CHAPTER-1
INTRODUCTION

1.1 PROBLEM STATEMENT:

Student attendance is a critical aspect of the educational process, impacting both student learning and institutional resources. Traditional manual attendance tracking methods are time-consuming, prone to errors, and can be inefficient. This project addresses the challenge of accurately and efficiently determining student attendance in educational settings by developing a computer vision-based system for automatically counting students in images. The system aims to alleviate the burden on educators by automating a tedious task and providing a more reliable and objective method for recording student attendance. This can lead to improved data accuracy, enhanced resource allocation, and ultimately, a more effective and efficient learning environment for students. Specifically, the project focuses on developing a deep learning model that can effectively analyse images of classrooms or other educational settings to accurately estimate the number of students present. The model will be trained on a dataset of images with annotated student counts, enabling it to learn the visual patterns and features associated with different student populations.

1.2 MOTIVATION:

In today's data-driven educational landscape, accurate and timely student attendance data is crucial for effective teaching and learning. Traditional methods of tracking attendance, such as manual roll calls and sign-in sheets, while widely used, present significant challenges.

- **Time-Consuming and Labour-Intensive:** Manual methods require significant time and effort from educators, diverting their attention from core teaching responsibilities and lesson planning.
- **Prone to Human Error:** Manual data entry and record-keeping are susceptible to human errors, leading to inaccuracies in attendance data. These errors can have unintended consequences, impacting student funding, resource allocation, and the overall assessment of student performance.
- **Lack of Real-time Insights:** Manual systems often provide delayed or incomplete data, hindering timely interventions for students experiencing attendance issues.

- **Limited Data Analysis:** Traditional methods offer limited capabilities for analyzing attendance data to identify trends, identify at-risk students, and inform data-driven interventions.

To address these challenges, this project explores the potential of automated student attendance tracking systems. These systems leverage advanced technologies such as deep learning techniques to efficiently and accurately capture and analyze student attendance data.

By automating attendance tracking, educational institutions can:

- **Improve Accuracy and Reliability:** Minimize human error and ensure the accuracy and reliability of attendance data.
- **Increase Efficiency:** Free up educators' time by automating a time-consuming task, allowing them to focus on teaching and student support.
- **Gain Real-time Insights:** Provide real-time or near real-time attendance data, enabling timely interventions for students experiencing attendance issues.
- **Enable Data-Driven Decision Making:** Facilitate data-driven analysis of attendance patterns, identify at-risk students, and inform targeted interventions to improve student engagement and success.
- **Enhance Resource Allocation:** Optimize resource allocation based on accurate attendance data, ensuring that resources are effectively utilized to support student learning.

By embracing automated attendance tracking systems, educational institutions can leverage the power of technology to improve the efficiency and accuracy of attendance tracking, enhance student support, and ultimately create a more effective and data-driven learning environment for all students.

1.3 SCOPE:

This project focuses on developing and evaluating a deep learning-based system for automatically counting students in static images. The system will be designed to analyze images of classrooms or other educational settings and estimate the number of students present.

The scope of this project includes:

- **Data Collection and Preparation:** Gathering and preparing a suitable dataset of images with accurate student counts.
- **Model Development:** Designing and training a deep learning model, such as a Convolutional Neural Network (CNN), to effectively analyze the images and predict student counts.

- **Model Evaluation:** Evaluating the performance of the trained model using appropriate metrics, such as accuracy and precision.
- **System Implementation:** Developing a user-friendly interface for interacting with the system, allowing users to easily upload images and receive predictions. The project will not address the following:
  - Counting students in real-time video streams.
  - Handling complex scenarios such as occlusions, crowded scenes, or varying lighting conditions.
  - Integrating the system with existing school management systems.

1.4 OUTLINE:

Traditional methods of tracking student attendance in educational institutions, such as manual roll calls or sign-in sheets, present significant challenges in today's fast-paced and data-driven environment. These methods often fall short in effectively and accurately capturing student attendance, leading to various inefficiencies and potential inaccuracies.

- **Time-Consuming and Labor-Intensive:** Manual attendance tracking demands considerable time and effort from educators. Teachers must meticulously record student presence, often during valuable instructional time, which can disrupt the learning process and detract from their core responsibilities of teaching and lesson planning. This manual process can be particularly burdensome in larger classes, where recording attendance for numerous students can be time-consuming and prone to errors.

- **Prone to Human Error:** Human error is an inherent risk in manual attendance tracking. Teachers may inadvertently miss marking a student present, overlook absences, or make transcription errors when transferring data to digital records. These inaccuracies can have significant consequences, impacting student funding, resource allocation, and the overall assessment of student engagement and academic progress.

- **Lack of Efficiency and Scalability:** Manual systems struggle to keep pace with the growing demands of modern education. In large institutions with multiple classes and high student enrollment, managing and analyzing attendance data manually can become overwhelming and inefficient. These systems often lack the scalability and flexibility to accommodate the increasing volume of data and the evolving needs of educational institutions.

- **Data Integrity and Security Concerns:** Manual systems often rely on paper-based records, which are susceptible to loss, damage, or unauthorized access. This raises concerns about data integrity and security, potentially compromising student privacy and the accuracy of attendance records.

These limitations of traditional methods necessitate the development of more efficient, accurate, and reliable solutions for tracking student attendance. By leveraging technology and automation, educational institutions can overcome these challenges and improve the overall effectiveness of their attendance tracking systems.

This revised problem statement provides a more detailed and comprehensive analysis of the challenges associated with traditional student attendance tracking methods, emphasizing the specific issues and their potential impact on educational institutions.

CHAPTER-2
LITERATURE SURVEY

- Manual attendance methods like roll-calling and sign-in sheets are error-prone, disrupt classroom flow, and waste teaching time.
- Biometric systems are costly and require direct interaction, making them less suitable for real-time environments.
- The proposed system leverages deep learning for non-intrusive, scalable student attendance tracking.
- Cholakkal et al. (2019) introduced density map-based object counting using image-level supervision.
- Their method predicts global object counts and spatial distributions without instance-level annotations.
- Psychological studies on subitizing inspired their reduced annotation approach, focusing on 1-4 objects.
- The loss function integrates spatial and global terms to improve count accuracy and spatial predictions.
- Performance evaluations on PASCAL VOC and COCO datasets showed superior results to traditional methods.
- This density map approach aligns with attendance systems by enabling accurate, real-time student counting.
- Image-level supervision eliminates intrusive interactions, meeting the needs of non-intrusive solutions.
- Transformer models like DETR enhance detection by capturing long-range dependencies and eliminating region proposals.
- DETR's suitability for crowded settings makes it ideal for handling overlapping students in classrooms.
- Challenges include handling occlusions in dense classrooms, requiring spatio-temporal analysis from videos.
- Real-time processing necessitates optimization for hardware constraints typical in classrooms.
- Privacy concerns must be addressed to ensure ethical deployment and acceptance in educational settings.

- Integrating transformers with density maps can achieve accurate detection and counting in real-time.
- Such advancements reduce the need for manual annotations and improve scalability across institutions.
- Future work should focus on combining motion information to overcome occlusion challenges.
- The proposed solution enhances efficiency by streamlining attendance, reducing errors, and saving time.
- Automated attendance systems based on these methods provide a robust foundation for educational improvements.

CHAPTER-3
PROPOSED WORK, ARCHITECTURE, TECHNOLOGY STACK & IMPLEMENTATION DETAILS

3.1 PROPOSED WORK:

The proposed system aims to address the limitations of traditional attendance tracking methods by leveraging computer vision and deep learning. The core idea is to develop a system that can automatically count students in classroom images.

Key Components:

1. Image Acquisition: The system will utilize images captured by cameras installed in classrooms. These images can be captured periodically throughout the class or at specific intervals.
2. Object Detection and Counting:
    - The system will employ a deep learning model, specifically a Convolutional Neural Network (CNN), trained on a large dataset of classroom images with annotated student counts.
    - The trained model will be used to detect and localize individual students within the classroom images.
    - The system will then count the number of detected students, providing an automated and objective assessment of class attendance.
3. Data Storage and Analysis:
    - The system will store attendance data in a secure and organized manner, enabling easy access and analysis.
    - The stored data can be used to generate reports, identify attendance trends, and identify students who may be at risk of absenteeism.
4. User Interface:
    - A user-friendly interface will be developed to allow educators to easily access and review attendance data, generate reports, and configure system settings.

Expected Benefits:

- Improved Accuracy: Automated student counting can significantly reduce human error and improve the accuracy of attendance records.
- Increased Efficiency: The system will automate a time-consuming task, freeing up educators to focus on teaching and other important responsibilities.
- Real-time Data: The system can provide real-time or near real-time attendance data, enabling timely interventions for students with excessive absences.
- Data-Driven Insights: The system will enable data-driven analysis of attendance patterns, allowing educators to identify potential issues and implement targeted interventions.

This proposed system leverages the power of computer vision and deep learning to address the limitations of traditional attendance tracking methods, offering a more efficient, accurate, and data-driven solution for educational institutions.

3.2 TECHNOLOGY STACK:

S.no	Technology Stack	Version
1.	HTML	HTML5
2.	CSS	CSS3
3.	JavaScript	JavaScript ES7
4.	React	React 18.3
5.	Flask	Flask 3.1.0

Table 3.1 Tech Stack

3.2.1 Front End Technologies:

For building our project, we primarily used HTML, CSS, JavaScript, and the React library to craft a seamless and interactive user interface.

HTML (HTML5): HTML5 is the latest version of the HyperText Markup Language, which structures web content.

Key Features:

- Semantic Elements: HTML5 introduces elements like <header>, <footer>, and <article>, which improve code readability and accessibility.
- Multimedia Support: Built-in support for audio and video without needing plugins.
- Improved Forms: New input types and attributes enhance form usability, such as email and date.

CSS (CSS3): CSS3 is used for styling the web pages, ensuring they are visually appealing and responsive.

Key Features:

- Media Queries: Enables responsive design, ensuring layouts adjust for different screen sizes.
- Transitions and Animations: Allows for smooth effects and animations.
- Grid and Flexbox: Simplifies the design of complex layouts.

JavaScript (ES7): JavaScript ES7 (ECMAScript 2016) is a scripting language that makes web pages interactive.

Key Features:

- Array Methods: Includes methods like Array.prototype.includes() for better data handling.
- Exponentiation Operator: Simplifies mathematical calculations.

React (React 18.3): React is a popular frontend library developed by Jordan Walke and maintained by Meta. It simplifies building dynamic and responsive UIs.

Key Features:

- Component-Based Architecture: React uses encapsulated, reusable components that manage their state.
- Virtual DOM: Optimizes rendering by updating only the necessary parts of the real DOM.

- Declarative Syntax: Makes UI creation intuitive by describing how the UI should appear in different states.
- Flexibility: Integrates well with other libraries and frameworks.
- JSX Syntax: Combines JavaScript and HTML, enhancing code readability and simplicity.
- React, coupled with these technologies, powers the project's frontend, ensuring a smooth and dynamic user experience.

3.2.2 Back End Technologies:

For server-side operations, we have used Flask, a lightweight web framework for Python, to build and manage the backend logic of our project.

Flask (3.1.0): Flask is a minimalistic yet powerful framework that simplifies backend development. It is known for its simplicity and flexibility, making it a popular choice for both small projects and complex applications.

Key Features:

- Microframework Design: Flask provides essential tools and libraries without imposing strict guidelines, giving developers complete control over project architecture.
- Built-in Development Server: Comes with a lightweight server for testing and debugging.
- URL Routing: Enables clean and readable URLs, enhancing navigation and user experience.
- Template Engine (Jinja2): Flask uses Jinja2 to dynamically generate HTML pages with server-side data.
- Integration with WSGI: Supports WSGI, allowing it to handle multiple requests efficiently.

3.3 IMPLEMENTATION:

3.3.1 Architecture code:

```
class StudentCounter(nn.Module):
    def _init_(self, backbone='resnet50', pretrained=True):
        super(StudentCounter, self)._init_()

        # Load pretrained backbone
        if backbone == 'resnet50':
            self.backbone = models.resnet50(pretrained=pretrained)
            feature_dim = self.backbone.fc.in_features
            self.backbone.fc = nn.Identity()  # Remove classification layer

        # Shared features
        self.shared_layers = nn.Sequential(
            nn.Linear(feature_dim, 512),
            nn.ReLU(),
            nn.Dropout(0.3),
            nn.Linear(512, 256),
            nn.ReLU(),
            nn.Dropout(0.2)
        )

        # Branch for total count
        self.total_branch = nn.Sequential(
            nn.Linear(256, 128),
            nn.ReLU(),
            nn.Dropout(0.1),
            nn.Linear(128, 1)
        )

        # Branch for intersection count
        self.intersection_branch = nn.Sequential(
            nn.Linear(256, 128),
            nn.ReLU(),
            nn.Dropout(0.1),
            nn.Linear(128, 1)
        )

        # Learnable parameters for loss weighting
        self.total_weight = nn.Parameter(torch.tensor([1.0]))
        self.intersection_weight = nn.Parameter(torch.tensor([1.0]))

    def forward(self, x):
        # Extract features
        features = self.backbone(x)
        shared = self.shared_layers(features)

        # Get predictions
        total_count = self.total_branch(shared)
        intersection_count = self.intersection_branch(shared)

        # Ensure counts are positive using softplus
        total_count = F.softplus(total_count)
        intersection_count = F.softplus(intersection_count)

        return total_count, intersection_count
```

3.3.2 Model Training code:

```
def train_model(model, train_loader, val_loader, device, num_epochs=100):
    optimizer = optim.AdamW(model.parameters(), lr=0.001)
    scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, mode='min', patience=5)
    criterion = CustomLoss()

    best_val_loss = float('inf')
    best_model_state = None
    for epoch in range(num_epochs):
        # Training phase
        model.train()
        train_loss = 0.0
        train_samples = 0

        for batch_idx, batch in enumerate(train_loader):
            if (batch_idx + 1) % 10 == 0:
                print(f'Epoch {epoch+1}/{num_epochs} - Batch
{batch_idx+1}/{len(train_loader)}', end='\r')

            images = batch['image'].to(device)
            total_counts = batch['total_count'].to(device)
            intersect_counts = batch['intersect_count'].to(device)
            optimizer.zero_grad()

            total_pred, intersect_pred = model(images)

            loss = criterion(total_pred, inter   sect_pred,
                        total_counts, intersect_counts,
                        model.total_weight, model.intersection_weight)

            loss.backward()

            # Gradient clipping
            torch.nn.utils.clip_grad_norm_(model.parameters(), max_norm=1.0)

            optimizer.step()

            train_loss += loss.item() * images.size(0)
            train_samples += images.size(0)

        avg_train_loss = train_loss / train_samples

        # Validation phase
```

```
model.eval()
val_loss = 0.0
val_samples = 0
total_mae = 0.0
intersect_mae = 0.0

with torch.no_grad():
    for batch in val_loader:
        images = batch['image'].to(device)
        total_counts = batch['total_count'].to(device)
        intersect_counts = batch['intersect_count'].to(device)

        total_pred, intersect_pred = model(images)

        loss = criterion(total_pred, intersect_pred,
                        total_counts, intersect_counts,
                        model.total_weight, model.intersection_weight)

        val_loss += loss.item() * images.size(0)
        val_samples += images.size(0)

        # Calculate MAE
        total_mae += torch.abs(total_pred - total_counts).sum().item()
        intersect_mae += torch.abs(intersect_pred - intersect_counts).sum().item()

avg_val_loss = val_loss / val_samples
avg_total_mae = total_mae / val_samples
avg_intersect_mae = intersect_mae / val_samples

# Update learning rate
scheduler.step(avg_val_loss)

# Save best model
if avg_val_loss < best_val_loss:
```

```
            best_val_loss = avg_val_loss
            best_model_state = model.state_dict().copy()

        print(f'\nEpoch {epoch+1}/{num_epochs}:')
        print(f'Train Loss: {avg_train_loss:.4f}')
        print(f'Val Loss: {avg_val_loss:.4f}')
        print(f'Total Count MAE: {avg_total_mae:.2f}')
        print(f'Intersection Count MAE: {avg_intersect_mae:.2f}')
        print(f'Current LR: {optimizer.param_groups[0]["lr"]:.6f}')

    # Load best model
    model.load_state_dict(best_model_state)
    return model
```

CHAPTER-4
RESULTS AND DISCUSSIONS

4.1 DATASETS:

Data Collection : NGIT classroom CCTV images

Summary:

- 200 images
- 02 Columns
- **Train:** 160 images
- **Test:** 40 images
- **Columns:** FileName , HeadCount

4.2 MODELS:

1. CNN1
2. CNN2
3. ResNet9

4.2.1 Architecture (CNN1):

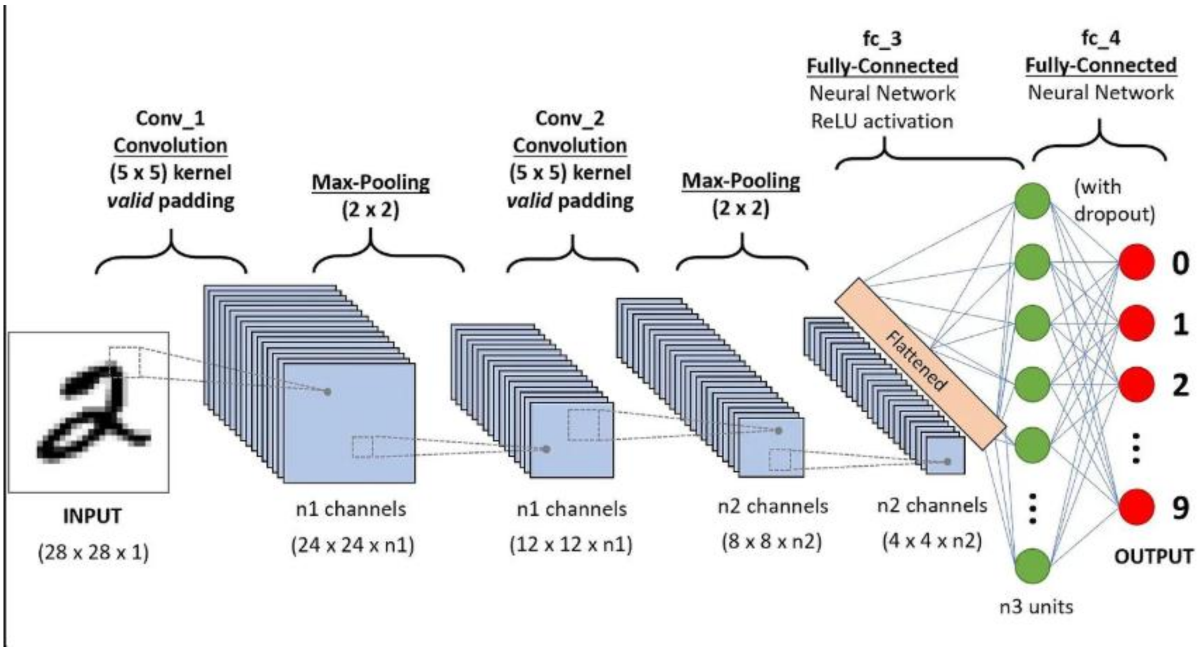


Fig 4.1: Architecture CNN1

- **Loss Function :** (MSE)^2*N= ∑(Predicted−True) ^4/N
- **Optimizer :** Adam optimizer

  Training epochs : 10

  Evaluation Metric : R^2 Score
- **Train Loss (R^2) = -**0.1798
- **Test Loss (R^2) = -**0.0895

4.2.2 Architecture (CNN2):

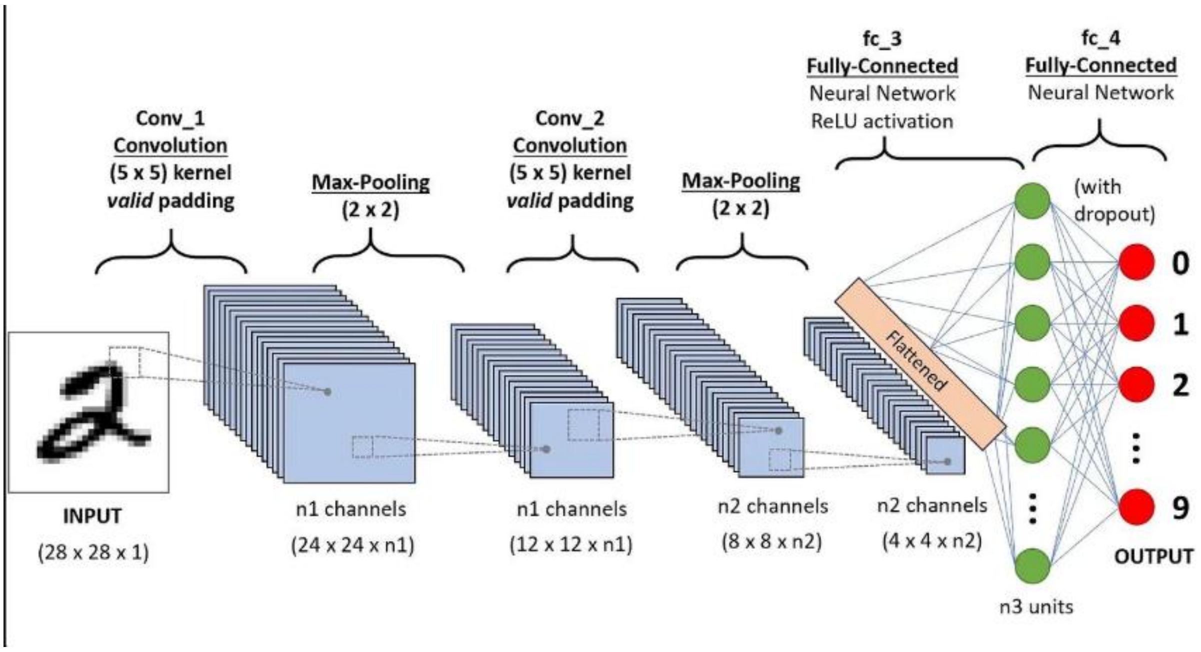


Fig 4.2: Architecture CNN2

- **Loss Function :** (MSE)^2*N= ∑(Predicted−True) ^4/N
- **Optimizer :** Adam optimizer

  Training epochs : 20

  Evaluation Metric : R^2 Score
- **Train Loss (R^2) = -**0.0486
- **Test Loss (R^2) = -**0.0692

4.2.3 Architecture (ResNet9):

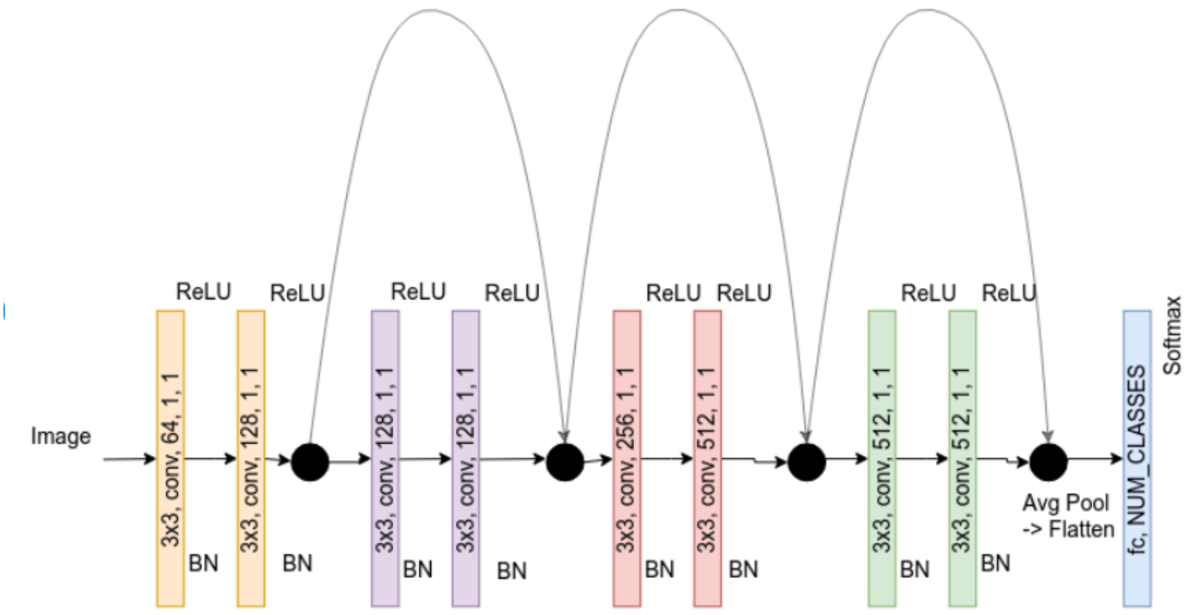


Fig 4.3: Architecture ResNet9

- **Loss Function :** (MSE)^2*N= ∑(Predicted−True) ^4/N
- **Optimizer :** Adam optimizer

  Training epochs : 20

  Evaluation Metric : R^2 Score
- **Train Loss (R^2)** = -0.0684
- **Test Loss (R^2)** = -0.1271

4.3 MODEL COMPARISON:

Model	Layers	Train Acc. (R^2)	Test Acc. (R^2)
CNN1	4xCCP + 4xLinear	-0.0895	-0.1798
CNN2	5xCCP + 6xLinear	-0.0486	-0.0692
ResNet9	2 Conv + Residual	-0.0684	-0.1271

Table 4.1: Model Comparison

4.3.1 Analysis of Results:

From the comparison table, it is evident that the **CNN2 architecture** performs slightly better than **CNN1** and **ResNet9** in terms of both training and testing accuracy. However, the results still indicate room for improvement, as all models exhibit negative R^2 scores, suggesting a lack of generalization capability.

Key Observations

1. **CNN1**:
   - A basic convolutional neural network with four convolutional and four linear layers.
   - Shows the lowest performance among the three models, with high test error due to underfitting.
2. **CNN2**:
   - Incorporates one additional convolutional and linear layer compared to CNN1.
   - Marginally better performance, indicating that deeper architectures may capture features more effectively.
3. **ResNet9**:
   - Introduces residual connections to combat vanishing gradient problems.
   - Despite its sophisticated design, the model underperforms, potentially due to the small dataset size or inadequate training epochs.

4.4 MULTI OUTPUT REGRESSION MODEL

4.4.1 DATASETS:

Data Collection : NGIT classroom CCTV images

Summary:

- Stitched 2 cams view into 1 image
- 100 images
- 03 Columns
- **Train:** 80 images
- **Test:** 20 images
- **Columns:** FileName , TotalCount, IntersectionCount

4.4.2 MODELS:

1. ResNet50 with transfer learning

4.4.3 Architecture (ResNet50):

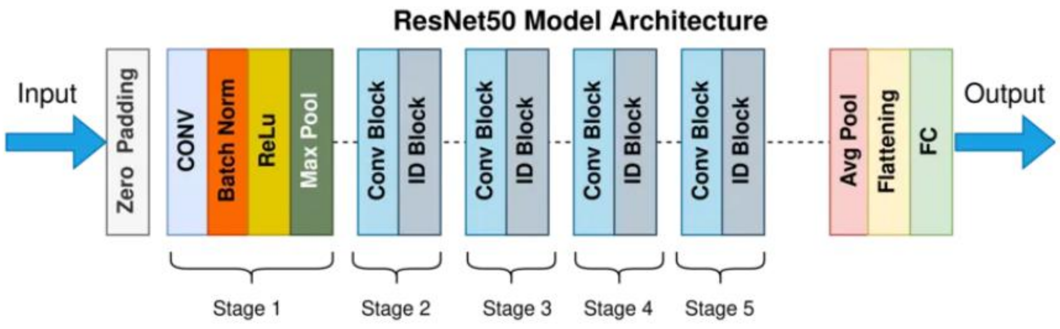


Fig 4.4.1: Architecture ResNet50

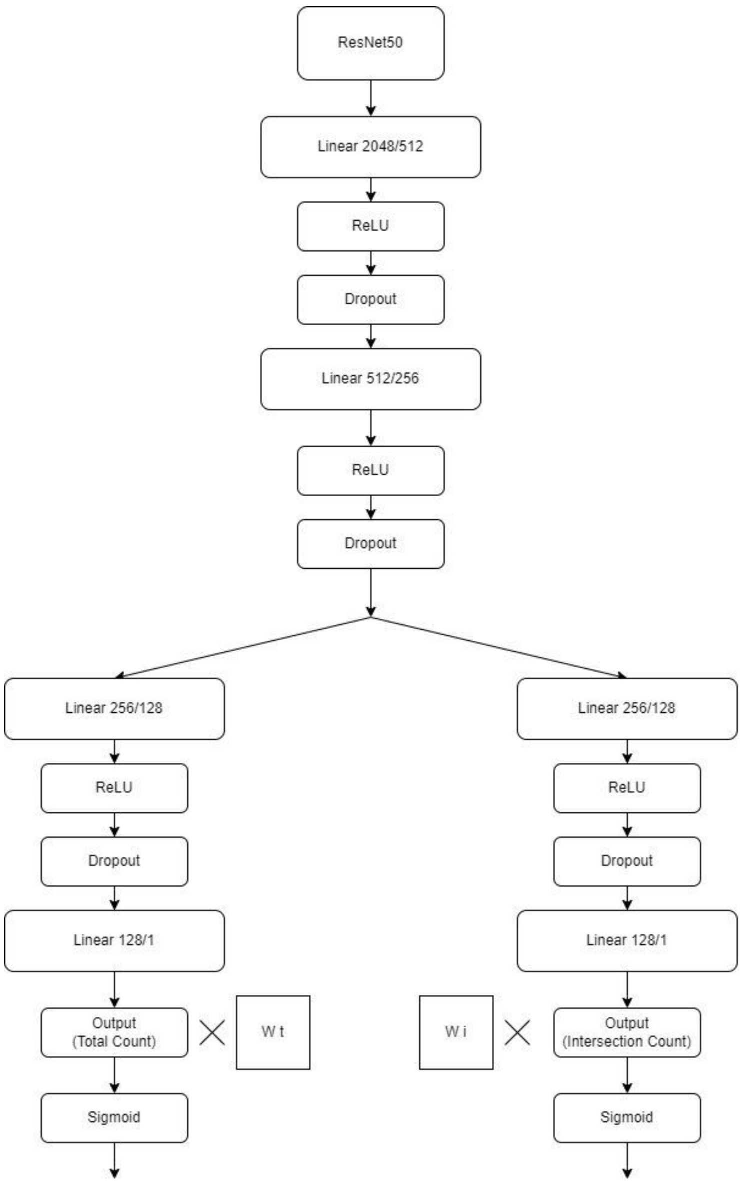
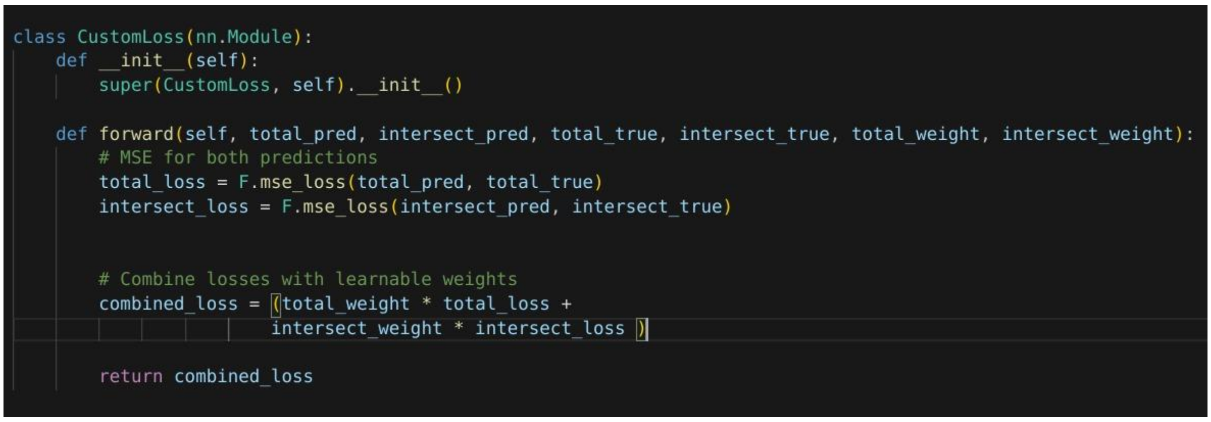


Fig 4.4.2: Architecture Custom ResNet50

- **Loss Function :** MSE(Mean Squared Error) :

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

- **Custom Loss :**



```
class CustomLoss(nn.Module):
    def __init__(self):
        super(CustomLoss, self).__init__()

    def forward(self, total_pred, intersect_pred, total_true, intersect_true, total_weight, intersect_weight):
        # MSE for both predictions
        total_loss = F.mse_loss(total_pred, total_true)
        intersect_loss = F.mse_loss(intersect_pred, intersect_true)


        # Combine losses with learnable weights
        combined_loss = (total_weight * total_loss +
                        intersect_weight * intersect_loss )

        return combined_loss
```

Fig 4.5: Custom Loss

Model	Layers	Training Loss (MAE)	Validation Loss (MAE)
ResNet50	Transfer Learning With 30 Images	T.C: 31.01 I.C: 15.83	T.C: 24.35 I.C: 12.08
ResNet50	Transfer Learning With 100 Images	T.C: 5.17 I.C: 3.13	T.C: 4.74 I.C: 2.60

Table 4.2: ResNet50 Output

T.C -> Total Count

I.C -> Intersection Count

4.5 Challenges Identified:

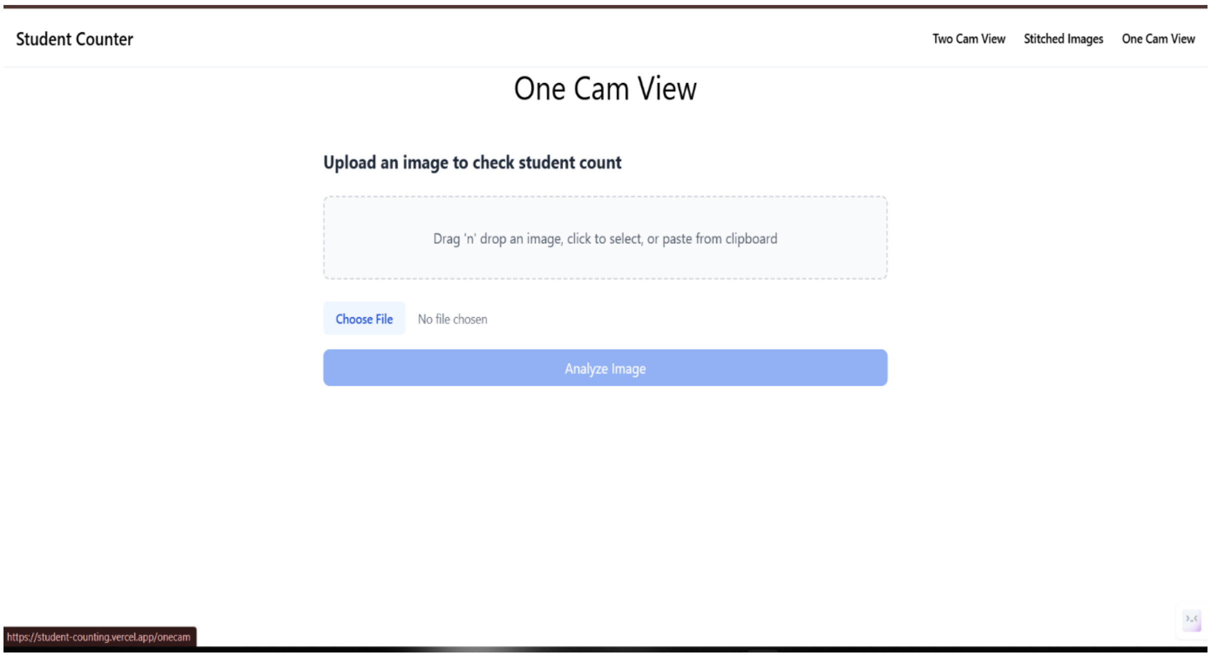
1. **Dataset Limitations**:
   - The dataset may not have sufficient diversity or size to train the models effectively.
   - Balancing the dataset (e.g., ensuring even distribution of head counts) could improve results.

2. **Overfitting and Underfitting**:
    - CNN1 and ResNet9 show signs of underfitting, while CNN2, although slightly better, does not generalize effectively.
    - Techniques like dropout, data augmentation, or regularization could address these issues.
3. **Evaluation Metrics**:
    - R^2 score is negative for all models, indicating the predictions are worse than a simple mean-based model.
    - Additional metrics such as Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE) could provide more insights into model performance.

4.6 Proposed improvements:

1. **Data Augmentation**:
    - Apply transformations such as rotation, scaling, flipping, and cropping to artificially increase the dataset size and diversity.
2. **Hyperparameter Tuning**:
    - Optimize parameters such as learning rate, batch size, and number of epochs to improve convergence.
    - Experiment with different optimizers like SGD or RMSprop for better learning.
3. **Architecture Refinement**:
    - Explore deeper or more advanced architectures like **EfficientNet**, or **DenseNet**.
    - Implement attention mechanisms (e.g., self-attention layers) to focus on critical features.
4. **Transfer Learning**:
    - Leverage pre-trained models (e.g., ResNet50, InceptionV3) to initialize weights and fine-tune on the dataset.
5. **Improved Loss Function**:
    - Use Custom Loss Function depending upon the architecture.
6. **Increased Training Epochs**:
    - Extend training epochs with proper learning rate schedules to allow the models to converge more effectively.

1.Home page:


Student Counter

Two Cam View Stitched Images One Cam View

One Cam View

Upload an image to check student count

Drag 'n' drop an image, click to select, or paste from clipboard

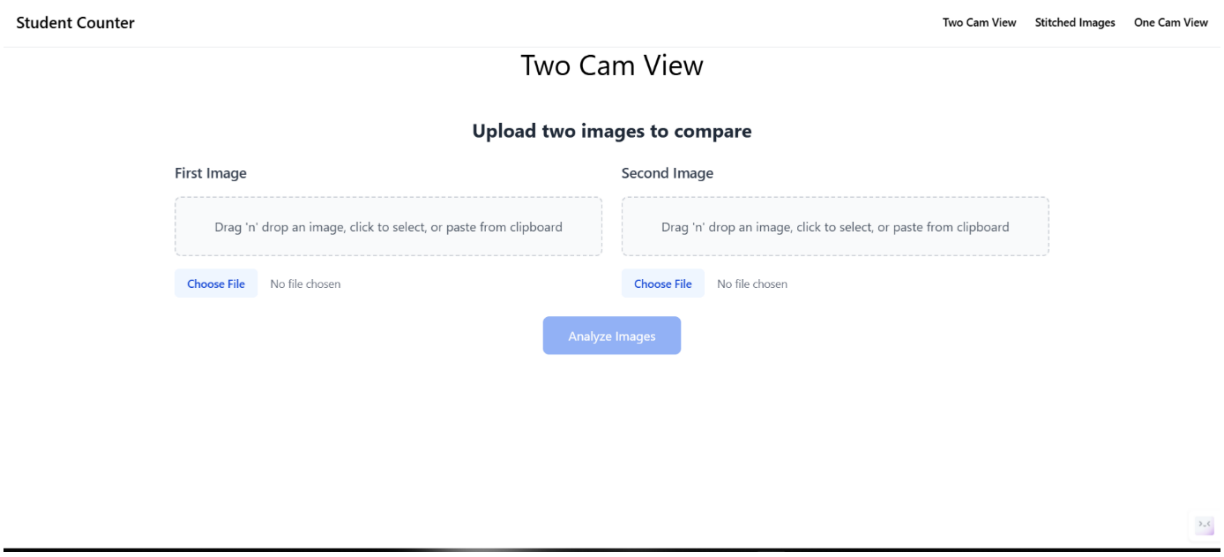
Choose File No file chosen

Analyze Image

https://student-counting.vercel.app/onecam

Fig 4.6: Home Page

2.Two cam view Page:


Student Counter

Two Cam View Stitched Images One Cam View

Two Cam View

Upload two images to compare

First Image

Drag 'n' drop an image, click to select, or paste from clipboard

Choose File No file chosen

Second Image

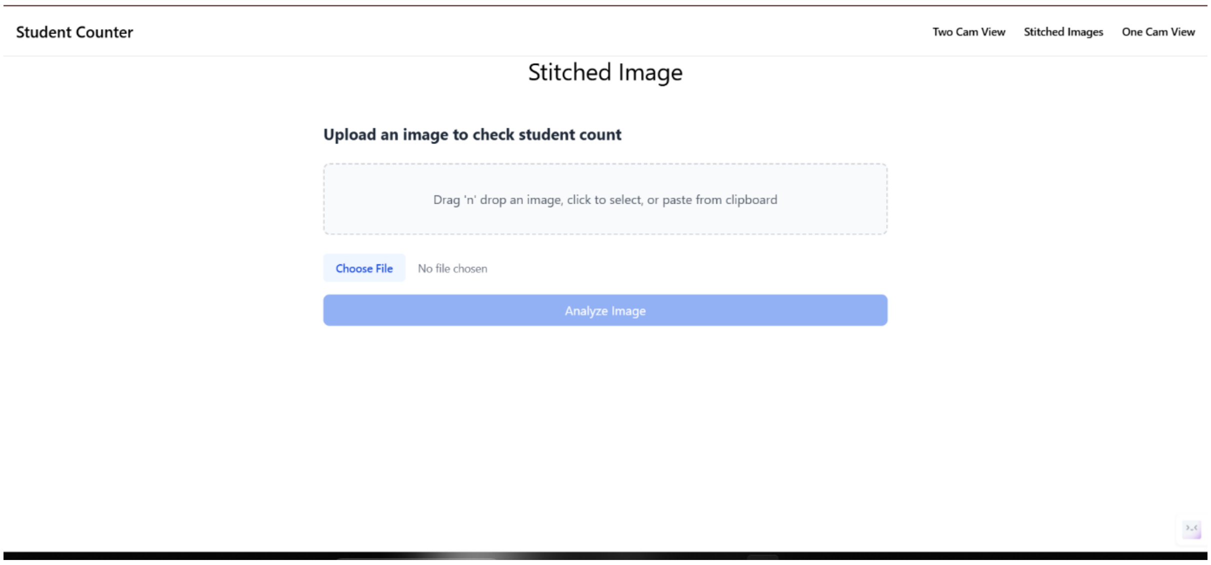
Drag 'n' drop an image, click to select, or paste from clipboard

Choose File No file chosen

Analyze Images

Fig 4.7: Two cam view Page

3.Stitched Image Page:


Student Counter

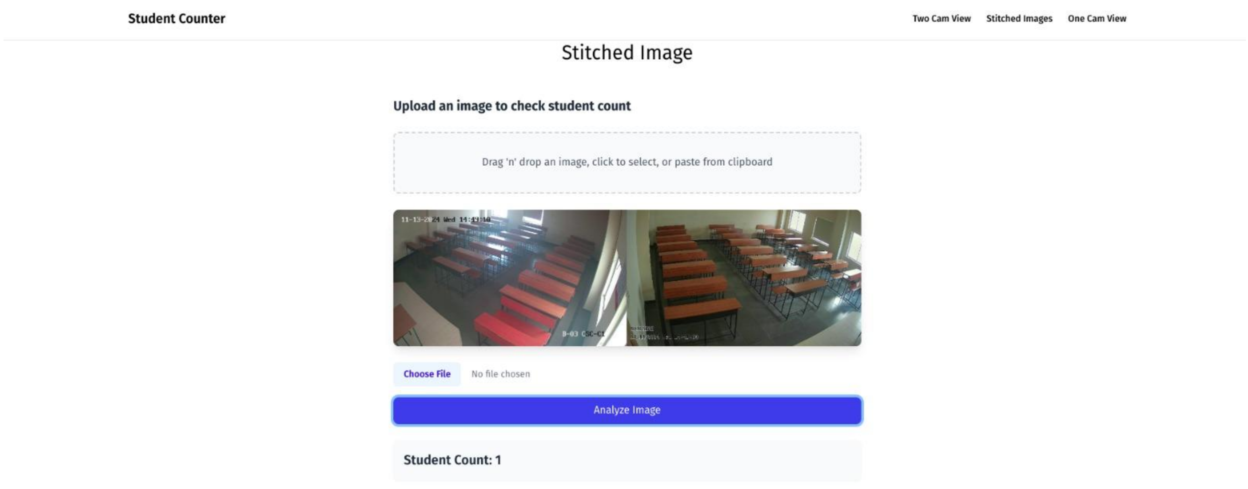
Stitched Image

Upload an image to check student count

Drag 'n' drop an image, click to select, or paste from clipboard

Fig 4.8: Stitched Image Page

OUTPUTS:

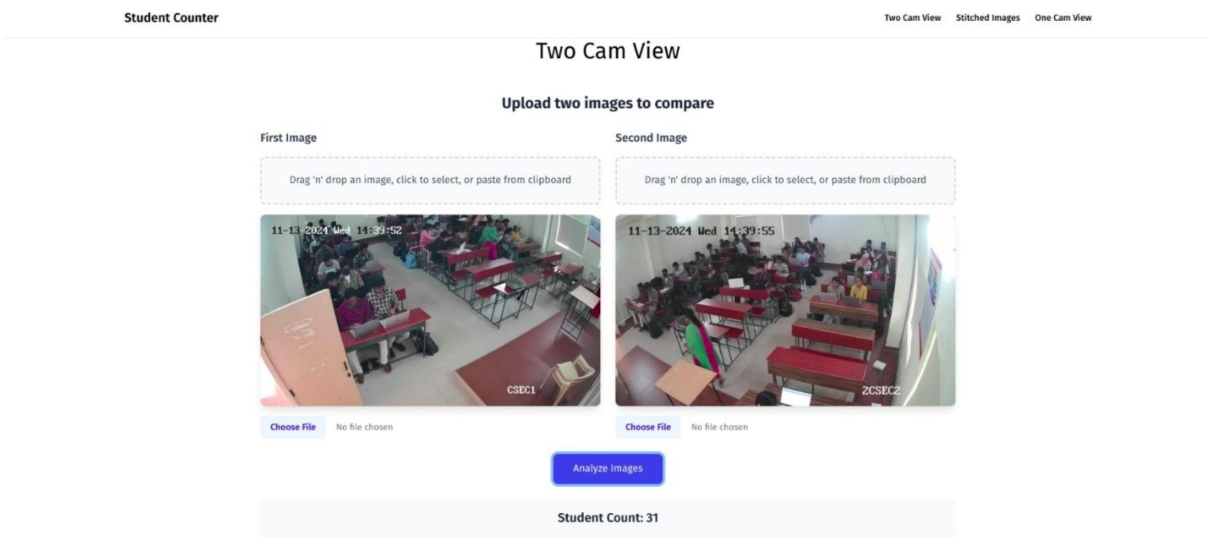

Student Counter

Stitched Image

Upload an image to check student count

Drag 'n' drop an image, click to select, or paste from clipboard

Student Count: 1


Student Counter

Two Cam View Stitched Images One Cam View

Two Cam View

Upload two images to compare

First Image

Drag 'n' drop an image, click to select, or paste from clipboard

Choose File No file chosen

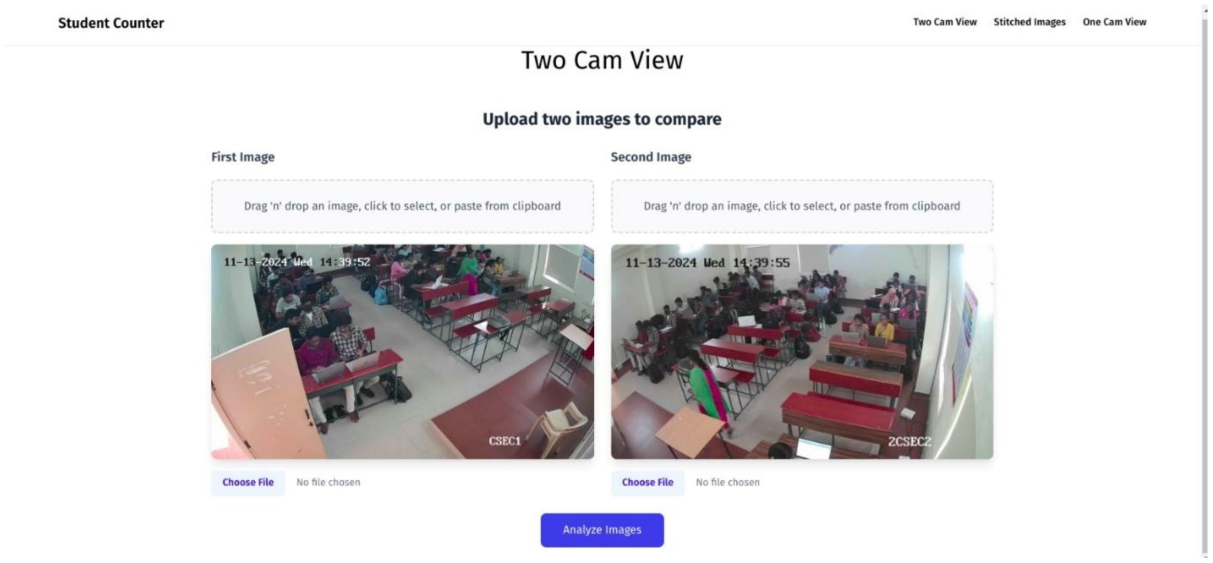
Second Image

Drag 'n' drop an image, click to select, or paste from clipboard

Choose File No file chosen

Analyze Images

Student Count: 31


Student Counter

Two Cam View Stitched Images One Cam View

Two Cam View

Upload two images to compare

First Image

Drag 'n' drop an image, click to select, or paste from clipboard

Choose File No file chosen

Second Image

Drag 'n' drop an image, click to select, or paste from clipboard

Choose File No file chosen

Analyze Images

CHAPTER-5
CONCLUSION AND FUTURE SCOPE

5.1 CONCLUSION:

The proposed Classroom Attendance Tracker aims to revolutionize student attendance management by automating the process through deep learning and transformer models. By analyzing real-time video or image feeds, the system accurately and efficiently counts students in the classroom, eliminating the need for manual methods like roll calls or sign-in sheets. This not only saves valuable teaching time but also minimizes human error, ensuring accurate and reliable attendance records. The system offers a non-intrusive and scalable solution that can be easily integrated into various classroom environments.

5.2 FUTURE SCOPE:

- The current system can be further enhanced in several ways. Future improvements could include:
- Real-time alerts: Implement real-time alerts to notify teachers of student absences or late arrivals, enabling immediate action.
- Face recognition: Integrate facial recognition technology to identify individual students, providing personalized attendance data and enabling more comprehensive student tracking.
- Improved robustness: Enhance the system's robustness to handle challenging conditions such as varying lighting, occlusions, and changes in student positions.
- Integration with existing school management systems: Seamlessly integrate the attendance data with existing school management systems for efficient record-keeping and reporting.

CHAPTER-6

REFERENCES

[1] Slimane Ennajar and Walid Bouarifi. "Monitoring Student Attendance Through Vision Transformer-Based Iris Recognition." Mathematical Team and Information Processing - National School of Applied Sciences, SAFI Cadi AYYAD University, Marrakech, Morocco.

[2] Hudaverdi Demir and Serkan Savaş. "Class Attendance System in Education with Deep Learning Method."

[3] Ankan Ghosh. "YOLOv8 Object Tracking and Counting with OpenCV."

[4] Kaiming, et al. "Deep Residual Learning for Image Recognition." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[5] He, K., Zhang, X., Ren, S., & Sun, J.
Deep Residual Learning for Image Recognition.
Published in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).