MLPR | SEM 5

INDOOR NAVIGATION

Team:

Aman, Anshika, Anushka

PROBLEM?

- Plaksha’s makerspace is dynamic; a creative environment where students, faculty, and staff engage in a wide range of hands-on activities, from 3D printing to electronics fabrication to woodworking to sewing.
- These spaces, often characterized by **complex layouts with machines, and workstations**, and can pose **navigational challenges**, particularly for first-time users.
- For **campus tours**, the **lack of an autonomous guiding mechanism** hinders the potential for self-guided, interactive explorations of the space.
- **Human assistance is limited** during peak hours or off-hours, thereby reducing the availability of guided help. A **plethora of directional signs and information overwhelm visitors**, hindering rather than aiding navigation.
- Efficient navigation within these spaces is crucial to maximize productivity and ensure safety.

PROBLEM?

- The current challenge in the university **makerspace** is twofold:
    - **Efficiently guiding individuals**, including faculty and visitors, to specific sections within its segmented layout
    - Facilitating **real time autonomous campus tours** without the need for human assistance.

SOLUTION

Interactive Navigation Assistant

Real-time dynamic maps that **updates based on the user's location** and desired destination.

An intelligent, **automated navigation tool** tailored for the makerspace region

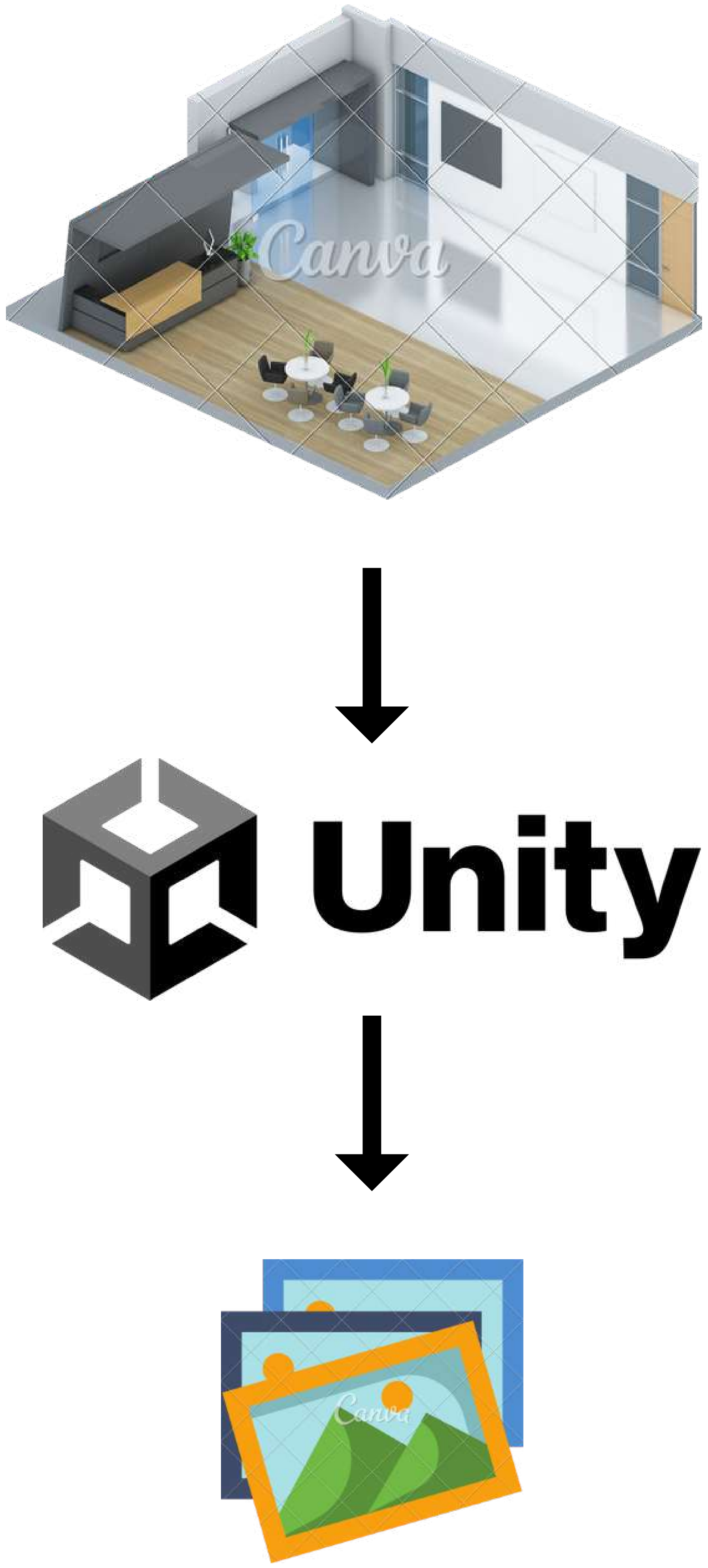
Anticipated Impact

- Enhancing Visitor Experience
- Operational Improvements
- Accessibility and Inclusivity

LITERATURE REVIEW

- Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images
- Learning to Detect Scence Landmarks for Camera Localization
- OrienterNet: Visual Localization in 2D Public Maps with Neural Matching
- Back to the Feature: Learning Robust Camera Localization from Pixels to Pose


HELP
EDIT
AR VIEW
MEASURE
SHARE


Unity


(0.27, 0.00, -0.54)_1.2_0
(0.27, 0.00, -0.54)_1.2_30
(0.27, 0.00, -0.54)_1.2_60
(0.27, 0.00, -0.54)_1.2_90
(0.27, 0.00, -0.54)_1.2_120
(0.27, 0.00, -0.54)_1.2_150
(0.27, 0.00, -0.54)_1.2_180
(0.27, 0.00, -0.54)_1.2_210
(0.27, 0.00, -0.54)_1.2_240
(0.27, 0.00, -0.54)_1.2_270
(0.27, 0.00, -0.54)_1.2_300
(0.27, 0.00, -0.54)_1.2_330
(0.27, 0.00, -0.54)_1.4_0
(0.27, 0.00, -0.54)_1.4_30
(0.27, 0.00, -0.54)_1.4_60
(0.27, 0.00, -0.54)_1.4_90
(0.27, 0.00, -0.54)_1.4_120
(0.27, 0.00, -0.54)_1.4_150
(0.27, 0.00, -0.54)_1.4_180
(0.27, 0.00, -0.54)_1.4_210
(0.27, 0.00, -0.54)_1.4_240
(0.27, 0.00, -0.54)_1.4_270
(0.27, 0.00, -0.54)_1.4_300
(0.27, 0.00, -0.54)_1.4_330
(0.27, 0.00, -0.54)_1.6_0
(0.27, 0.00, -0.54)_1.6_30
(0.27, 0.00, -0.54)_1.6_60
(0.27, 0.00, -0.54)_1.6_90
(0.27, 0.00, -0.54)_1.6_120
(0.27, 0.00, -0.54)_1.6_150
(0.27, 0.00, -0.54)_1.6_180
(0.27, 0.00, -0.54)_1.6_210
(0.27, 0.00, -0.54)_1.6_240
(0.27, 0.00, -0.54)_1.6_270
(0.27, 0.00, -0.54)_1.6_300
(0.27, 0.00, -0.54)_1.6_330
(0.27, 0.00, -0.54)_1.8_0
(0.27, 0.00, -0.54)_1.8_30
(0.27, 0.00, -0.54)_1.8_60
(0.27, 0.00, -0.54)_1.8_90
(0.27, 0.00, -0.54)_1.8_120
(0.27, 0.00, -0.54)_1.8_150
(0.27, 0.00, -0.54)_1.8_180
(0.27, 0.00, -0.54)_1.8_210
(0.27, 0.00, -0.54)_1.8_240
(0.27, 0.00, -0.54)_1.8_270
(0.27, 0.00, -0.54)_1.8_300
(0.27, 0.00, -0.54)_1.8_330
(0.27, 0.00, -0.54)_1_0
(0.27, 0.00, -0.54)_1_30
(0.27, 0.00, -0.54)_1_60
(0.27, 0.00, -0.54)_1_90
(0.27, 0.00, -0.54)_1_120
(0.27, 0.00, -0.54)_1_150
(0.27, 0.00, -0.54)_1_180
(0.27, 0.00, -0.54)_1_210
(0.27, 0.00, -0.54)_1_240
(0.27, 0.00, -0.54)_1_270
(0.27, 0.00, -0.54)_1_300
(0.27, 0.00, -0.54)_1_330

FEATURE EXTRACTION

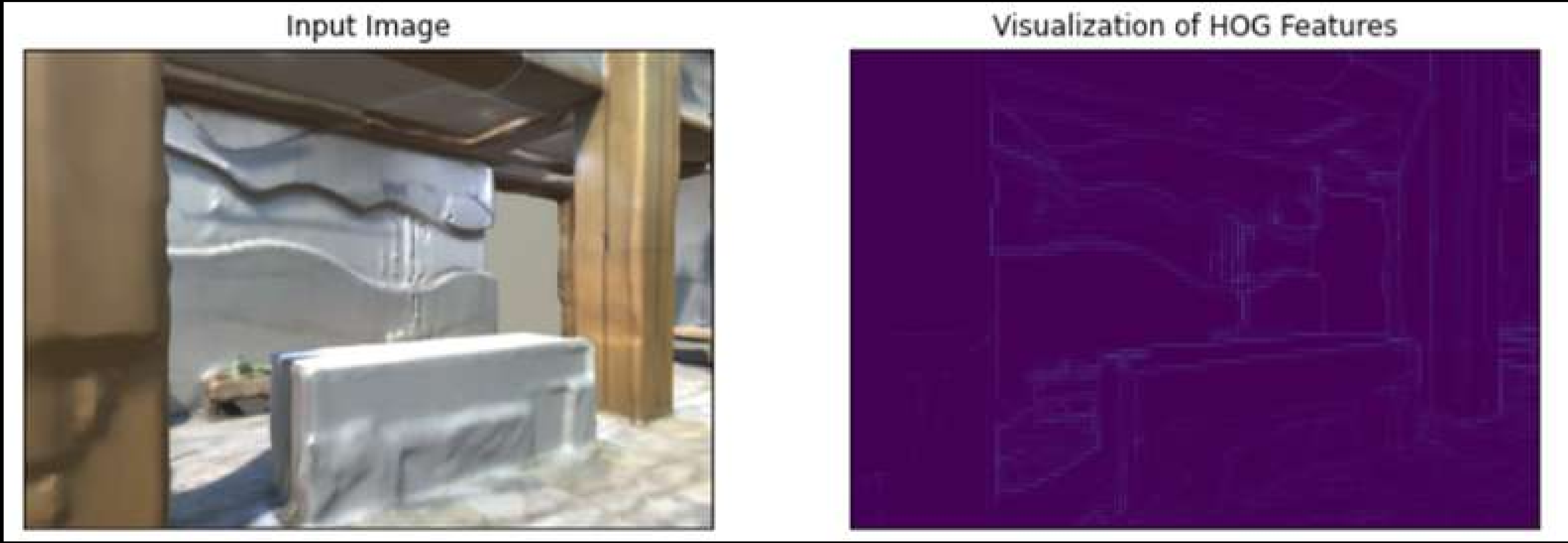
HOG for Feature Extraction

Extracts distinctive image features by quantifying the orientation of gradients. It captures object shapes and textures by constructing histograms of color and brightness directions and then transforms them into a feature vector.

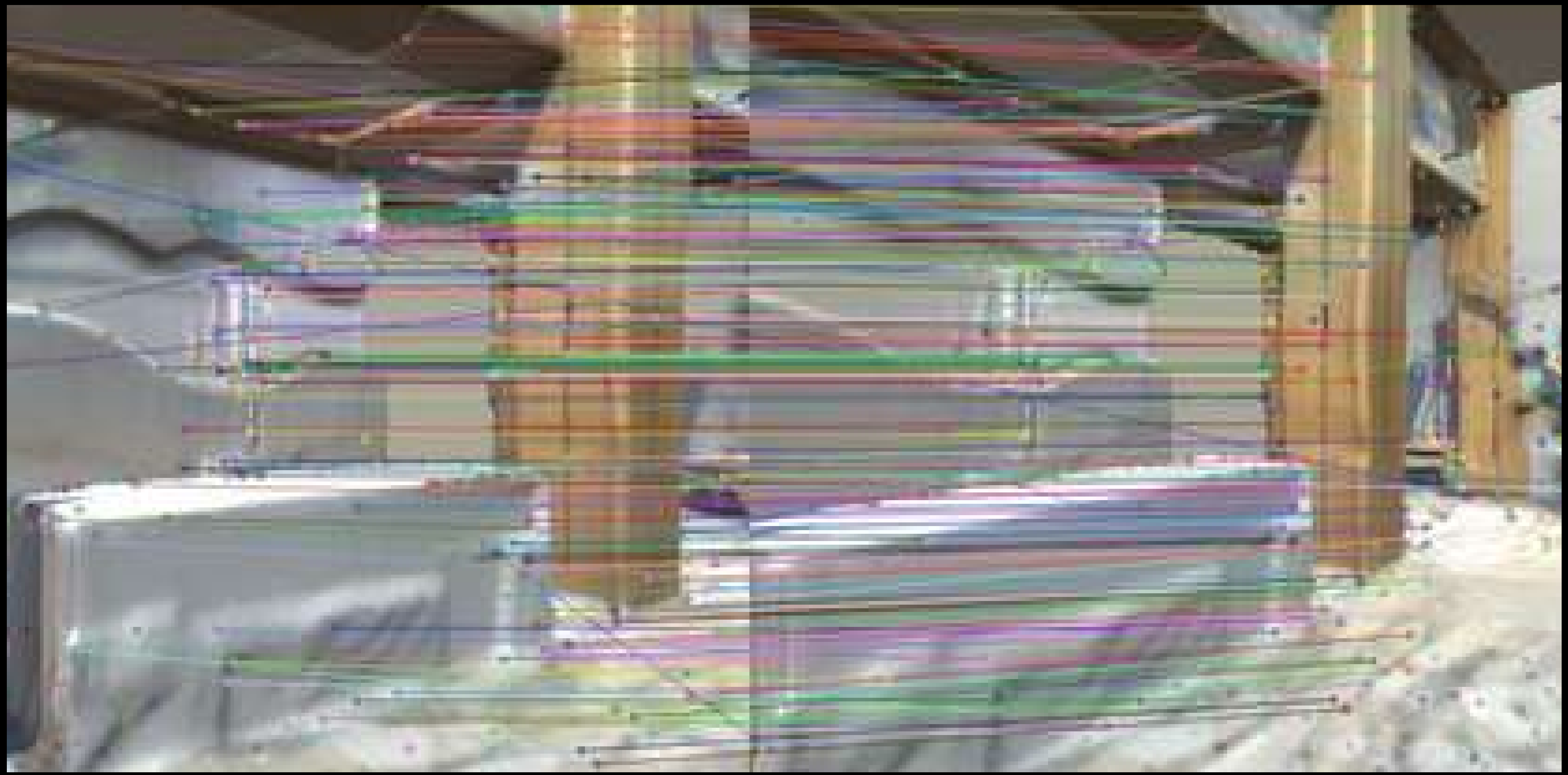
SIFT and ORB

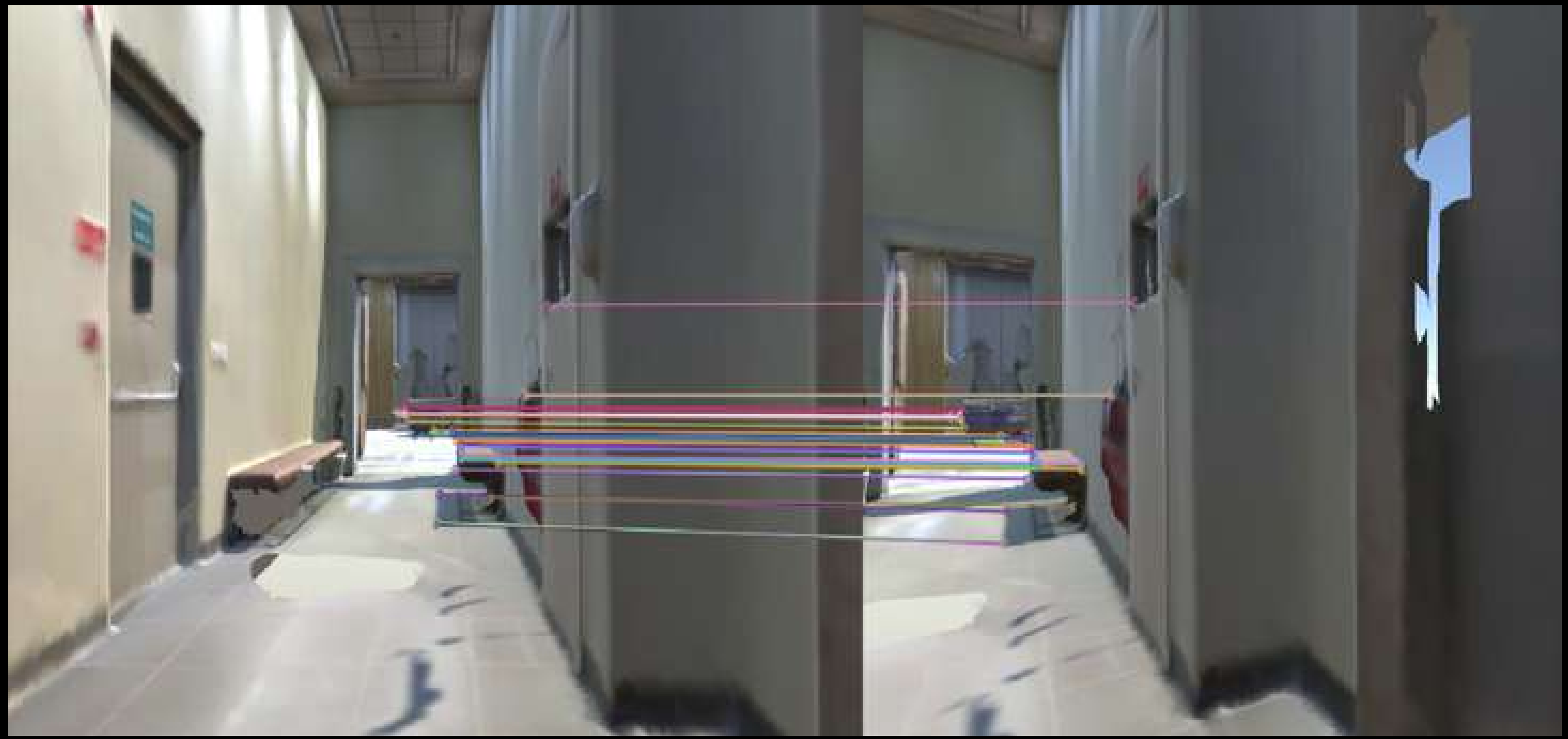
Feature descriptors like SIFT or ORB are designed to be invariant to scale and rotation, making them suitable for multi-view recognition

HOG

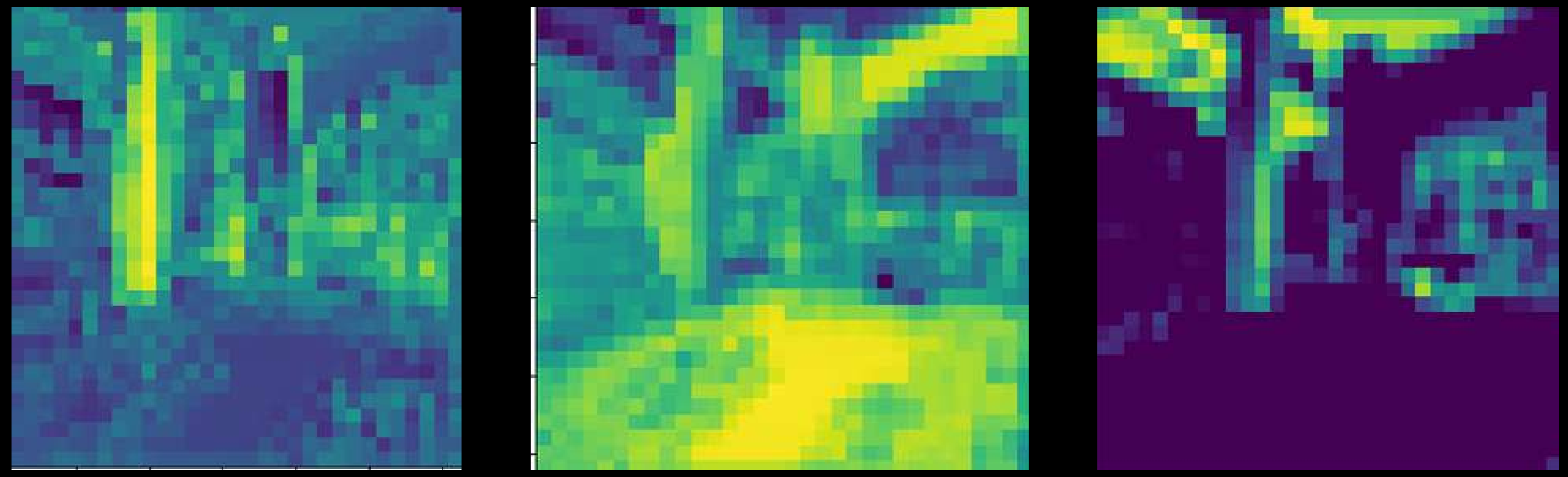


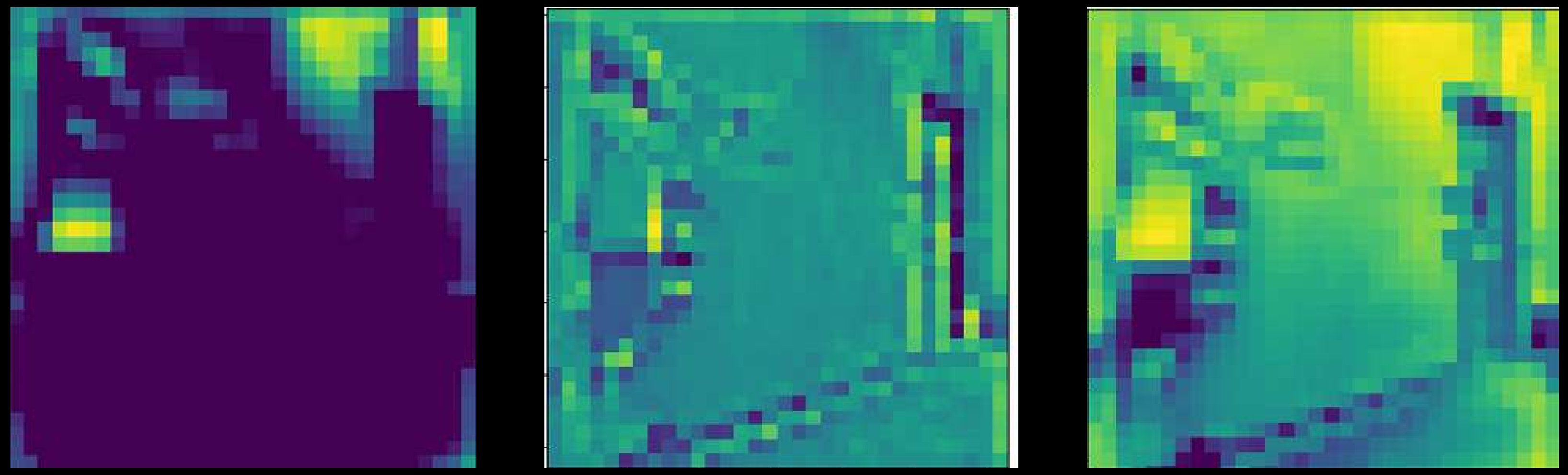
SIFT AND ORB



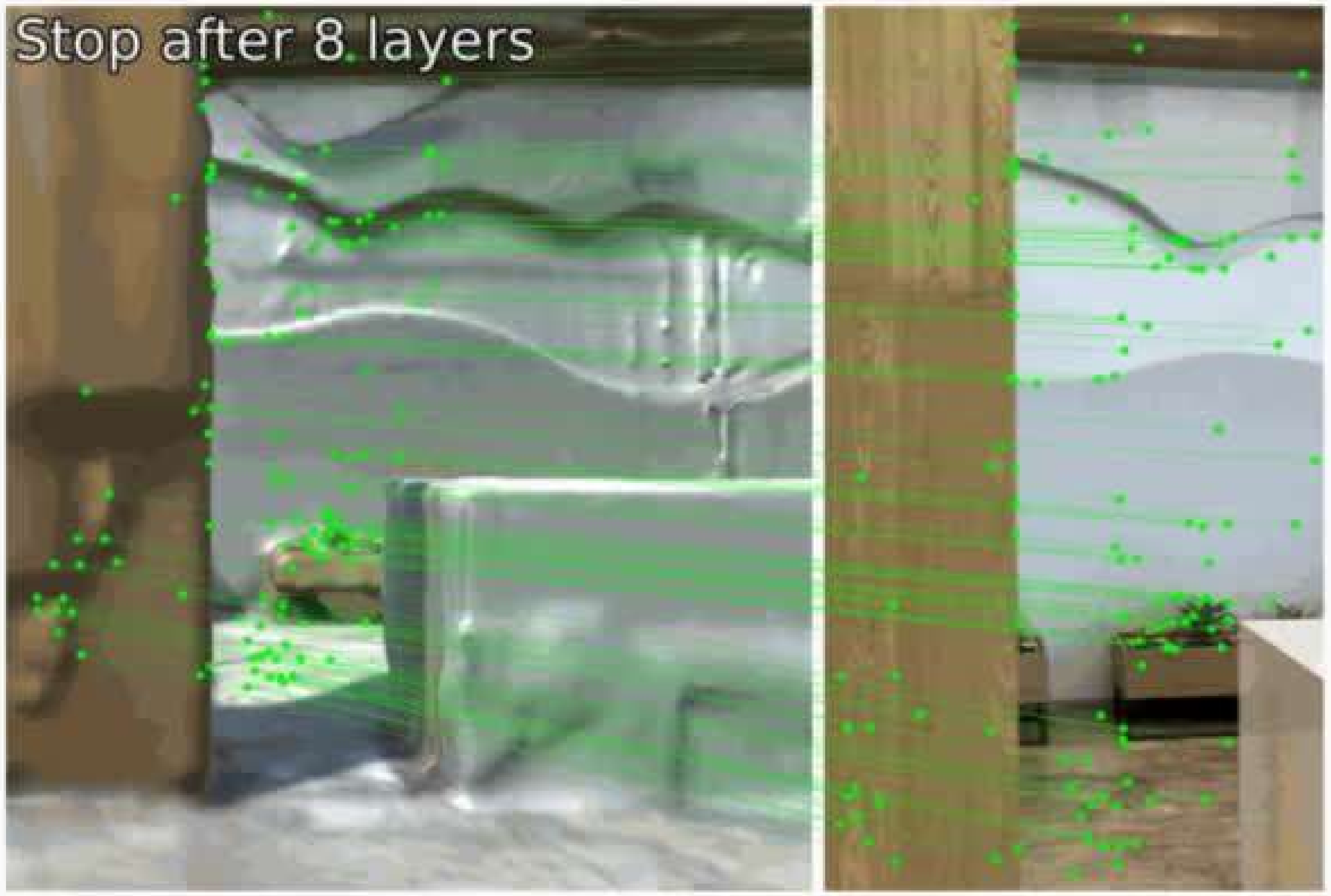


CNN AUTOENCODER



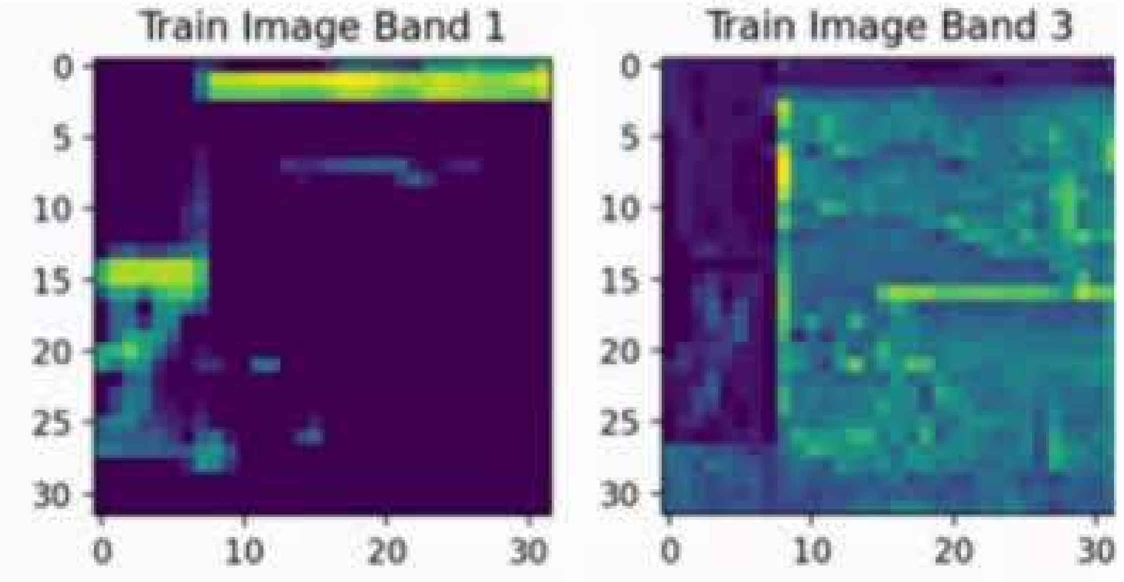


Synthetic vs Test Data : LightGlue/ORB

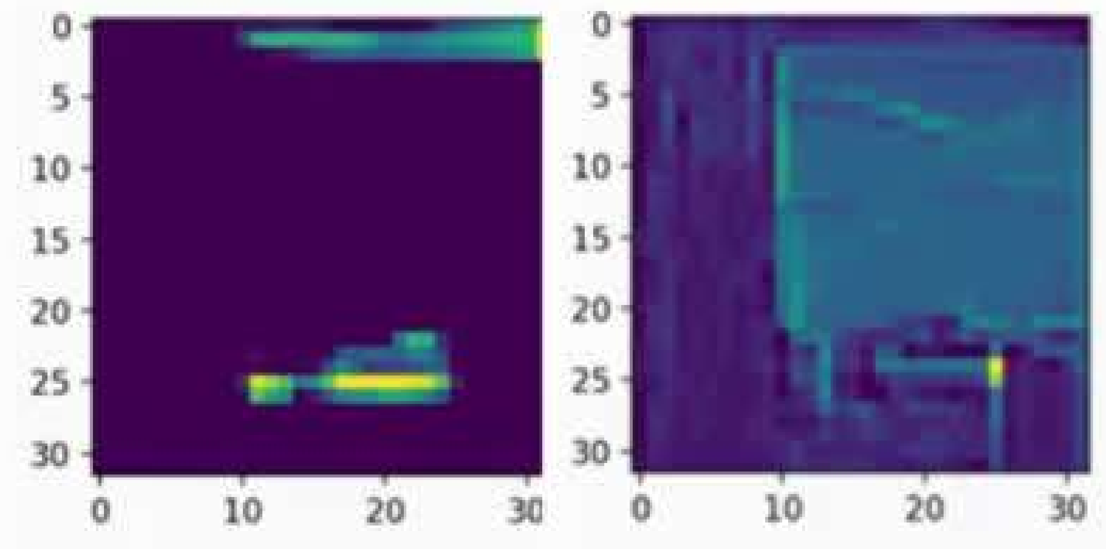


Frame 30 Distance: 2.608750152893151

Autoencoded Synthetic



Autoencoded Test Video



CHALLENGES IN THE MIDSEM

- Since we created a 3D model using LiDAR, there was a **drop in data quality**. This affects the accuracy of feature matching.

- Achieving **real-time performance** in applications like image matching might be a challenge we face.

THE LIGHTBULB MOMENT..

Our primary goal was **Navigation,** not **Pinpoint Positioning.**

In the real world, and especially in our makerspace, **precise X and Y coordinates were less critical** than efficiently guiding someone to their desired location.
Our makerspace, after all, is divided into **distinct sections.**

PIECES OF THE PUZZLE

Localization

First, we need to where in the makerspace the person is in.

Path Finding

Then, we need to create a path planning solution for the them.

Phone Friendly

They should be able to navigate using their mobile phones.

LITERATURE REVIEW

Indoor positioning and wayfinding systems: a survey

Computer Vision-Based Navigation and Wayfinding Systems:

- Focus on aiding visually impaired (VI) individuals.
- Example: ISANA system with Google Tango device and smart cane.
- Features advanced algorithms for map editing, object detection, and path planning.

Computer Vision-Based Positioning and Localization Systems:

- Utilizes computer vision for indoor localization and positioning.
- Emphasizes on scene recognition and detection of specific objects like doors.

Communication Technology Based Indoor Positioning and Wayfinding Systems:

- Involves signal measurement from devices like Wi-Fi access points and BLE beacons.
- Employs methods such as TOA, TDOA, and AOA for positioning.

Computer vision-based navigation and wayfinding systems

References	Beneficiary	Computer-vision solution	Path planning solution	Remarks /findings
Lie et al. [29]	People with VI	Google Tango VPS	Prism MST algorithm, A* algorithm	(+) Haptic feedback system provided safe navigation in noisy environments
Tian et al. [32]	People with VI	Canny edge detector, Tesseract and Omni page OCRs	Not available	(–) Path planning module is absent
Lee and Medioni [33]	People with VI	Corner-based motion estimator algorithm	SLAM and D* Lite algorithm	(–) Inconsistency in constructed maps
Garcia and Nahapetian [37]	People with VI	Canny edge detector and Hough line transform	Not available	(–) Detection failed for bulletin boards as well as low contrast wall pixels
Manlises et al. [38]	People with VI	Image subtraction, Histogram backpropagation	D* algorithm	(–) Low brightness and noise in indoor areas will affect the recognition and feedback systems, respectively
Bai et al. [39]	People with VI	Deep learning-based object recognition, scene parsing, Currency recognition functions	Vision-based slam	(+) Improved location awareness for the users
Athira et al. [49]	Customers of shopping mall	Gist descriptors	Not available	(–) Does not support navigation between floors
Pearson et al. [50]	Visitors of library	Bar code recognition	A* algorithm	(–) Misplaced books and books without barcodes can limit the system functionalities
Li et al. [51]	Normal people	SIFT descriptors	Self-adaptive dynamic-Bayesian network	(+) Scalability

Computer vision-based positioning, localizing and scene recognizing systems

References	Purpose	Solution	Performance /findings
Huang et al. [62]	Indoor positioning	3D signature of places for feature detection, Novel K-locations algorithm	(+) 90% of the exposed errors are within 25 cm and 2° for location and orientation respectively
Kawaji et al. [65]	Indoor positioning	PCA-SIFT features and locality sensitive hashing	(+) Running time reduced while comparing with the pure SIFT features-based system
Deniz et al. [67]	Localization using texts in boards and banners	Canny edge detector, Tesseract and ABBY fine reader OCRs	ABBY fine reader showed better recognition rate than Tesseract
Adorno et al. [78]	Floor detection method	Superpixel segmentation and Hough line transform	(+) Accuracy: 87.6% for the unstructured environment and 93% for the structured environment
Murillo et al. [61]	Personal localization in indoor areas	GIST and SURF-based feature detector, Extended Kalman filter monocular SLAM	(+) 82% correct localization
Xiao et al. [68]	Indoor positioning in large indoor areas	CNN and SIFT features	(+) Low cost, accuracy: less than 1 m
Chen et al. [71]	Indoor positioning	CNN and ORB features	(+) Average position error: less than 0.35 m
Kendall et al. [75]	Indoor and outdoor localization	CNN	(+) Robust to various lighting and motion blur scenarios
Bashiri et al. [79]	Indoor object recognition to assist people with VI	Transfer learning based on the CNN model (AlexNet)	(+) Accuracy: 98%
Jayakanth [81]	Indoor object recognition to assist people with VI	CNN and texture features	(+) Accuracy: 100%
Afif et al. [82]	Indoor object detection to assist people with VI	CNN	Mean average precision: 84.16%

Indoor–Outdoor Scene Classification with Residual Convolutional Neural Network

Objective:

- Develop a ResNet-18 based framework for classifying scenes as indoor or outdoor.
- Applicable to both color and depth images.

Significance:

- Useful in scene classification/retrieval and single-image depth estimation tasks.
- Addresses challenges in data complexity and scene analysis.

Methodology:

- Modified ResNet-18 architecture for efficient learning and classification.
- Trained and validated on four datasets: KITTI, Make3D, NYU Depth, Matterport.

Key Features:

- Handles different dimensional images and weather conditions.
- Employs pretrained weights from ImageNet for improved classification accuracy.

Results:

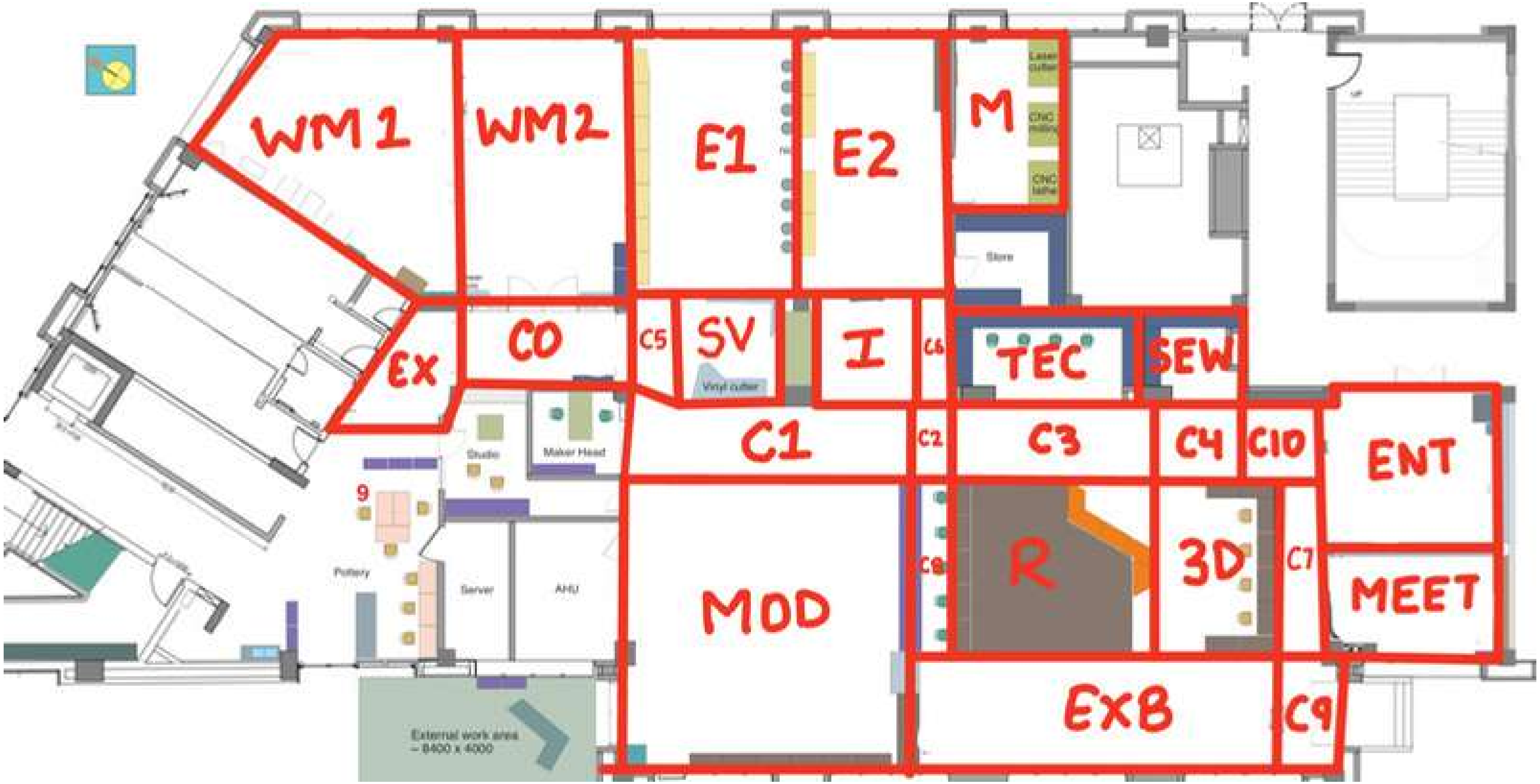
- High accuracy in discriminating indoor vs. outdoor scenes.
- Outperforms conventional and CNN-based approaches, especially in depth map analysis.

Conclusion:

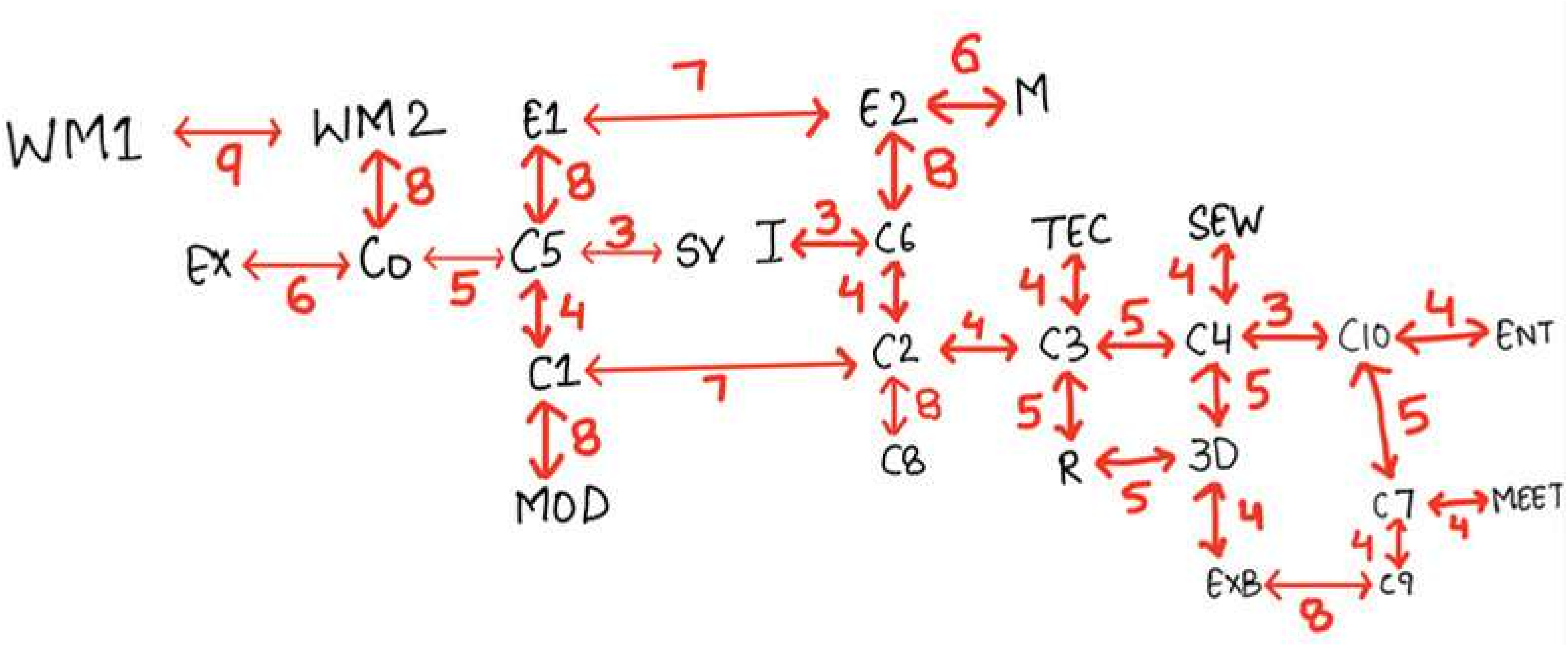
- Demonstrates potential for use in depth estimation and scene image analysis.
- Offers scope for future enhancements and applications.

MAKERSPACE LAYOUT

DIVISION IN ZONES

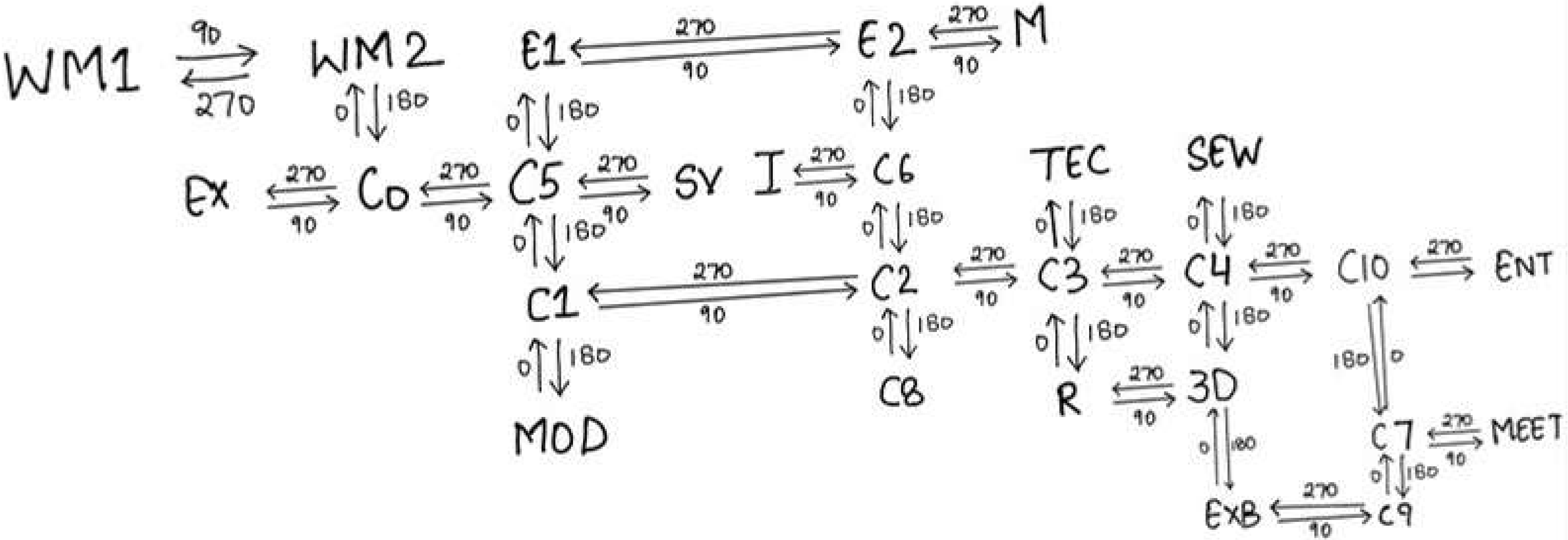


SCALED DISTANCE

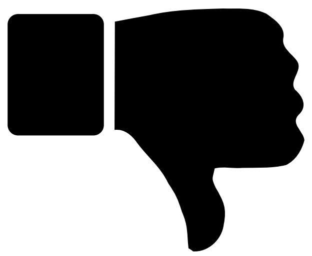


→ **Breadth First Search Algorithm**

LOCAL DIRECTION



TRIED WIRELESS FIDELITY!



- **Signal Interference**
  - WiFi signals were disrupted by physical obstructions and electronic interference.
- **Signal Strength Variability**
  - Environmental changes and device differences affected signal consistency.
- **Access Point Placement**
  - Required a dense network with strategically placed access points wasn’t met.
- **Unstable Plaksha WiFi**
  - Proved to be very unstable, failing to provide a reliable and consistent internet connection, which was a crucial requirement..

DATA COLLECTION

01 Divide Makerspace Zones

02 Take Videos of Each Zone

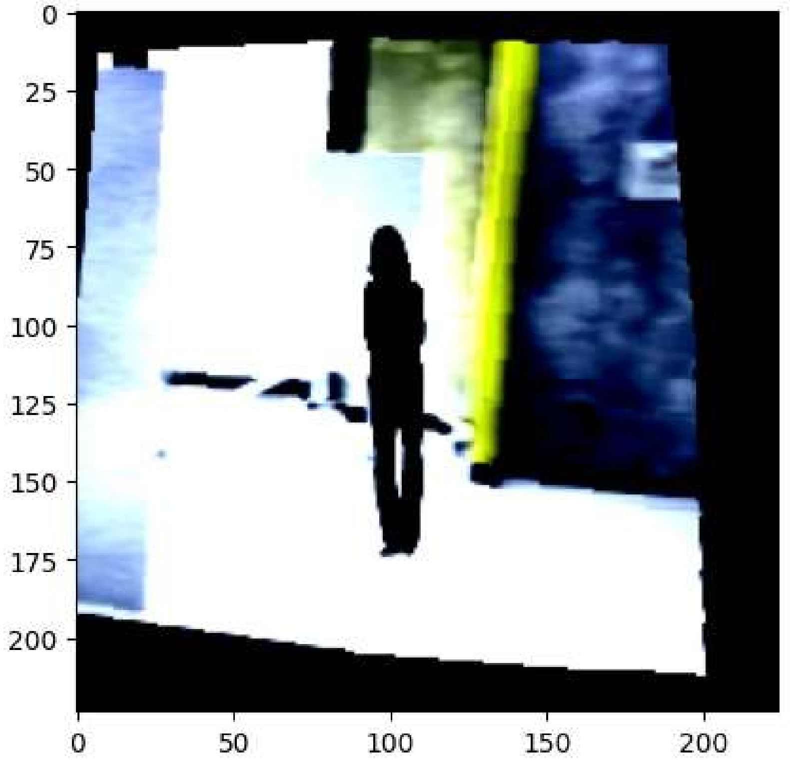
03 Extract Frames

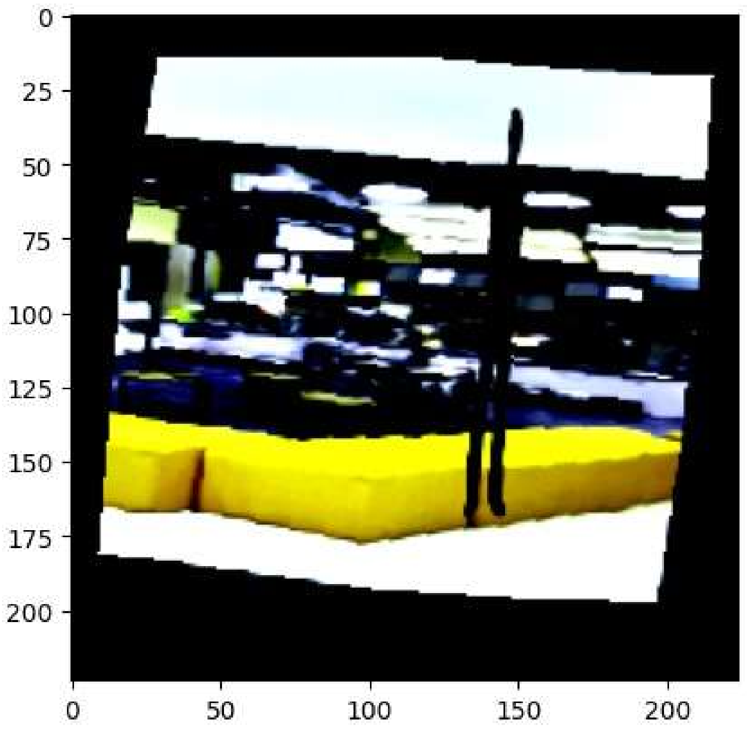
04 Eat,Sleep,Repeat

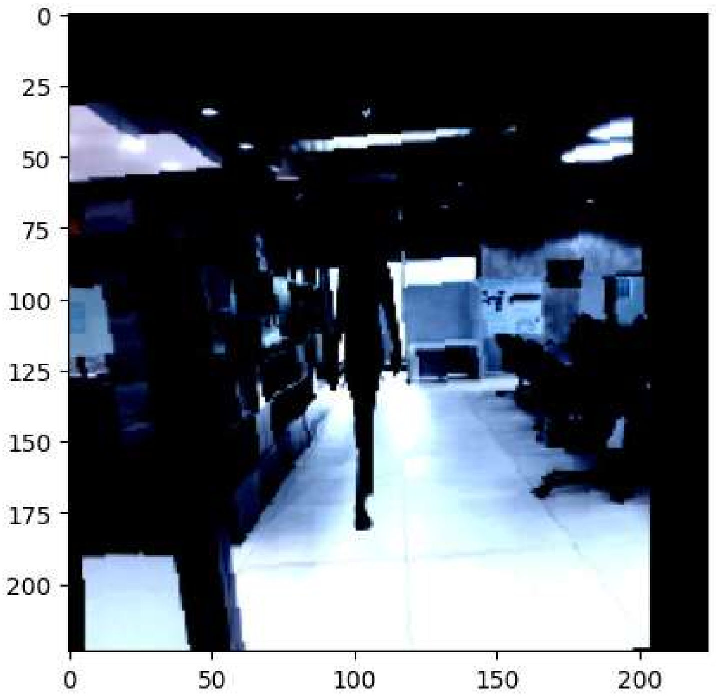
- **Varied Conditions:** Data was collected at **different times of day** to account for **varying lighting conditions and occupancy levels.**
- **Device Diversity:** We used both **Android and iOS smartphones** for capturing videos, ensuring our dataset reflects the diversity of devices.
- **Privacy:** The videos didn't include any people, asked them to move, ensuring their privacy.

TRANSFORMATIONS

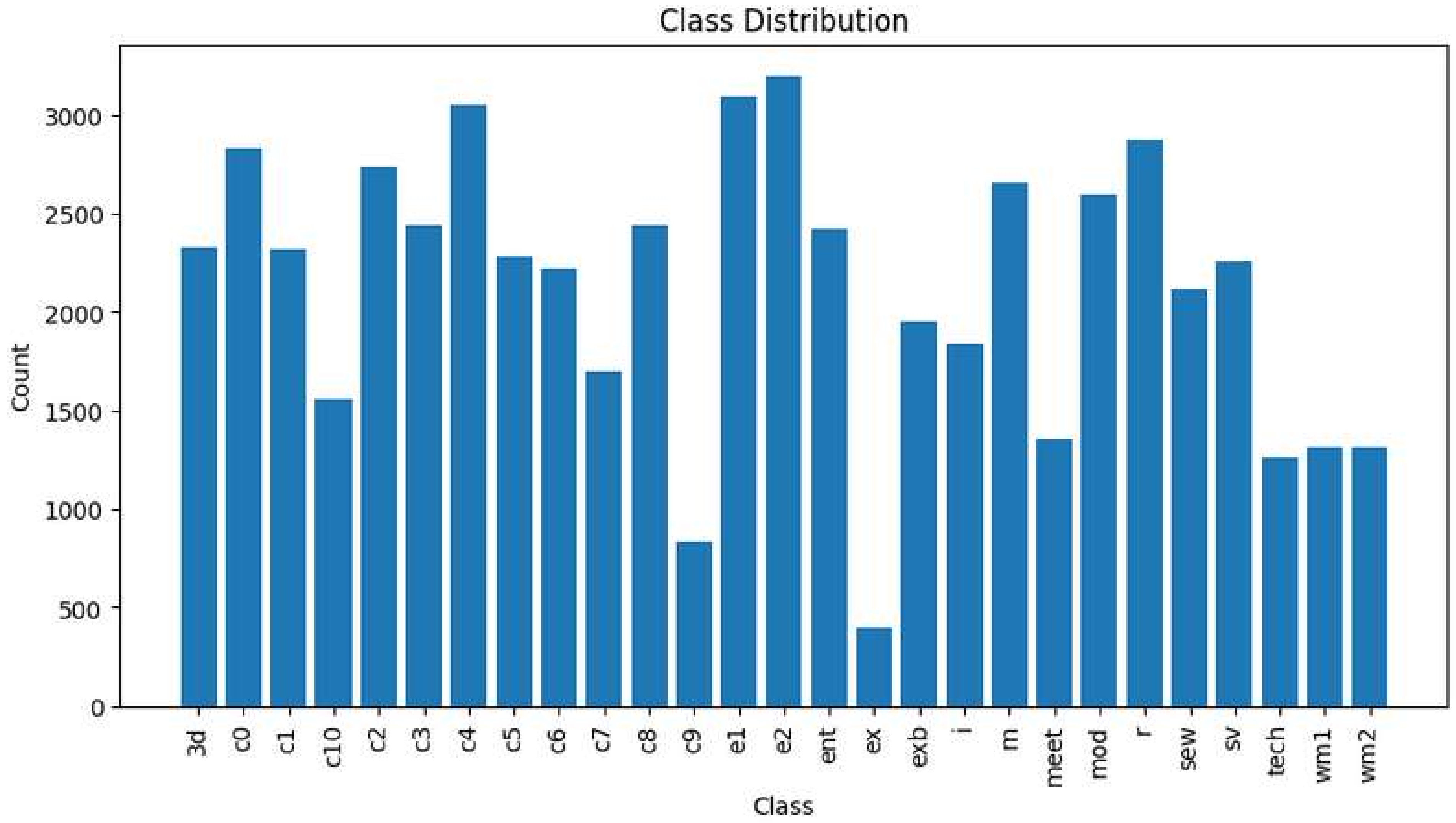
- **Geometric Transformations:** This included **rotations, scaling, and translations** to simulate different viewing angles and distances, mimicking how different users might see the space.
- **Color and Lighting Adjustments:** We altered **brightness, contrast, and saturation** to account for various lighting conditions that occur naturally throughout the day in the makerspace.
- **Synthetic Silhouette Placement for Occlusion:** To address the challenge of **human occlusion** – a common occurrence in a busy makerspace – we introduced **synthetic silhouettes into images.**







CLASS DISTRIBUTION

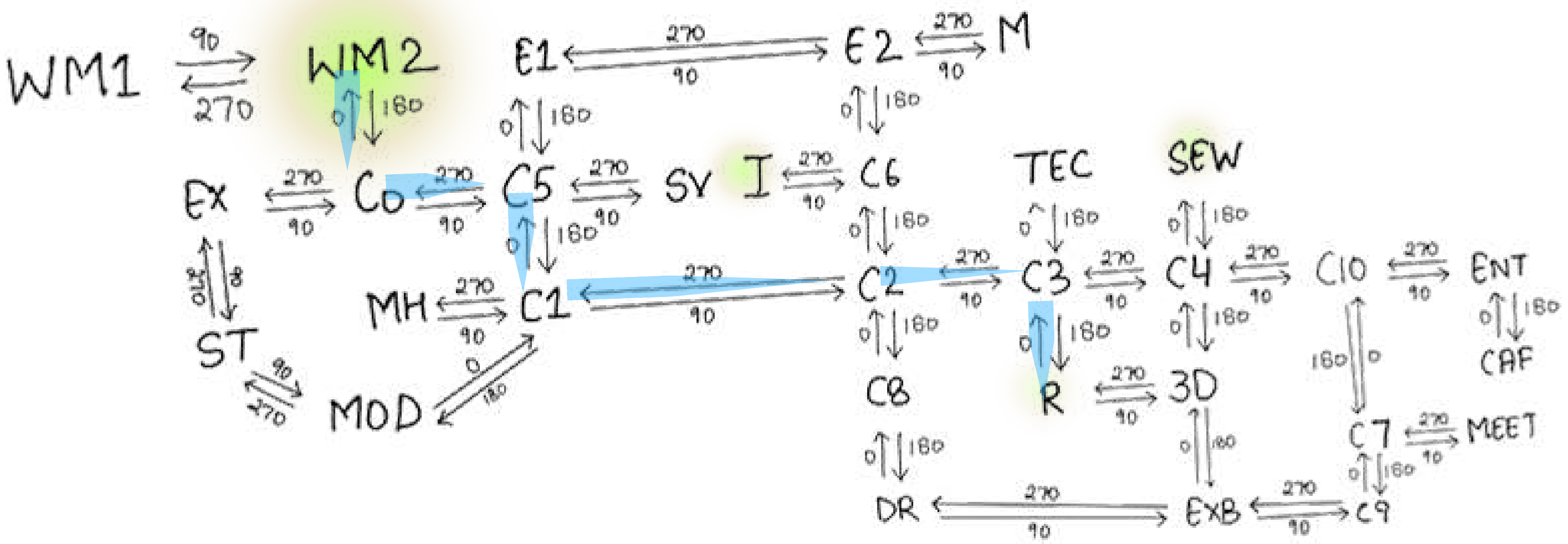


- Crude Data
- Classes were imbalanced
- Random oversampling, for classes with low frequency.

ML METHODOLOGY

makerspace divided in zones

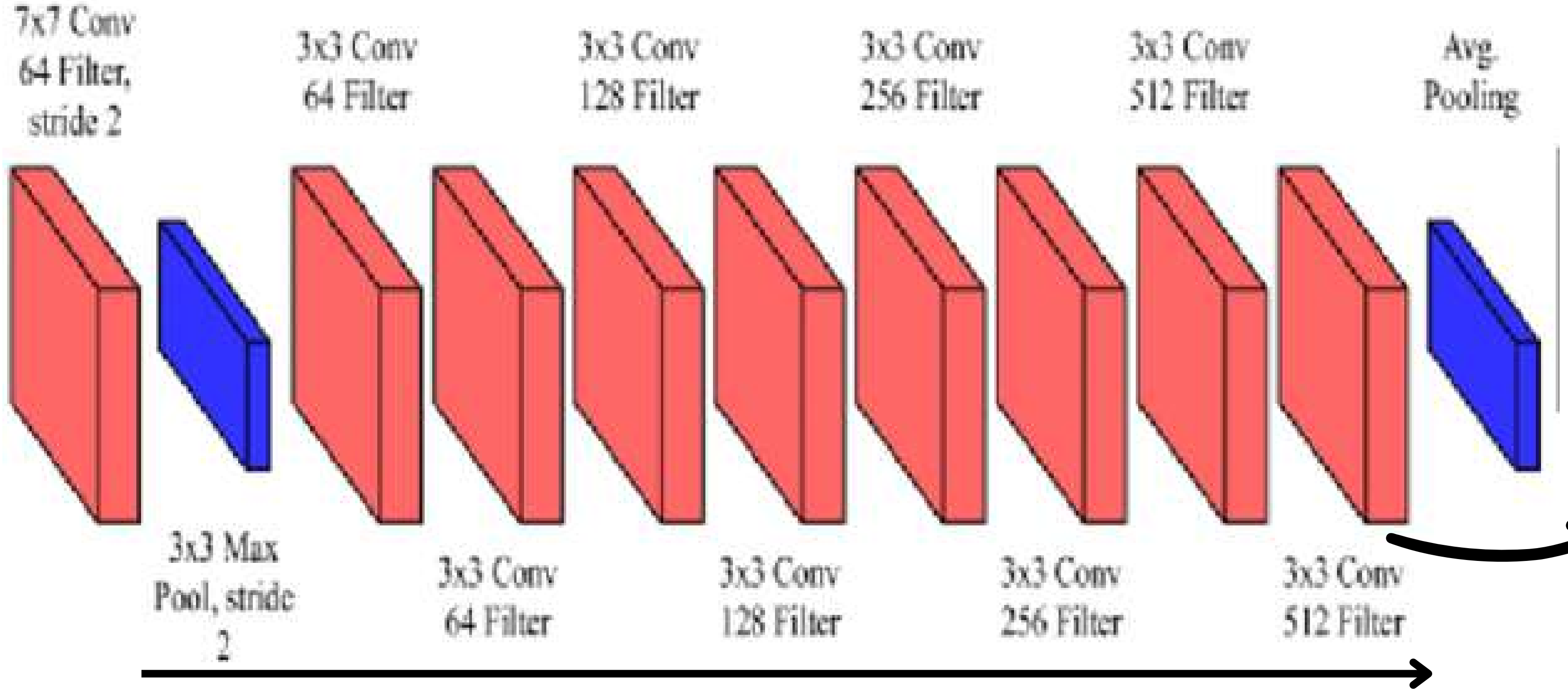
graphical representation & way finding



INDOOR NAVIGATION

modelling zone images

input image layer



Finetuned ResNet

illustration lab
meeting room
corridor 1
woodworks
robotics
3d printing
sewing lab
tech support

27 FC layers

OVERVIEW

01 ResNet 18

02 ResNet 50 (FineTuned)

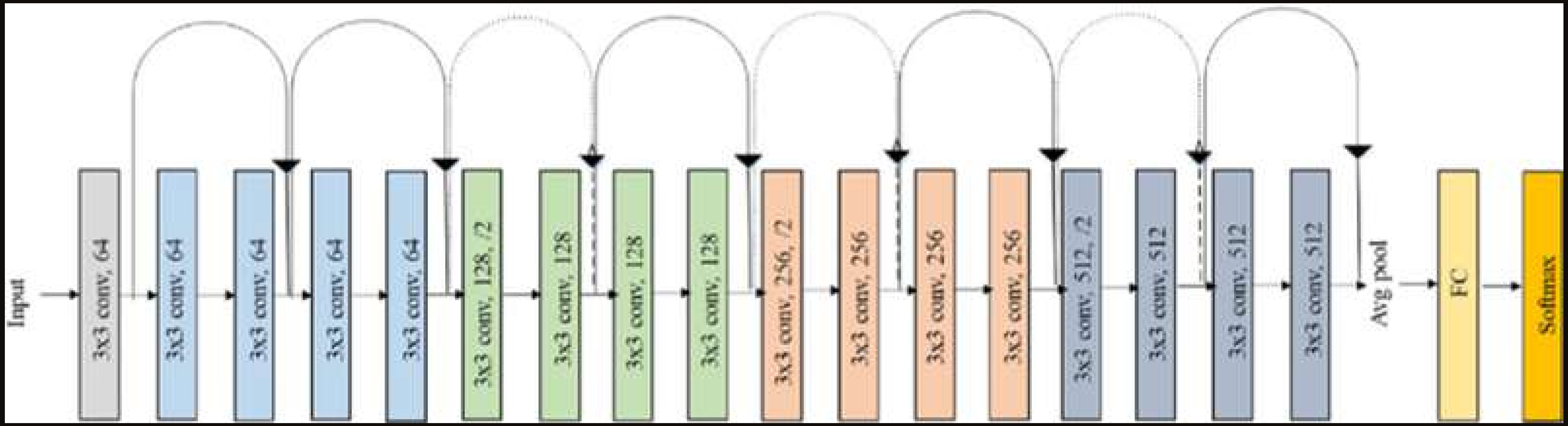
03 ResNet 101

04 Custom CNN

RESNET18

- **Pre-Trained Model**: ResNet18 is a **lighter model compared to ResNet50 and ResNet101.** It's faster but **less complex**, which could be beneficial for real-time applications or when computational resources are limited.

- **Final Layer Modification**: Adjusted to output predictions for the number of zones in the makerspace.

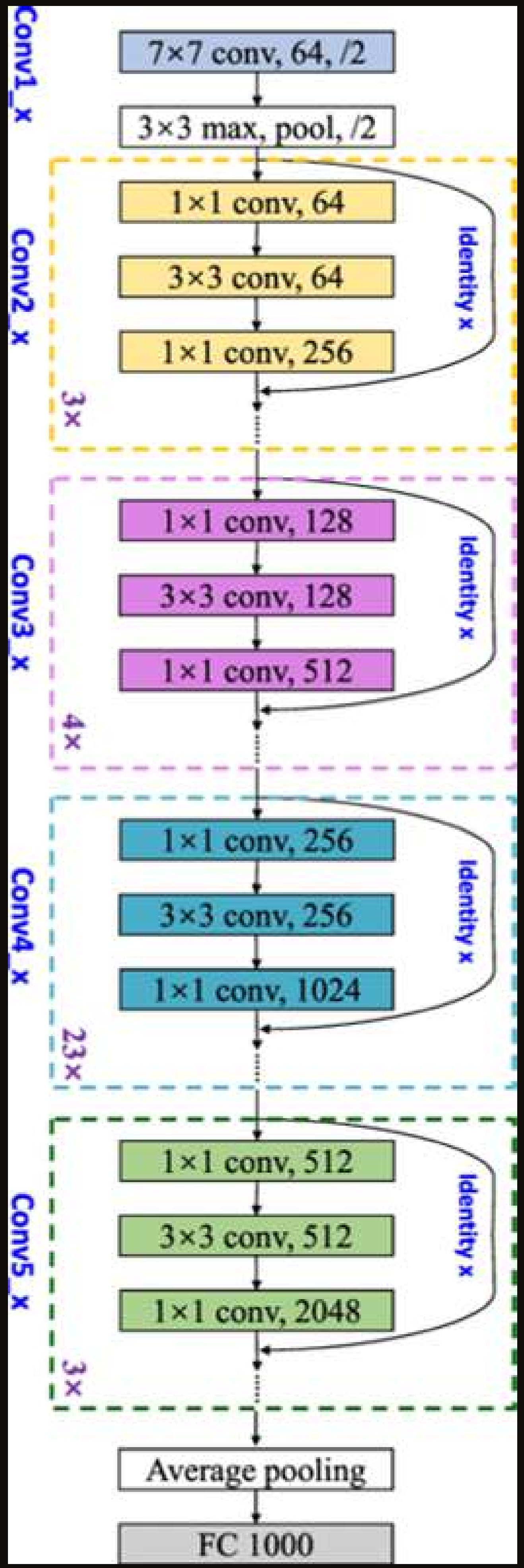


RESNET101

- **Pre-Trained Model:** Employs a pre-trained ResNet101 model, which offers a **deeper architecture than ResNet50.** The additional layers in ResNet101 enable it to capture **more complex features**, making it highly effective for intricate classification tasks.

- **Final Layer Adjustment:** The final fully connected layer of ResNet101 is modified to align with the number of zones in the makerspace. This customization ensures the model is precisely tailored to the specific classification requirements of the navigation system.

- **Optimization and Loss Function:** Utilizes the same Adam optimizer and CrossEntropyLoss function for ResNet18



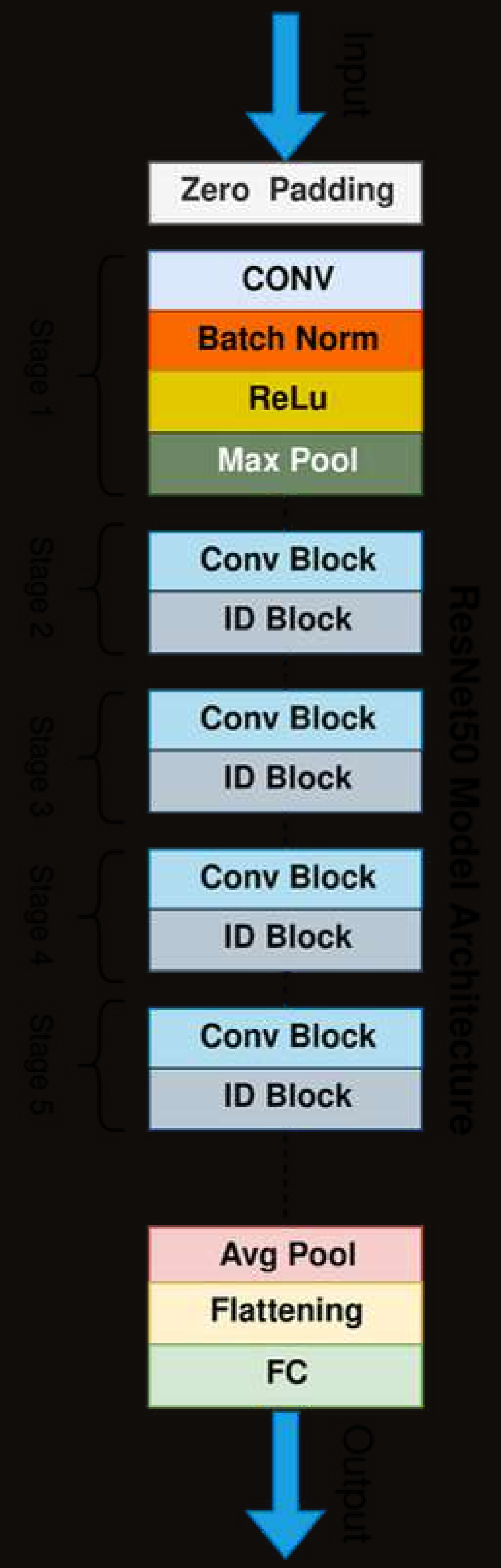
RESNET50

- **Pre-Trained Model:** Utilizes a pre-trained ResNet50 model, capitalizing on features learned from a comprehensive dataset like ImageNet.

- **Fine-Tuning Layers:** Specifically **unfreezes 'layer3' and 'layer4'** of the ResNet50 model for **fine-tuning**. This allows these layers to update during training, adapting to the unique features of the makerspace environment.

- **Final Layer Customization:** Modifies the final fully connected layer (fc) to output predictions for the number of zones (27).

- **Optimization and Loss Function:** Employs the **Adam optimizer** for efficient training, coupled with the **CrossEntropyLoss** function.



CUSTOM CNN

Purpose: Designed to accurately determine indoor positions using image data.

Architecture Overview:

- Layered Convolutional Design: Three-tier structure, refining image features progressively (64, 128, and 256 filters).
- Max Pooling for Spatial Hierarchy: Condenses information after each convolutional layer, emphasizing salient features and reducing computational load.
- Efficient Activation and Flattening: ReLU activation for non-linearity and tensor flattening for seamless transition from spatial feature extraction to classification.
- Efficient in extracting and processing features critical for position determination.
- Scalable and adjustable for different indoor environments and image datasets.

```python
class CustomLabZoneModel(nn.Module):
    def __init__(self, num_classes=37):
        super(CustomLabZoneModel, self).__init__()
        # Define the layers
        self.conv1 = nn.Conv2d(in_channels=3, out_channels=64, kernel_size=3, padding=1)
        self.conv2 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, padding=1)
        self.conv3 = nn.Conv2d(in_channels=128, out_channels=256, kernel_size=3, padding=1)
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        self.fc1 = nn.Linear(256 * 28 * 28, 1024)  # Adjust the input features size accordingly
        self.fc2 = nn.Linear(1024, num_classes)

    def forward(self, x):
        # Define the forward pass
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = self.pool(F.relu(self.conv3(x)))
        x = x.view(-1, 256 * 28 * 28)  # Flatten the tensor
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

ADDITIONAL FUNCTIONS

- **Moving Average:** Using the last 30 frames of the 50 stored to identify where we are or else asking the user to look around. Using the threshold number of images to set a cut off for majority images.

```python
def find_shortest_path_dijkstra(graph, start_node, end_node):
    try:
        # Compute the shortest path using Dijkstra's algorithm
        path = nx.dijkstra_path(graph, source=start_node, target=end_node, weight='weight')
        for i in range(len(path)-1):
            edge = (path[i], path[i+1])
            direction = graph.edges[edge]['label']
            distance = graph.edges[edge]['weight']
        return path
    except (nx.NetworkXNoPath, KeyError):
        # Return an empty list if no path exists or if the nodes are not in the graph
        return []
```

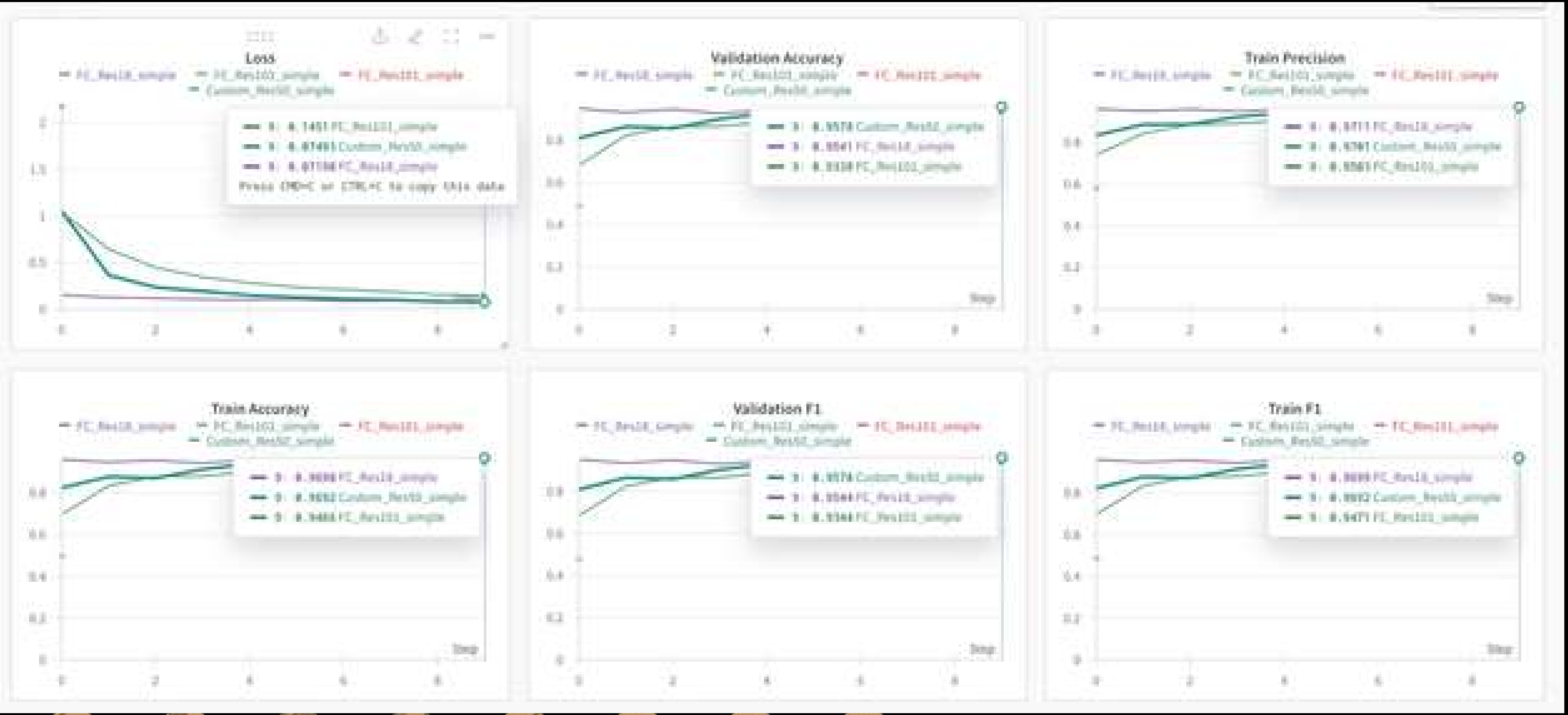
- **BFS in a weighted graph:** Using distance based weighted graph to represent the search space of tiles and then finding the cheapest path and guiding user accordingly.

```python
@app.post("/navigate")
async def navigate(request: NavigateRequest):
    try:
        print("navigate", request.session_id, request.end, request.alpha)
        # Fetch final prediction for the session
        final_query = predictions.select().where(predictions.c.session_id == request.session_id).order_by(-predictions

        final_row = await database.fetch_one(final_query)
        current_position = final_row["final_prediction"]
        if current_position is None:
            raise HTTPException(status_code=400, detail="No final prediction available")

        path, directions = graph.navigate(current_position, request.end, request.alpha)
```

CHALLENGES <>

1. **Synthetic Data Limitations**: Experienced poor performance due to the synthetic nature of the training data, which presented significant outliers. This issue particularly impacted edge/gradient-based methods and feature matching.
2. **Regression Training Difficulties**: The use of Convolutional Neural Networks (CNNs) for regression tasks showed low effectiveness, indicating a potential mismatch between the model architecture and the task.
3. **Cost Implications of Autoencoders**: Employing autoencoders for comparing smartphone-captured images with synthetic images proved costly. This approach, based on cosine similarity, required extensive image comparisons, escalating computational demands.
4. **Data Collection Challenges:** Faced hurdles in data acquisition, including the high volume of images needed for effective training and the lack of readily available 3D cameras for more accurate data capture.
5. **Managemant of overwhelming amount of data and training multiple models around it.**

PERFORMANCE METRICS



1 epoch time : 1.5-3 hours

BREAKDOWN (RESULTS)

Validation Accuracy

ResNet18: 95.4%

ResNet50: 95.8%

ResNet101: 93.4%

Train Accuracy

ResNet18: 96.7%

ResNet50: 97%

ResNet101: 94.7%

Loss

ResNet18: 0.72

ResNet101: 0.1451

ResNet50: 0.75

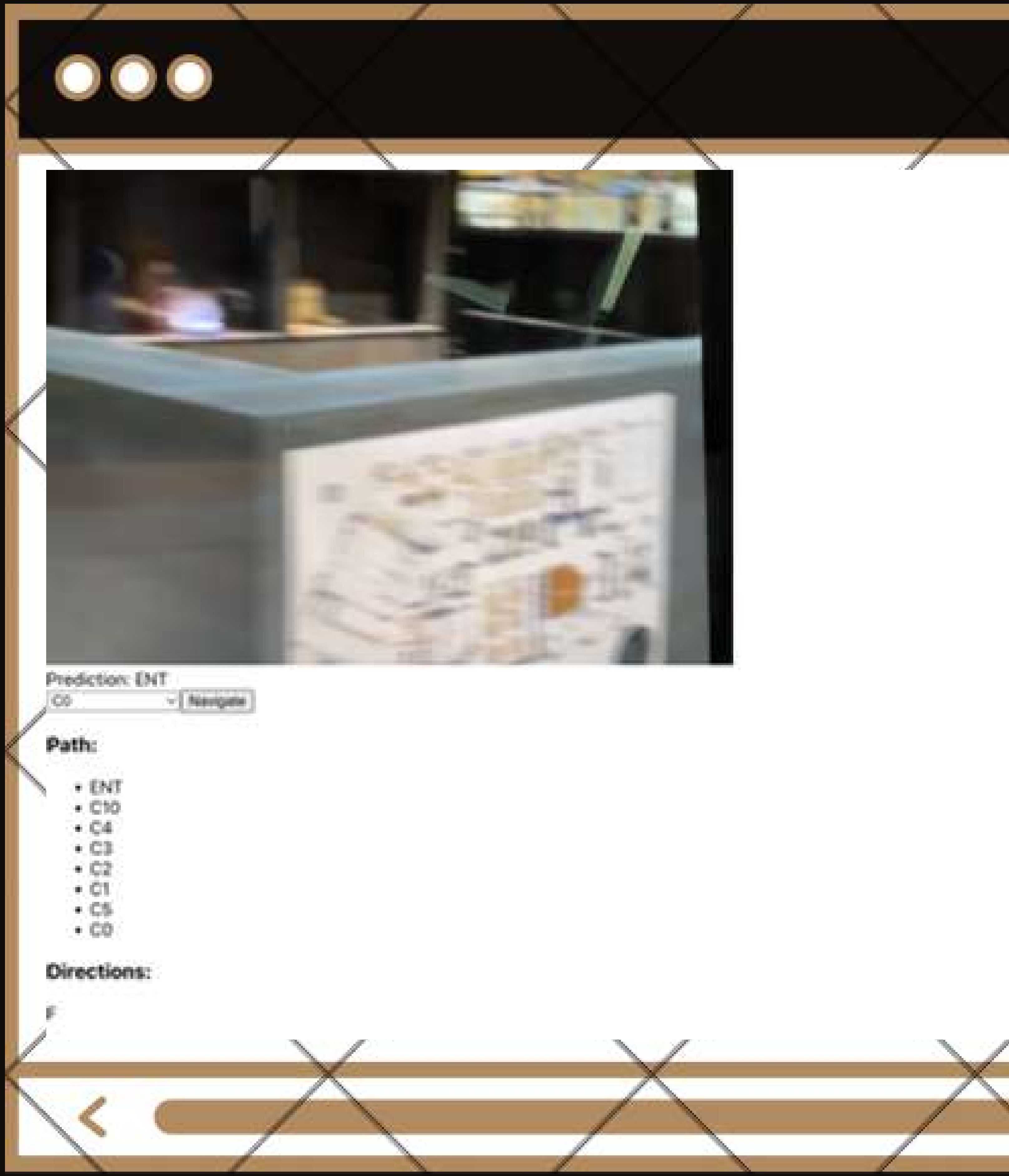
DEPLOYMENT

Frontend

Using React powered website for accessibility. Anyone can just scan a code at makerspace and reach the website to experience the whole process

Backend

A simple Fast API based server hosted locally on a plaksha server, using GPU to serve the model. Incorparates session for each new user to help the user navigate using Moving average and graph based way finding.



Team:
Aman, Anshika, Anushka

THANK YOU!