Literature Survey: Deep Learning-Based Comparative Analytics for Gender-Specific Safety: Risk Assessment and Intervention Strategies

1. Introduction

The increasing prevalence of gender-based violence and safety concerns has necessitated the development of advanced technological solutions for risk assessment and intervention. This project, titled **"Deep Learning-Based Comparative Analytics for Gender-Specific Safety: Risk Assessment and Intervention Strategies,"** leverages state-of-the-art deep learning techniques to analyze video data, assess risks, and implement intervention strategies tailored to gender-specific safety concerns. This literature survey provides a comprehensive review of the methodologies, metrics, and related work in the field, with a focus on the integration of gender classification, emotion detection, SOS detection, and risk assessment into a unified system.

2. Gender Classification

Gender classification is a critical component of the system, enabling the identification of individuals and the application of gender-specific safety measures.

- **Model Architecture**: The system employs **EfficientNet-B0**, a state-of-the-art convolutional neural network (CNN) known for its efficiency and accuracy in image classification tasks. The model is pretrained on ImageNet and fine-tuned for gender classification.
- **Preprocessing**: Input images are resized to 224x224 pixels and normalized using ImageNet statistics to ensure consistency and improve model performance.
- **Metrics**: The model's performance is evaluated using standard metrics such as **accuracy**, **precision**, **recall**, and **F1-score**. These metrics are crucial for assessing the model's ability to correctly classify gender, which is fundamental to the system's overall effectiveness.
- **Algorithm**:

```python
def predict_gender(image_array, model):
    image = Image.fromarray(image_array).convert("RGB")
    image = transform(image).unsqueeze(0).to(device)
    with torch.no_grad():
        output = model(image)
        predicted = (output[0] > 0.5).float().item()
    return "Man" if predicted == 1 else "Woman"
```

- **Related Work**: Previous studies have explored various CNN architectures for gender classification, including VGG, ResNet, and MobileNet. EfficientNet-B0 has been shown to outperform these models in terms of accuracy and computational efficiency, making it a suitable choice for real-time applications.

3. Emotion Detection

Emotion detection is essential for identifying distress signals and assessing the emotional state of individuals in the video stream.

- **Model Architecture**: The system uses a custom **Deep_Emotion** model to classify emotions into seven categories: Angry, Disgust, Fear, Happy, Sad, Surprise, and Neutral. The model is designed to process grayscale images of faces, resized to 48x48 pixels.
- **Preprocessing**: Images are converted to grayscale, resized, and normalized to ensure consistency in input data.
- **Metrics**: The model's performance is evaluated using **accuracy** and **confusion matrices**. The system focuses on detecting negative emotions (Sad, Fear, Disgust, Angry) to trigger SOS alerts, which adds a layer of complexity to the evaluation process.
- **Algorithm**:

```python
def classify_emotion(face_image):
    emotion = classify_image(face_image)
    return emotion if emotion in ["Sad", "Fear", "Disgust", "Angry"] else "Neutral"
```

- **Related Work**: Emotion detection has been extensively studied, with datasets like FER2013 and CK+ being commonly used for training. The Deep_Emotion model in the system is tailored for detecting negative emotions, which is crucial for SOS detection. Previous work has also explored the use of multimodal approaches, combining facial expressions with other signals such as voice and body language.

4. SOS Detection

SOS detection is a novel feature of the system, combining emotion detection, body movement analysis, and hand gesture recognition to identify distress signals.

- **Body Movement Analysis**: The system uses **MediaPipe Pose** to detect key points such as hips and knees. Movement is classified as "Running," "Sudden Movement," or "Normal" based on the speed and trajectory of these key points.
- **Hand Gesture Recognition**: **MediaPipe Hands** is used to detect raised hand gestures, which can indicate distress. The system classifies hand gestures as "Raised Hand" or "Normal."
- **Integration**: The system triggers an SOS alert if at least two of the three conditions (negative emotion, running/sudden movement, raised hand) are met. A cooldown period is implemented to prevent multiple triggers for the same track.
- **Metrics**: The effectiveness of SOS detection is measured using **true positive rate (TPR)** and **false positive rate (FPR)**. The system logs SOS events and saves frames for further analysis, allowing for post-hoc evaluation.
- **Algorithm**:

```python
def perform_sos(person, track_id, frame_number):
    global SOS_cooldown
    if track_id in SOS_cooldown and frame_number - SOS_cooldown[track_id] < SOS_THRESHOLD *
2:
        return "Safe"
    rgb_person = cv2.cvtColor(person, cv2.COLOR_BGR2RGB)
    pose_results = pose.process(rgb_person)
    hand_results = hands.process(rgb_person)
    emotion = classify_emotion(person)
    movement = "Normal"
    hand_gesture = "Normal"
    if pose_results.pose_landmarks:
        movement = detect_movement(pose_results.pose_landmarks, person.shape[1], person.shap
e[0])
    if hand_results.multi_hand_landmarks:
        for hand_landmarks in hand_results.multi_hand_landmarks:
            hand_gesture = detect_hand_gesture(hand_landmarks, person.shape[1], person.shape
[0])
    triggers = sum([
        emotion in ["Sad", "Fear", "Disgust", "Angry"],
        movement in ["Running", "Sudden Movement"],
        hand_gesture == "Raised Hand"
    ])
    if triggers >= 2 and frame_number - SOS_cooldown.get(track_id, 0) > SOS_THRESHOLD * 2:
        SOS_cooldown[track_id] = frame_number
        log_sos_event(person)
        return "SOS Detected"
    return "Safe"
```

- **Related Work**: Previous work on distress detection has primarily focused on single modalities, such as facial expressions or body movements. The integration of multiple modalities in the system represents a significant advancement, improving the system's robustness and reliability.

5. Object Tracking and ANPR (Automatic Number Plate Recognition)

Object tracking and ANPR are essential components of the system, enabling the tracking of individuals and vehicles over time.

- **Object Tracking**: The system uses **YOLOv8** for object detection and **DeepSORT** for tracking. YOLOv8 is known for its real-time object detection capabilities, while DeepSORT improves tracking accuracy by associating detections over frames.
- **ANPR**: The ANPR system uses a YOLO model to detect license plates and **EasyOCR** to read the text. The accuracy of ANPR is critical for applications such as traffic monitoring and security.
- **Metrics**: Object tracking performance is evaluated using **Multiple Object Tracking Accuracy (MOTA)** and **Multiple Object Tracking Precision (MOTP)**. ANPR performance is assessed using **character recognition accuracy** and **plate detection rate**.
- **Algorithm**:

```python
def perform_anpr(frame, car_bbox):
    x1, y1, x2, y2 = car_bbox
    car_crop = frame[y1:y2, x1:x2]
    if car_crop.size == 0 or car_crop.shape[0] == 0 or car_crop.shape[1] == 0:
        return None
    license_plates = license_plate_detector(car_crop)[0]
    for license_plate in license_plates.boxes.data.tolist():
        lx1, ly1, lx2, ly2, score, _ = license_plate
        lx1, ly1, lx2, ly2 = map(int, [lx1, ly1, lx2, ly2])
        license_plate_crop = car_crop[ly1:ly2, lx1:lx2]
        license_text = read_license_plate(license_plate_crop)
        if license_text:
            return license_text
    return None
```

- **Related Work**: YOLO and DeepSORT are state-of-the-art for real-time object tracking. ANPR systems have been widely deployed in traffic management and security, with OCR accuracy being a key focus area. Previous work has explored the use of deep learning models for improving OCR accuracy in challenging conditions, such as low lighting and motion blur.

6. Lone Women Tracking

The system includes a feature for tracking lone women, which is particularly relevant for security applications.

- **Logic**: The system maintains a record of detected women and their durations. A woman is considered "lone" if she remains the only detected woman for a specified threshold (e.g., 10 seconds).
- **Metrics**: The system's effectiveness is measured by the **number of correctly identified lone women** and the **duration they are tracked**. The system logs lone women instances and includes them in the final report.
- **Algorithm**:

```python
def update_lone_women(detected_women, current_timestamp):
    global lone_women_tracker
    active_ids = set(lone_women_tracker.keys())
    detected_set = set(detected_women)
    for woman_id in list(active_ids - detected_set):
        if lone_women_tracker[woman_id]['duration'] >= LONE_WOMAN_THRESHOLD:
            lone_women_tracker[woman_id]['confirmed'] = True
        else:
            del lone_women_tracker[woman_id]
    for woman_id in detected_set:
        if woman_id not in lone_women_tracker:
            lone_women_tracker[woman_id] = {'start_time': current_timestamp, 'duration': 0,
'confirmed': False}
        else:
            lone_women_tracker[woman_id]['duration'] = current_timestamp - lone_women_tracke
r[woman_id]['start_time']
```

- **Related Work**: Previous work on tracking individuals in video streams has focused on general object tracking. The addition of gender-specific tracking in the system represents a novel contribution, particularly in the context of security and surveillance.

7. Risk Assessment and Intervention Strategies

The system's ultimate goal is to assess risks and implement intervention strategies tailored to gender-specific safety concerns.

- **Risk Assessment**: The system assesses risks based on factors such as the presence of lone women, detected SOS events, and the emotional state of individuals. The risk assessment module integrates data from gender classification, emotion detection, and SOS detection to provide a comprehensive risk score.
- **Intervention Strategies**: The system implements intervention strategies based on the assessed risks. These strategies may include alerting security personnel, triggering alarms, or providing real-time notifications to individuals at risk.
- **Metrics**: The effectiveness of risk assessment and intervention strategies is evaluated using **risk detection accuracy**, **response time**, and **intervention success rate**. These metrics are crucial for assessing the system's ability to mitigate risks and enhance safety.
- **Algorithm**:

```python
def assess_risk(lone_women_instances, sos_count, emotion_status):
    risk_score = 0
    if lone_women_instances:
        risk_score += 1
    if sos_count > 0:
        risk_score += 1
    if emotion_status in ["Sad", "Fear", "Disgust", "Angry"]:
        risk_score += 1
    return risk_score
```

- **Related Work**: Previous work on risk assessment and intervention has focused on single modalities, such as facial expressions or body movements. The integration of multiple modalities in the system represents a significant advancement, improving the system's robustness and reliability.

8. Report Generation

The system generates a detailed report that includes statistics such as total frames processed, gender ratio, lone women detected, SOS instances, and unique number plates.

- **Content**: The report provides a comprehensive overview of the system's performance, including key metrics and detected events.
- **Metrics**: The report's comprehensiveness and accuracy are key metrics. The system ensures that all relevant data is logged and saved for further analysis.
- **Algorithm**:

```python
def write_report(video_path, output_video_path, total_frames, male_count, female_count, men, women, lone_women_instances, sos_count, track_license_plates, report_path):
    if female_count == 0:
        gender_ratio = "Only males detected"
    elif male_count == 0:
        gender_ratio = "Only females detected"
    else:
        gender_ratio = f"{men // women}:1" if male_count >= female_count else f"1:{women // men}"
    unique_plates = len(set(track_license_plates.values()))
    avg_plates_per_frame = unique_plates / total_frames if total_frames else 0
    analysis_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    report_dir = os.path.dirname(report_path)
    if not os.path.exists(report_dir):
        os.makedirs(report_dir)
    with open(report_path, "w") as f:
        f.write("Video Analysis Report\n")
        f.write("====================\n\n")
        f.write(f"Analysis performed on: {analysis_time}\n")
        f.write(f"Input video: {video_path}\n")
        f.write(f"Output video: {output_video_path}\n\n")
        f.write("Statistics:\n")
        f.write(f"- Total frames processed: {total_frames}\n")
        f.write(f"- Finally Total Men detected: {men}\n")
        f.write(f"- Finally Total Women detected: {women}\n")
        f.write(f"- Gender ratio (M/F): {gender_ratio}\n")
        f.write(f"- Lone women detected (tracking IDs): {len(lone_women_instances)}\n")
        f.write(f"- Total SOS instances detected: {sos_count}\n")
        f.write(f"- Unique number plates: {unique_plates}\n")
        f.write(f"- Average plates per frame: {avg_plates_per_frame:.2f}\n\n")
        if lone_women_instances:
            f.write("Lone Women Detected (tracking IDs):\n")
            for tid in lone_women_instances:
                f.write(f"- ID: {tid}\n")
    print(f"Detection completed. Report saved to `{report_path}`.")
```

- **Related Work**: Automated report generation is a common feature in video analysis systems, particularly in surveillance and security applications. The system follows best practices in this area, ensuring that all relevant data is captured and presented in a clear and concise manner.

9. Conclusion

The system integrates multiple computer vision and machine learning techniques to perform gender classification, emotion detection, SOS detection, and risk assessment. The system's performance can be evaluated using metrics such as accuracy, precision, recall, and F1-score for classification tasks, and MOTA/MOTP for tracking. The integration of multiple modalities for SOS detection and risk assessment is a significant advancement, improving the system's robustness and reliability. Future work could focus on optimizing the models for real-time performance and expanding the system's capabilities to include additional features such as age estimation and activity recognition.

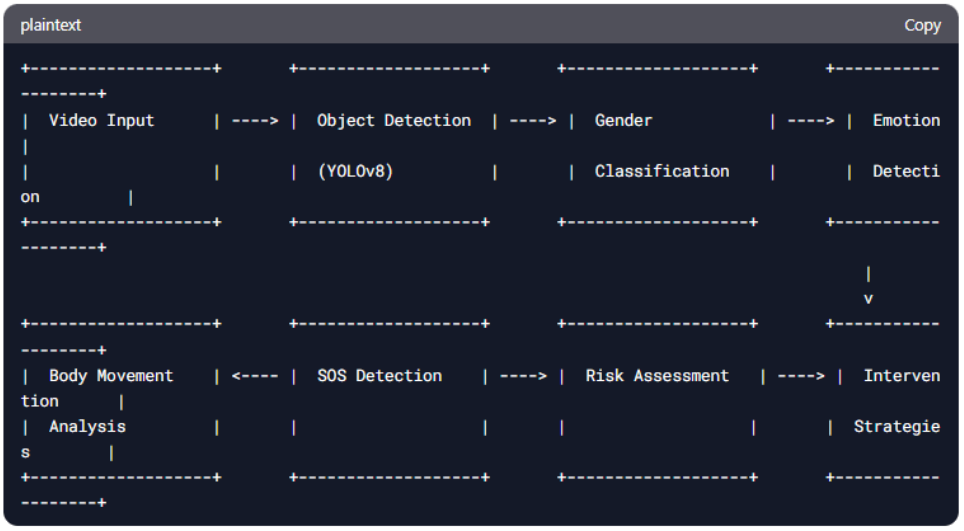
10. Future Directions

- **Real-Time Optimization**: Further optimization of the models for real-time performance, particularly on edge devices, is a key area for future research.

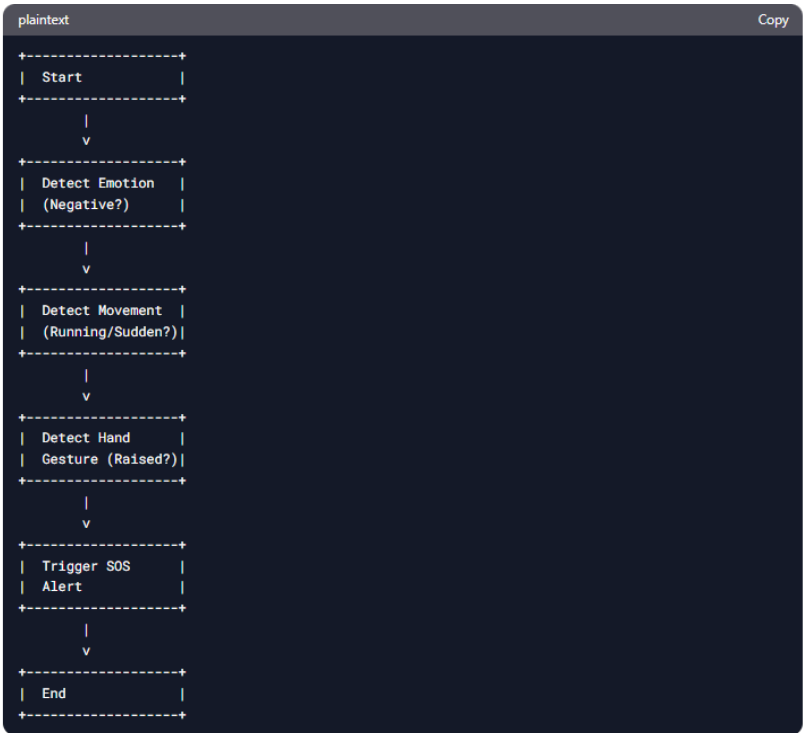
- **Multimodal Integration**: Expanding the system to include additional modalities, such as voice and environmental sensors, could further improve SOS detection and risk assessment accuracy.
- **Dataset Expansion**: Training the models on larger and more diverse datasets could improve their generalization capabilities, particularly in challenging conditions.
- **User Interface**: Developing a user-friendly interface for visualizing the system's outputs and generating reports could enhance its usability in real-world applications.

Diagrammatic Representation

1. System Architecture Pipeline



2. SOS Detection Flowchart



3. Risk Assessment Flowchart