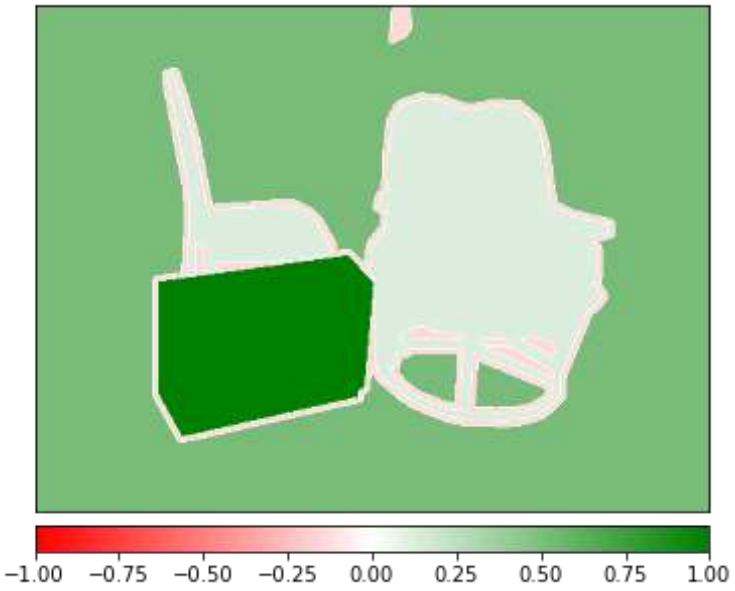
LIME (Local Interpretable Model-Agnostic Explanations) – Example 6

- **Perturbation:** LIME perturbs the input data by making small modifications (e.g., removing or slightly altering features in the input).
- **Fitting a Simple Model:** LIME fits a simple, interpretable model (e.g., linear regression) to these perturbed data points to approximate the behavior of the complex model near the prediction.
- Interpretability: This simple model tells you which features of the input were most important for the specific prediction.
- **Model-agnostic:** It works with any black-box model (deep learning, decision trees, etc.).
- **Local explanations:** It provides clear insights into individual predictions.
- **Example Use Case:** You're using a complex neural network to classify images. By applying LIME to a single prediction, you can understand which parts of the image were most influential in that prediction.



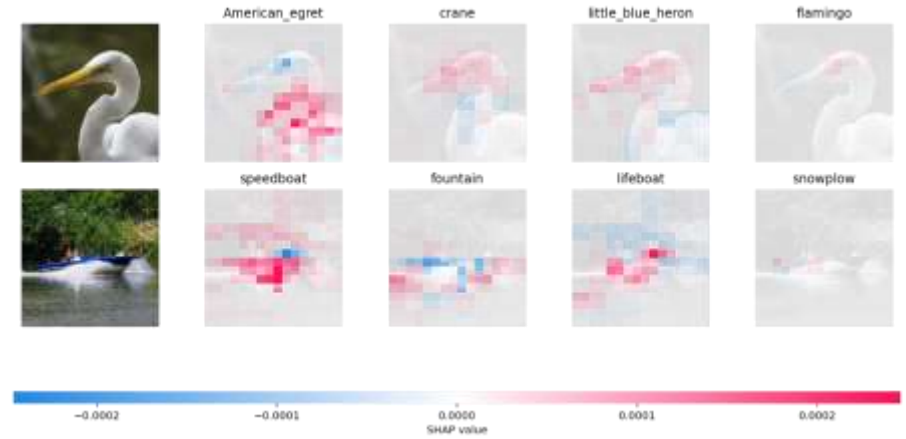
LIME – Image Segmentation Example

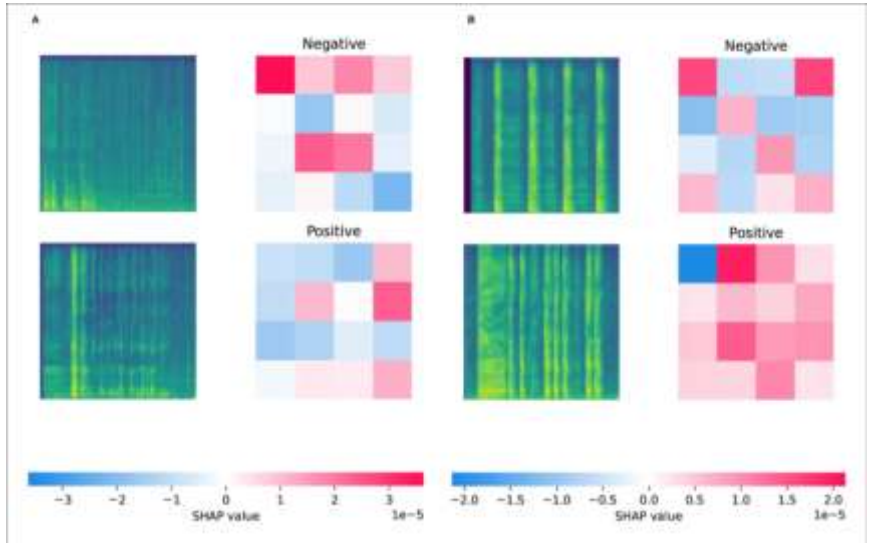
- Image Segmentation: LIME breaks the image into several super pixels (segments of the image).
- Perturbation: It perturbs or changes these super pixels by turning them on and off (masking) to see how the model's prediction changes when different parts of the image are modified.
- Explanation: LIME builds a simple, interpretable model based on these perturbed examples, then it highlights the super pixels that contribute most to the model's final prediction.
- Highlighted regions (super pixels) that LIME has identified as being **important for the model's decision**.



SHAP – Example 7

- SHAP values are based on game theory (Shapley values), and they measure the contribution of each feature to the model's prediction by considering all possible combinations of features.
- SHAP provides both local (per prediction) and global (overall model behavior) interpretability. It ensures that the contributions of features are fairly distributed based on their marginal contributions to the prediction.
- **Shapley Values:** These are originally from cooperative game theory and represent how to fairly distribute the "payout" (prediction) among all the "players" (features). SHAP values are a way of calculating this for machine learning.
- **Fairness Principle:** SHAP computes the contribution of each feature to the prediction by averaging its contribution across all possible feature combinations. This ensures that each feature gets a fair share of credit (or blame) for the prediction.
- **SHAP Explainer Models:** SHAP uses different methods like TreeSHAP (for tree-based models) or KernelSHAP (for black-box models) to compute these values efficiently.
- **Example Use Case:** You have a complex model predicting whether a loan should be approved. By using SHAP values, you can understand which features (e.g., income, credit score) contribute most to the prediction and ensure that the model is making decisions fairly.





Physics Informed Neural Networks (PINNs) - Introduction

What are PINNs?

- Imagine you have two students trying to solve a physics problem, say, predicting the path of a thrown ball
  - "Standard" Student -A Regular Neural Network, who has seen a set of examples and tries to find a pattern in it.
  - However, if you ask the NN to find the path if thrown from a new spot it will make a will guess.
  - This is because it does not understand the underlying physics.
  - The "Smart" Student (A Physics-Informed Neural Network): This student has not only seen a few examples but has also studied the laws of gravity and motion (the "physics").
  - They use this knowledge to guide their predictions. Even with limited data, they can make a much more accurate prediction because their answers must follow the rules of physics.
- That's the core idea of a PINN! It's a "smarter" type of neural network that doesn't just learn from data; it also incorporates our understanding of the physical world.
- You can define PINNs as neural networks that are trained to respect the laws of physics.

Why does it work so well?

- A standard neural network's test only has one question: "How well did you match the data?"
- A PINN's test has two questions:
  - "How well did you match the data?"
  - "How well did you follow the laws of physics?"
- The PINN tries to get a good score on both parts of the test.
- Applications:
  - Weather Forecasting: Instead of just looking at past weather data, a PINN can also use the equations that govern atmospheric fluid dynamics to make more accurate predictions.
  - Engineering and Design: Imagine designing a new airplane wing. A PINN can simulate how air will flow over the wing, helping engineers create more efficient designs. This is faster than traditional simulation methods.
  - Healthcare: PINNs can model blood flow in arteries or how a drug spreads through the body, helping doctors understand diseases and treatments better.
  - Predicting the trajectory of a ball: As in our analogy, a PINN can accurately predict where a ball will go, even with only a few data points, because it understands gravity.

Why PINNs?

- What makes them so useful?
  - They need less data: Because they already "know" the physics, they don't need as many examples to learn from. This is great for situations where collecting data is expensive or difficult.
  - PINNs can tackle problems that are very difficult to solve with traditional methods, like those involving complex geometries or incomplete information.
  - In some cases, PINNs can run simulations much faster than traditional physics simulators.
- Some challenges are:
  - Computational Cost: Training these models can require a lot of computing power.
  - We need to know the physics: PINNs work best when we have a good understanding of the physical laws governing the system. If our understanding is wrong, the PINN's predictions will be too.
  - Complexity of real-world physics: For very complex physical phenomena, the equations might be too difficult to incorporate effectively.

Example:

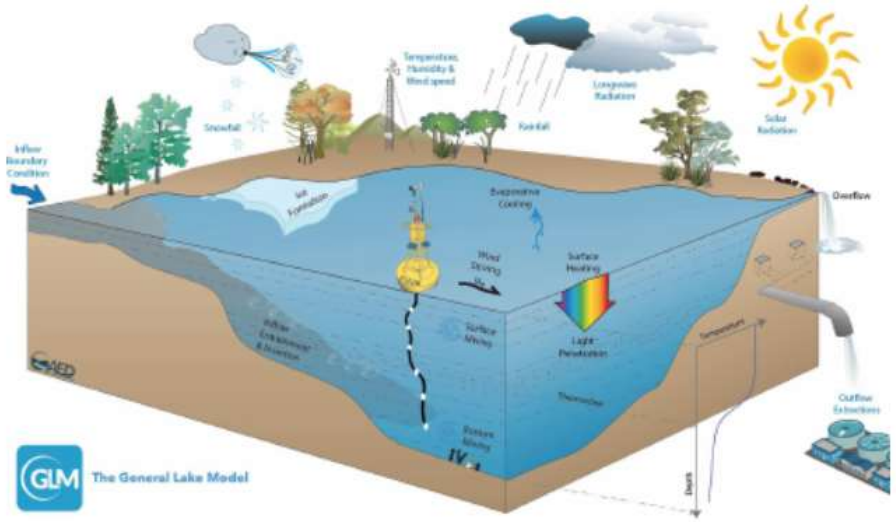
Physics-based model that captures the processes governing the dynamics of temperature in a lake

- With sufficient information about the current situation, a well-made physics-based model enables simulating lake water temperature, which is critical for understanding the impact of changing climate on aquatic ecosystems and assisting in aquatic resource management decisions.

- The task is to develop a model that can predict the water temperature as a function of depth and time, for a given lake which is governed by a variety of physical processes pictorially shown in Fig.

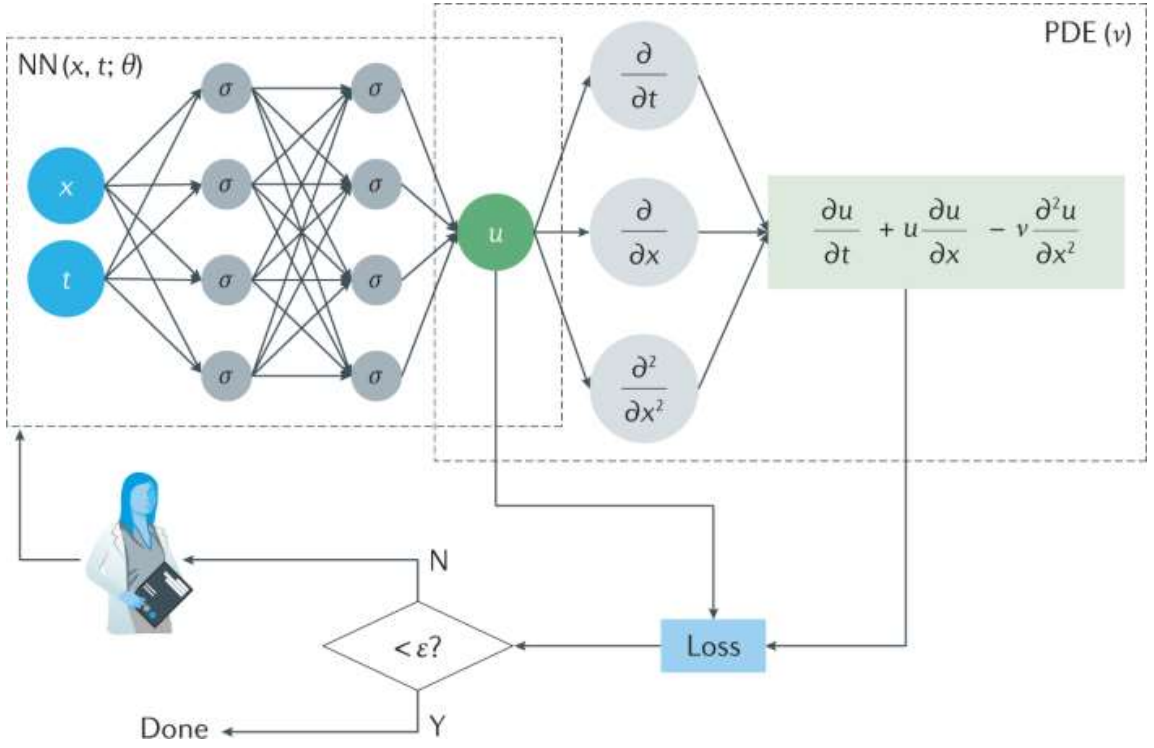
- The temperature of water in a Heating of the water surface due to incoming shortwave, radiation from the sun, the attenuation of radiation beneath the surface and the mixing of layers with varying energies at different depths, and the dissipation of heat from the surface of the lake via evaporation or longwave radiation

- Example of applying the physical law part:
  - The relationship between water temperature and its density is known.
  - Hence predictions should follow the fact that the deeper the point, the higher the predicted density.
  - If for a pair of points, the model predicts higher density for the point closer to the surface, this is a physically inconsistent prediction.
  - So, we can punish the model by penalizing the cost function.



The “Magic” happens in the formulation of the “Loss” function

- The key to a PINN is its special loss function. Think of the loss function as a "scorecard" that tells the neural network how well it's doing.
- Unlike a standard neural network that only cares about matching the data, a PINN's scorecard has two parts.
  - **NN Loss:** We take our known data points (our historic dataset), we feed the t and x into the neural network, and we compute the MSE.
  - **Physics Loss:** the output u of the neural network isn't just a final answer. We can actually calculate its derivatives with respect to the inputs (like ∂u/∂t, ∂u/∂x, ∂²u/∂x²)
  - We then plug these derivatives into our physics equation (the Partial Differential Equation,). In your image, it should satisfy: ∂u/∂t + u * ∂u/∂x - v * ∂²u/∂x² = 0.
  - Any result other than zero represents a "physics error." This error is our physics loss (often called Loss_physics).
- The calculation: The total loss is a weighted sum of the data loss and the physics loss. →Total Loss = Loss_data + λ * Loss_physics
- The neural network's goal is to minimize this Total Loss.



Building a PINN

1. This is the simple heat transfer equation also know as the diffusion equation

2. u(x,t) is the temperature at position x and time t, α is the thermal diffusivity (a constant).

3. You create a neural network where the input is x (position) and t (time), and the output is u(x,t) (temperature).

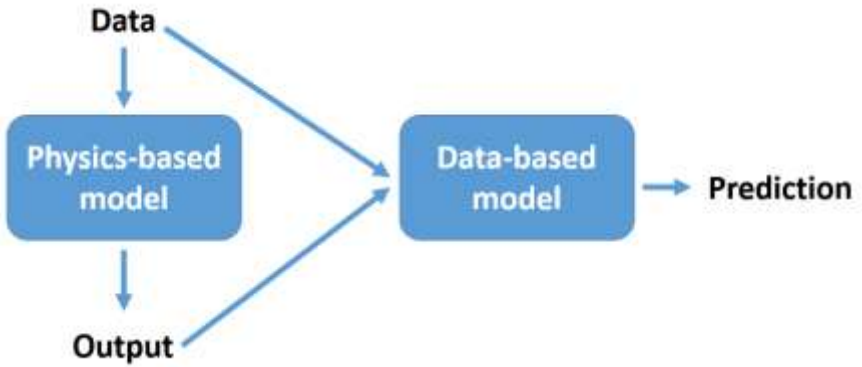
4. If you have some temperature data at certain points (say initial conditions or boundary conditions), you use these data points to train the network to match observed values.

5. The key part of PINNs is that you **add a term to the loss function that forces the network to respect the heat equation**.

6. You use automatic differentiation from PyTorch (a technique neural networks already use) to compute derivatives

7. The loss function would include not just the difference between predicted and actual data, but also the residual of the heat equation:
   Loss=Data Loss + Physics

$$\frac{\partial u(x,t)}{\partial t} = \alpha \frac{\partial^2 u(x,t)}{\partial x^2}$$



PyTorch – Auto Differentiation

How Differentiation with torch helps in building PINN

1. In PyTorch, you can compute differentials (derivatives) by using automatic differentiation, which is provided by the **torch.autograd** module.

2. PyTorch automatically computes gradients of tensors that have the **requires_grad=True** attribute set, which is especially useful when working with functions or models where you want to calculate the derivative with respect to inputs or parameters.

3. In Physics-Informed Neural Networks (PINNs), we typically need to compute first-order and second-order partial derivatives with respect to the input variables (e.g., time and space).

4. Let us take the heat equation: $\frac{\partial u}{\partial t} - \alpha \frac{\partial^2 u}{\partial x^2} = 0$

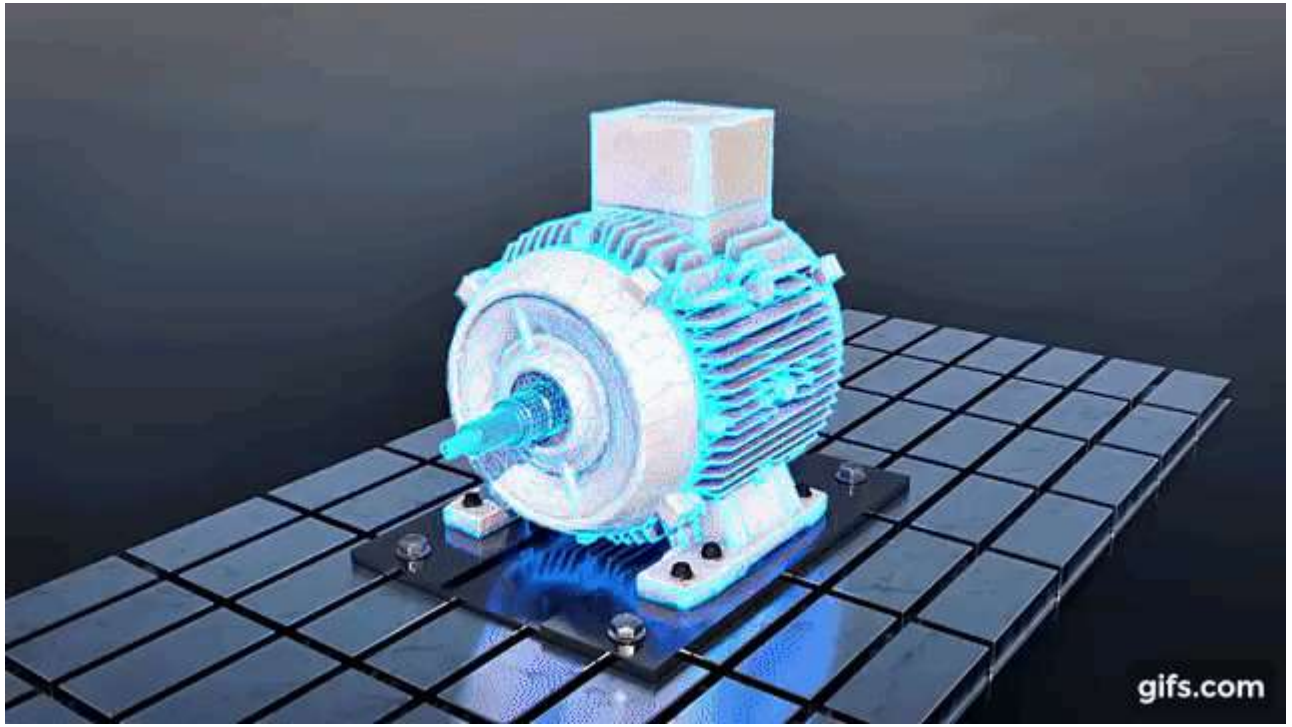
5. Suppose, we build a model to predict the temperature in a metal rod at position 'x' and time 't', the model will output a single, scalar value for 'u' which let us say represents temperature.

6. In PINN, the output of the model we are building must obey the above law of thermodynamics. In other words, if we do the first order partial differential w.r.t the output 'u' and then differentiate that differential to get the second order partial differential and substitute it in the above equation (alpha being a constant for the material), the result should be 'zero'.

7. If the result is not zero (or very close to zero) we have a physics loss.

8. Therefore, we want to be able to compute these differentials.

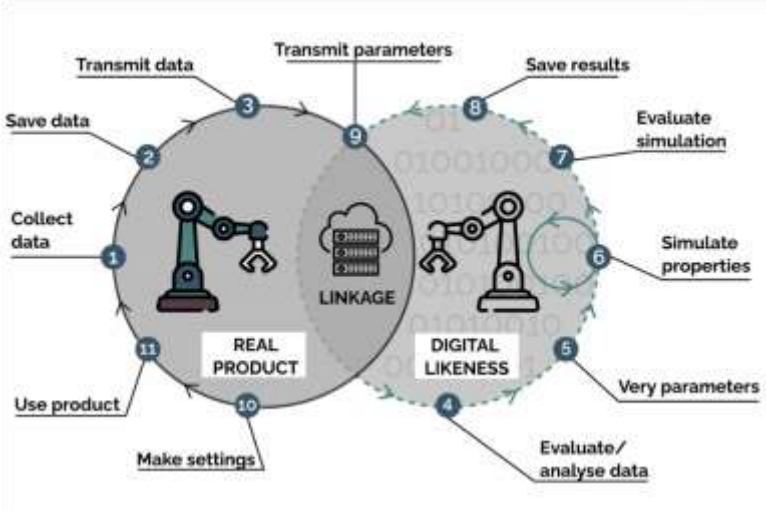
PINN:
Coded Examples of building blocks



Digital Twins

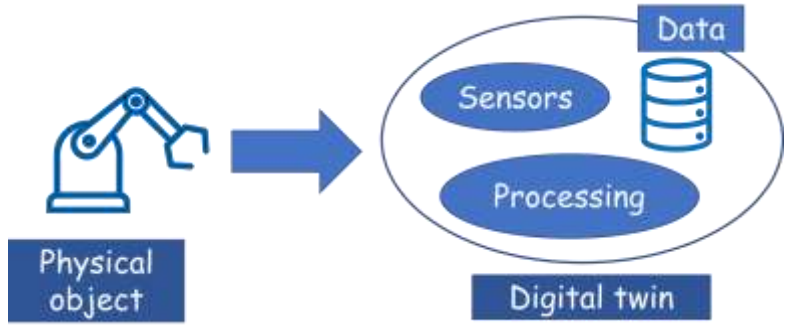
Digital Twin - Basics

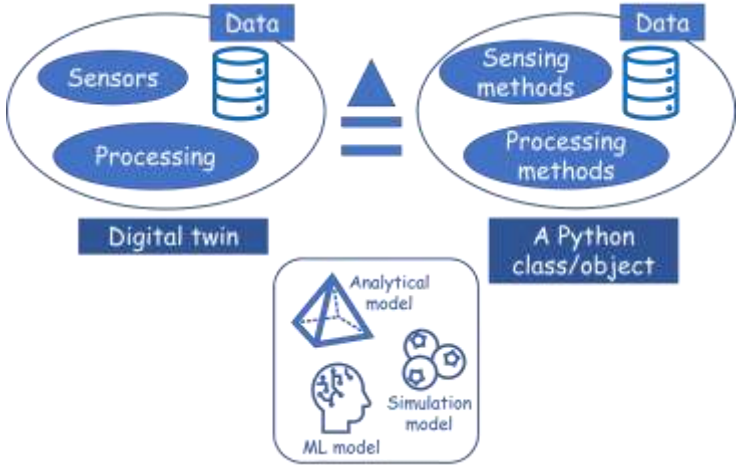
- Digital Twins are functional models of real systems.
- They are what we would call – 3D modeling + physics model.
- The Digital Twin can simulate the complete operation of a system.
- We can get insights and test operational conditions of the real system with a software driven model.
- We will learn by building a Digital Twin of a simple product in python.



Digital Twin - Schematic

- Main enabling factors for creating a digital twin are the **sensors** that gather data and the **processing system**
- Then we can insert the data in into the digital copy of the object.
- Once informed with such data, the virtual model can be used to **run simulations, study performance** issues and **generate possible improvements**
- We can *emulate* the sensors and data processors with suitable methods/functions, store the gathered data in a database or internal variables, and *encapsulate* everything into a Python class.
- The Python object
  - can be used in suitable simulation programs,
  - can be probed for data, and
  - can even be subject to optimization routines for enhancing suitable internal parameters.
- We start simple and add sophistication as we go along.





Digital Twin – A simple example with temperature control system
The Thermostat

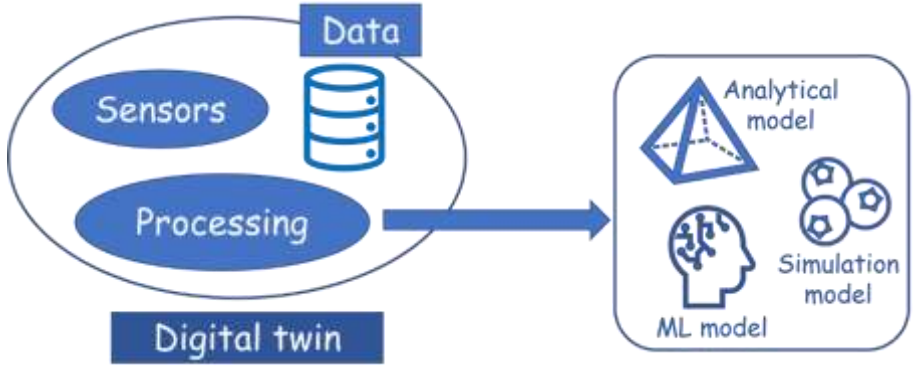
- Simulating a **closed-loop temperature control system.**
- A **PhysicalRoom class** to better simulate the physical world, including its own thermal dynamics (heating, cooling, and natural drift towards an ambient temperature).
- A set_point in the ThermostatTwin, representing the desired target temperature.
- Control logic within the twin that uses hysteresis (a dead-band) to decide when to turn the heater or A/C on or off, preventing rapid cycling.
- A feedback loop where the twin's command is sent back to the Physical Room to influence its state, thus creating a true control system.
- The ambient temperature is the temperature of the environment outside the room.
  - It represents the natural condition the room will revert to without heating or cooling.
  - If the room is heated to 24°C but the ambient (outside) temperature is 15°C, the room will naturally lose heat and cool down towards 15°C whenever the heater is off.
  - Conversely, if it were a hot day (e.g., ambient temp of 35°C), an air-conditioned room at 24°C would naturally heat up towards 35°C when the A/C is off.

Creating the Digital Twin of the Thermostat in a physical environment

- Create a Python Class which simulates the room dynamics with the following methods
  - **__init__:** which initializes the values of initial & ambient temperature and also sets the heat and cool rate
  - **get_temperature**: which retrieves the current temperature of the room
  - **apply_dynamics**: increases or decreases the temperature of the room when the HVAC commands are given
- Create another Python Class which simulates the thermostat with the following methods
  - **__init__**: initializes current temperature and set points and the time stamp
  - **get_control_command**: Here resides the core logic where the thermostat decides what they HVAC system should do
  - **update_state**: Receives feedback from the physical system and updates the digital twin

Use of ML in Digital Twins

- Digital Twins are largely made with analytical models and calculations from mathematical models of the physical process.
- This does not necessarily need tools from data science or machine learning. However, in the modern world, dominated by data-driven models, it is prudent to use such modeling techniques wherever applicable and suitable.
- In the Thermostat example
  - There are variables like cooling rate, heating rate and how quickly the room temperature changes with ambient temperature
  - There is no real data available for this and can only be measured by sensors and collected
  - However, if we have enough data over time, we could build a model to predict these rates for various ambient & room temperatures and incorporate it into the control system
- The reason we choose a ML model here and not an analytical model s
  - measurement of these values is often noisy
  - Actual value is also somewhat stochastic, being a strong function of the natural variability of the room dynamics over the seasons.



Let us see how the Digital Twin of the THERMOSTAT is built

End of Lesson