Week 02

canonNet Approach

1. Data collection
   - We collected 250 stl models and manually labelled them as flat or free form.
2. Converted the STL Files to Point Clouds
   - What it means is we picked 1024 random points from the surface of each model. These points are enough for us to roughly describe the shape of the object. Each point has (x, y, z) coordinates. We do this because it's easier for the neural network to process. We used the Trimesh library to do this and saved the points as .npy files for later use.
3. Preprocessing: Canonicalization (This is a preprocessing step)
   - Canonicalization standardizes point clouds so the model can learn shape features, not rotations:
     I. ***Centering*:** Move the point cloud so its center is at the origin. This is done by Computing the average of all points and subtract it from all points.
     II. ***PCA Alignment*:** Then we used PCA to rotate the model so it's always in the same direction. We applied pca here because this helps the network learn the shape instead of getting confused by how the model is placed.
     III. ***Sorting Points*:** Then we sorted points in a consistent order.
4. Match Point Clouds with Labels
   - Match each canonicalized point cloud with its label from the CSV file (the one we checked manually)
5. CanonNet Model
   - Then we made a simple neural network called CanonNet. The inputs we gave to this model are just the coordinates of the 1024 canonicalized points that represent each 3D model. The network learns from these points and outputs a prediction, whether the model is flat or free form.

6. Model Training

- We divide the data into batches using PyTorch Dataset (holds all the data) and DataLoader (gives the model small pieces of data step by step).
- What happens during training:
    - ➢ The model sees a batch of point clouds (the 3D data) and it tries to guess if each one is flat or free form.
    - ➢ It compares its guesses with the correct answers (labels) and the difference is called the error or loss.
    - ➢ Using a process called backpropagation, the model figures out which parts of itself (the weights) caused the error.
    - ➢ And it updates itself Adam optimizer adjusts the weights slightly so that next time, the model makes better guesses.

- (When we were training the model, we didn’t give all the data to the model at once. Instead, we used batches, which help the model to learn step by step, so the model learns step by step. Also, during training, the model learns the shapes, for example, flat objects usually have smooth, even surfaces, while free-form objects are more uneven and curved.)

7. Model Evaluation

- We keep aside some data that the model hasn’t seen during training (validation data).
- We check how well the model performs by calculating:
    - o Accuracy: The percentage of correct predictions out of all predictions.
    - o Confusion Matrix: A table showing how many times the model was correct or made mistakes for each class (flat or free form).

8. Using the Model on New Data

- To classify a new STL model:
    - o Convert it into a point cloud.
    - o Canonicalize the point cloud (center, align, and sort the points).
    - o Feed it into the trained CanonNet model to get a prediction.
- Then the model predicts whether each new model is flat or free form.

Group Members
Janitha
Pooja
Thinaya