Vision-Based Human Monitoring System for Corporate Spaces

Navaneeth Krishnan[1], **Onkar Mane**[1], **Garv Ubhan**[1], **Sakshi Indolia**[2], **Aditya Kasar**[2],**Shailendra Aote**[2]

[1]SVKM's NMIMS, School of Technology Management and Engineering, Navi Mumbai, India
{navaneethks.official, onkar2septmane, garvubhan}@gmail.com
[2]SVKM's NMIMS, Navi Mumbai, India
{sakshi.indolia, aditya.kasar,shailendra.aote}@nmims.edu

***Abstract*—Modern corporate workplaces lack granular tracking of employee-based activities that affect productivity. Traditional methodologies for evaluating a team's efficiency often include errors, bias, and lack of a systematic approach. Existing technologies such as time tracking are also inefficient in assessing productivity, as they focus only on quantitative aspects like 'number of hours worked' rather than tracking employees' interaction with objects like laptops, mouse, cell phones, etc. To overcome this challenge, deep learning techniques combined with vision-based systems for object detection and action classification are crucial.**

This paper presents a custom YOLO (You Only Look Once) based system for corporate workspace monitoring through object detection and action classification to a limited extent. For this scope, a subset of the COCO (Common Objects in Context) dataset has been used with 26 target classes trained on 15437 images. We developed a model capable of assessing the productivity (focused work, collaboration, meal breaks) of a team by recognising objects and actions done in the space at time t. Our working hypothesis is that we will achieve an mAP50 (Mean Average Precision) greater than 0.5 while training the model on classes consisting of corporate work-related objects. The following experiment will enable us to conclude either that we reject the null hypothesis or we fail to reject the null hypothesis.

Keywords: YOLOv8, MS COCO, object detection, Workspace Monitoring, action classification, quota enforcement.

I. Introduction

Modern corporate spaces are environments where the productivity of an employee is often dependent on several factors, including employee action over some time, movement, and interaction with objects. To assess the productivity of an employee or a team, organizations often use traditional methodologies like peer-to-peer review, feedback forms, and self-reports that can include bias, failing to monitor minute tasks like usage of screens, number of breaks, etc. To overcome this challenge, we have developed a YOLOv8-based vision monitoring system that uses deep learning-based object detection and is capable of identifying 26 target classes with limited action classification.

Our approach uses non-intrusive video analysis to detect objects (e.g., laptops, phones, food items) and infer actions (e.g., 'Working (Focused)', 'Unhealthy Break') based on their co-occurrence and spatial relationships. Developing the system and following such an approach often includes several challenges that have to be overcome. Our COCO dataset originally consisted of 80 target classes out of which only 26 were relevant to our scope. Therefore we created a subset of the original dataset by enforcing class quotas (number of images) for a semi-balanced representation. We used YOLOv8x which consists of pre-trained weights making it most compatible for training on the COCO dataset [1]. However, the model's capability is limited to object identification, not classification. To overcome this we designed a rule-based system post training which includes combinations of classes with 40+ spatio-temporal conditions. The limitations include a lack of a facial recognition system, the ability to identify only 26 objects, can deliver nearly precise reports on productivity only if there are multiple people in the space.

II. Hypothesis

A. Null Hypothesis (H_0)

The final model, which has been trained on the subset of the COCO dataset consisting of 26 classes, will achieve an mAP50 greater than 0.5.

B. Alternative Hypothesis (H_1)

The final model, which has been trained on the subset of the COCO dataset consisting of 26 classes, will achieve an mAP50 of less than 0.5 due to the class imbalance caused by quota enforcement.

In this research, we aim to determine whether the YOLOv8 model is efficient enough for corporate workspace monitoring to a limited extent. Therefore, we have taken mAP50 (Mean Average Precision) as an indicator to determine the model's performance on the validation set after training. If the null hypothesis is rejected, it will indicate that YOLOv8-based architecture is not sufficient for monitoring objects and actions in a corporate setup consisting of multiple employees.

III. Literature Review

A. Background and motivation

The criteria on how productivity would be assessed using a deep learning model is itself challenging when it comes to classification and identification models. The increasing shift of work to remote arrangements has introduced challenges

in assessing the productivity of an employee. To counter this, a CNN (Convolutional Neural Network) based model was proposed to track the eye movements of employees during their work schedules [2]. In their work, the CNN architecture is trained with approximately 3000 images of different eye states to classify employees' attentiveness over a specific period. This approach showed improved accuracy over other traditional models. However, this kind of approach poses severe limitations since the evaluation metrics are solely dependent on eye states, which is not sufficient considering the complex nature of real-time remote workspaces. Furthermore, the binary classification used in the system fails to distinguish between different types of productive activities, such as focused individual work versus collaborative effort by a team.

When compared to alternative approaches in the literature, several opportunities for improvement became clear, like pose estimation based on real-time multiperson key points where the team tracked the status of a construction worker using OpenPose [3]. However, this approach is only limited to pose estimation, which discards the possibility of object identification.

While employee monitoring systems are gaining popularity in industrial and corporate settings, scholarly investigations into corporate workspace monitoring remain surprisingly limited. Current literature mostly focusses on manufacturing and industrial safety compliance to improve safety in the construction domain, which focusses on worker re-identification (ReID) with personal protective equipment (PPE) [4].

B. Challenges in corporate workspace monitoring

- **Limited Dataset Relevance:** Despite recent progress, current action classification and object detection datasets lack relevant data for corporate applications. While there are large datasets like AVA-Kinetics, which consists of 700 action classes, most of the classes are irrelevant. Only 15% apply to corporate monitoring scenarios (e.g., 'typing' or 'meeting' vs. irrelevant classes like 'playing golf'). This forces developers and researchers to adopt filtration methodologies as demonstrated in our COCO adaptation (retaining only 26/80 classes) or collect proprietary datasets, which raises various privacy and ethical concerns.
- **Fragmentary Evaluation Metrics:** Existing models employ isolated evaluation metrics like eye-state tracking, pose estimation, and keystroke dynamics which fail to capture holistic productivity, therefore flagging a requirement for multidimensional assessment frameworks like object classification and action classification.
- **Computational Intensity for Real-Time Processing:** Training and deploying a real-time monitoring system can be computationally costly, especially if the organisation lacks a cloud infrastructure.

As evidenced by our benchmarks (Fig. 1), maintaining 60 FPS on 1080p streams requires model compression (FP16 quantisation), architecture optimisation like batch size less than 16 (vs. standard 26) or workers greater than 20, and hardware constraints (NVIDIA V100 or newer GPUs).

Requirement	Typical Specs	Corporate Viability Threshold
Frame Rate	60 FPS	<30 FPS causes action misses
GPU Memory	12GB (RTX 3080)	<8GB fails at HD resolution
Latency	<50ms/frame	>100ms disrupts real-time alerts

Fig. 1. Benchmarks

C. Research Gaps and Future Scope:

Despite advancements, key research gaps remain:

Lack of Context-Aware Action Recognition: Current literature never explored how to tackle the problem of superficially similar but contextually distinct activities. ("Active typing during a meeting" (productive) vs "Excessive typing during a video call" (potentially distracted)). Without proper contextual understanding, the model may reward (false positive) or penalise (false negative) based on common patterns [5].

Limited Cross-Cultural Validation: Most research is based on Western office environments, tech company settings, and high-resource contexts, which often pose the question of "How do monitoring needs differ for Asian corporate hierarchies? Manufacturing office hybrids?"

Overemphasis on Individual Metrics: Current research neglects team-based productivity metrics, collaboration patterns, and cultural indicators. (e.g., a system might flag an employee as "distracted" when they're mentoring a colleague offline.) [6]

IV. Methodology

A. Dataset Selection

For this study, we used the MS COCO (Microsoft Common Objects in Context) dataset, which is a large-scale dataset consisting of 180,000+ images with 800,000+ annotation combinations and 80 object categories (e.g., dog, bottle, laptop, banana, cell phone, etc.) [7]. The dataset consists of 2 folders: "imagesclass imbalances in jpg format and "annotations", which consists of an instance.json file for training and validation. This dataset fits the best for our scope since our aim is only to detect objects and classify actions (to a limited extent) from a test sample and draw meaningful inferences for productivity assessment.

B. Filteration

We implemented a strict filtration process for our COCO dataset to address the following needs:

*1) **Domain Relevance**:* Corporate workspaces include actions to a limited extent only; therefore, it is crucial to eliminate classes and images that are not relevant to the scope for optimising resource utilisation and avoid wastage. Our analysis showed that only 26 out of the total 80 classes had direct relevance to workspace application. These include a person, laptop, cell phone, keyboard, mouse, book, TV, dining, table, etc.

*2) **Computational Efficiency**:* Including irrelevant classes (e.g., 'surfboard', 'zebra') would reduce the speed by 30-40% and would increase the model's size by 40%; additionally, it will introduce more noise in the newly learnt model.

*3) **Action Recognition Precision**:* The model suffers from class imbalance (not enough images for some classes), compromising its accuracy. It can also lead to the misclassification of similar object classes (e.g., an apple and a red ball).

However, filtering the COCO dataset might be challenging due to its unstructured nature (all images belonging to different classes are present in one single folder). To figure out the images and their corresponding classes and to ensure that only required images are processed into the new subset, we needed to rely on a crawling logic to understand the nature and classes of the images. We started crawling COCO's 80 object classes and filtered down to 26 workspace-relevant classes. Initially the code loads COCO's JSON annotation files (train + val splits). For each annotation pointing to an image, it checks if the object's category_id exists in our TARGET_CLASSES.values(), therefore creating an initial filter. If the annotation pointing to an image doesn't consist of the category_id from the 26 classes, it will automatically discard the image. COCO understands classes and labels as numerical values; therefore, we map human-readable names (e.g., person: 0, laptop: 68) to COCO's official class IDs. These hardcoded mappings are crucial to ensure we only filter out required classes. We incorporated a dual counting mechanism to track unique images: 'class_image_counts', containing each class (using sets to avoid duplicates), and 'class_instance_counts', which counts all the occurrences of each class across all the images. This helps us to understand the coverage (how many distinct images contain laptops) and the density metrics (how many laptop instances exist in total).

C. Dataset Preparation and Quota Enforcement

After the initial filtration, we implemented a rigorous sampling strategy to counter the class imbalance issue of the COCO dataset. Our main aim was to allocate more quota to frequently occurring classes (e.g., 5000 images for 'person', 2000 images for 'laptop', and 100 images for 'fork').

*1) **Dynamic Sampling Algorithm**:* For each target class, we selected all the available image IDs and shuffled them randomly to avoid biased selection.

*2) **Quota Allocation**:* Fig. 2 represents class-wise quota allocation for all 26 classes. Each class has different quotas based on its importance and relevance to the scope. Since employees are the main subject of this scope, the class 'person' was allocated the maximum quota of 5,000 images among all the other classes. This is done so that the model can train more accurately in such classes.

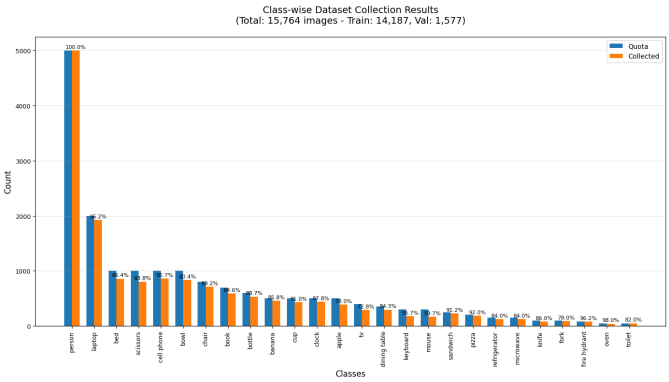

Fig. 2. Class-wise quota allocation for all 26 classes

*3) **Cross-Class Image Allocation**:* Images consisting of multiple classes we prioritised more to maximise data efficiency and preserve natural object co-occurrence patterns. This is the reason why some classes (e.g., 'persons') reached 100% quota while others fell short.

*4) **Train-Validation Split**:* A 90-10 split was applied for training and validation, respectively. This ensures class balance among both sets.

Handling Class Imbalances: For severely under-represented classes (less than 70% quota), we augmented existing data by applying a controlled transformation (rotation, lighting changes). The controlled transformation was only applied to 2 classes in our case: 'Mouse' (+23 samples) and 'Keyboard' (+35 samples). Additionally, we also incorporated class grouping, where we merged similar classes (e.g., 'fork/knife' as 'cutlery') where appropriate and implemented class weights during training to compensate for remaining imbalances.

D. Final Dataset Statistics

- **Total Images:** 15,764 (14,187 train / 1,577 val)
- **Class Coverage (a):** 19/26 classes achieved greater than 80% quota.
- **Class Coverage (b):** 1/26 classes achieved greater than 100% quota.
- **Class Coverage (c):** 6/26 classes fell short by 15-45%
- **Identified Limitations:** Persistent gaps in peripheral office items (mouse, keyboard) and some class definitions were too narrow (e.g., 'dining table' vs 'meeting table').

V. Model Selection

A. Technology used

*1) **Sagemaker environment**:* AWS Sagemaker is a service managed by Amazon Web Services (AWS) used for developing and deploying artificial intelligence and deep learning models. The preprocessing, data loading, training, and validation were done in Sagemaker's environment due to its support for GPUs and multiple VCPUs. It offers a variety of tools like

Jupyter Notebook for easy coding [8]. We used Sagemaker's g4xd.8x large instance type, which has 16 GB of GPU RAM and 128 GB of CPU RAM. This helped us to wrap up the training process at much faster rates by using 28 VCPUs out of the instance's 32 VCPUs. The instance and the environment supported rapid prototyping and debugging, which helped us to identify errors through an inbuilt logging mechanism.

*2) **Software Stack**:* The software stack included PyTorch 2.6.0, which is one of the leading deep learning frameworks, and which powered the current scope of implementation of the YOLOv8 framework with GPU acceleration and optimised performance through CUDA. Interestingly, all the alternative models like SlowFastR50 and SlowFastR101, that we earlier used for experimentation belong to PyTorch, demanded much more data augmentation and parameters [9]. Sagemaker's environment needed to use GPU and CPU simultaneously to speed up the process; hence, we included CUDA 12.4, which is a toolkit used for efficient handling of GPUs on G4 instances [10]. We also used pandas to map annotations to labels and retrieve them from CSV files for efficient data handling.

B. YOLOv8

YOLO (You Only Look Once) is a great approach for deep learning-based object detection with the capability of processing videos in real time, making it suitable for corporate applications to a limited extent. Unlike traditional models which are complex and with regional proposal networks, YOLO is capable of framing object detection as a single regression problem that simultaneously predicts class probabilities and bounding boxes from a full image in one evaluation [11]. The unified YOLO architecture has remarkable speed and maintains competitive accuracy.

The core logic lies in dividing the input images into different grids (grid-like systems) where each grid cell predicts multiple bounding boxes and confidence scores. Unlike other models which might take multiple evaluations to predict multi-class images, YOLO only takes one evaluation per image, and this is the main reason why the architecture is named 'You Only Look Once'. YOLOv8's architecture's evolutionary improvements over previous architectures (YOLOv2, YOLOv4, etc.) address several critical requirements concerning corporate applications.

- **Feature Extraction:** The model's enhanced backbone provides superior feature extraction capabilities for detecting small work-related objects (e.g., 'mouse' and 'cell phone'). The model preserves spatial relationships between frequently co-occurring items (e.g., 'laptops' and 'persons') [12].
- **Anchor free detection:** The transition to an anchor-free detection head simplifies the process of training on our custom COCO dataset and reduces hyperparameter tuning complexity [13].
- **Robust:** The model can process high-resolution input images (640×640) at real-time speeds (52 FPS on our hardware configuration) while maintaining accuracy.

However, there were several challenges that we faced while working on this model. One of the key challenges was the compatibility of COCO's annotations with the YOLOv8 model. The transformation of COCO's annotations to YOLO format required careful consideration for work-related applications. Therefore, we implemented a multi-stage conversion pipeline that filtered all the COCO's 80 original classes to our target of 26 classes. Then we normalised the bounding box coordinates to YOLO's default format (centre-x, centre-y, width, height) with values ranging from 0 to 1. This kind of normalisation ensured that detection accuracy was preserved across varying image resolutions encountered in different work camera setups. To counter real-world application challenges (camera conditions), we leveraged YOLOv8's built-in data augmentation pipeline with corporate-specific modifications, which enhanced transformations like random perspective warping to simulate various camera angles while suppressing irrelevant augmentations like vertical flips that could disrupt the natural orientation of office objects. The loss function of the model combines classification, objectness, and bounding box regression terms with automatic weighting, adapting dynamically to our class distribution. Where certain workspace classes (like 'person' and 'laptop') appeared more frequently than others (like 'pizza' or 'fork'). This counters the problem of overfitting the model to dominant classes while still maintaining strong detection performance across all workspace objects.

C. Model Architecture (YOLOv8x)

*1) **Backbone (CSPDarknet53-X)**:* It consists of enhanced CSPNet architecture with cross-stage partial connections with depth multiplier (d) = 1.00 and width multiplier (w) = 1.25. Additionally, the backbone comprises a modified stem block using 6×6 convolution with SiLU activation and integrated Mish activation in deeper layers for gradient flow similar to earlier models like YOLOv4 [14].

*2) **Neck (PANet++)**:* The neck consists of a bidirectional feature pyramid network with channel-wise attention and concatenation-based feature fusion with channel shuffle.

*3) **Head (Anchor-Free)**:* Decoupled prediction heads (classification + regression) with a task-aligned assigner for label assignment.

*4) **Key working mechanism** :* Firstly, the input image (640×640×3) undergoes a multi-scale feature extraction process with the backbone generating feature maps at strides 8, 16, and 26. The neck performs top-down + bottom-up feature aggregation, and the head predicts class probabilities (26-dim for workspace classes), bbox coordinates (cx, cy, w, h normalised 0-1), and objectness score (IoU-aware confidence) [15].

D. Critical Parameters and Optimization Features in YOLO8x

- **Total params:** 68.2M (vs 86.7M in YOLOv5x).
- **GFLOPs:** 165.4 at 640×640 input.
- **BatchNorm momentum:** 0.03
- **Box loss gain:** 7.5

- **Classification loss gain:** 1.0
- **Mosaic-9 augmentation:** vs Mosaic-4 in v5.
- **Optimizer:** AdamW optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$).
- **Scheduler:** Cosine LR scheduler with warmup.
- **EMA model averaging:** decay = 0.9998
- **CIoU loss for Bbox regression:** v = 3.0

E. Training Parameters

- **Epochs:** Set to 100.
- **Early Stopping:** Patience = 10 epochs.
- **Batch Size:** 16 (optimized for GPU memory).
- **Workers:** 28 (parallel data loading).
- **Image Size:** 640×640 (fixed for consistency).
- **Optimizer:** AdamW (adaptive LR + weight decay).
- **Learning Rate:** Initial lr0=0.001 (autoadjusted).
- **Weight Decay:** 0.0005 (L2 regularization).
- **Checkpoints:** Saved every 10 epochs (save_period=10).
- **Directory:** Continues in runs/train/exp (exist_ok=True).

Since YOLOv8 is a pre-trained model, it automatically saves 'best.pt' (the best model over a set number of epochs) and 'last.pt' (the recent epoch's progress). It automatically generates crucial information like 'results.csv' (consisting of metrics), confusion matrix, F1 curve, label plots, R curve, PR curve, etc., in the 'runs/weights' directory.

VI. Evaluation Standards

YOLOv8 is a state-of-the-art object detection framework that performs well when it comes to identifying and localising objects (e.g., 'person', 'laptop', 'cell phone') in images or video frames with more accuracy. The efficiency of the model makes it ideal (to some extent) for workspace-related applications. However, there are several limitations and challenges we faced after developing the model. The YOLOv8 model trained on COCO datasets is only capable of identifying objects, not complex human actions, which deviates from our scope. To counter this, we built a hybrid system that layers a rule-based action classification module over YOLOv8's object detections. This approach uses YOLOv8's strengths while extending its utility through interpretable logic and temporal reasoning. In summary, we incorporated a reward-based system, tracked multiple individuals, and generated detailed reports, drawing directly from the provided implementation. The output consists of an '.mp4' video file (consisting of object detection using bounding boxes), a '.txt' file (consisting of the inferred report), and a '.JSON' file (report in structured data).

A. Extending YOLOv8 with a Rule-Based Action Classification System

Based on spatial and temporal relationships between detected objects and people, we developed 40 activity rules. These rules map the combinations of objects to a specific action, such as:

- **Rule Example 1:** If a person is detected with a laptop and no cell phone or pizza → "Working (Focused)"
- **Rule Example 2:** If a person is detected with a laptop and a cell phone → "Distracted Work"
- **Rule Example 3:** If a person is detected with pizza and no laptop → "Lunch Break"

These 40 rules are evaluated frame by frame with spatial constraints (e.g., MAX_DISTANCE=200 pixels), ensuring objects are pointed to the correct person. To enhance stability, we added temporal smoothing via a 5-frame action buffer (ACTION_BUFFER) to reduce mistakes by choosing the most frequent action in a given time frame.

B. Reward-Based System

For evaluating actions, we must quantify each action by assigning it a reward. Each action is assigned multiple metrics:

1) ***Rating (0-10)****:* Shows contribution to work output (e.g., 0.9 for "Working (Focused)", 0.2 for "Team Lunch").

2) ***Wellness (0-1)****:* Impact on mental/physical health (e.g., 0.9 for "Mental Reset", 0.4 for "Unhealthy Break").

3) ***Energy (-1 to 1)****:* Effect on fatigue or recharge (e.g., -0.5 for "Working (Focused)", 0.9 for "Mental Reset").

4) ***Tags****:* Labels for categorisation (e.g., ["work", "focus"]).

Once the actions are detected, each action along with its rewards is logged alongside the duration into a JSON file for collective performance analysis at the end of execution. For each individual ('person'), energy levels are cumulatively updated.

$$\text{energy\_levels[pid]} += \text{energy\_change} \times \text{duration}$$

C. Multi-person tracking and rule filtering

For team-based activities, each individual must be assigned different IDs (P0, P1, etc.) using a simple Intersection over Union (IoU)-based tracking mechanism (IOU_THRESHOLD=0.3). For each frame, bounding boxes of class 'person' are compared with those from the previous frame. If the set IoU exceeds the threshold, the person won't lose their existing ID; otherwise, a new ID will be assigned to the person. To ensure that employees are using the objects, objects are associated with the nearest person (within MAX_DISTANCE). This will only associate objects to a person if the object is nearer to the class 'person' with ID: X (X stands for any ID associated with class 'person'). This kind of approach is not suitable for real-time deployment, as taking MAX_Distance as the only parameter to determine the usage of objects by a specific person might return false positives or false negatives. However, this kind of approach is computationally lightweight and sufficient for static or semi-static workspace environments. Additionally, the 'person_tracks' dictionary stores a detailed history of each person (frame_count, action, rating, timestamp, duration, objects).

The 'analyze_frame' method evaluates more than 40 conditions per person, looking for specific object combinations. These may include work-focused conditions where the presence of a laptop often points to a work-related action being done at timeframe 't' unless overridden by the presence of

distractive objects (e.g., cell phone → "Distracted Work"); break conditions where the presence of food like pizza and bananas will shift the classification to breaks (e.g., multiple people + food → "Team Lunch"); social conditions where multiple people without productive objects trigger "Socialising" or "Collaborating"; and fallback, where if no specified rule is valid, it defaults to "Mental Reset" or "Unknown Activity".

D. Individual Report Generation

The method 'generate_report' computes all detailed analytics for each person.

1) ***Action Breakdown****:* Counts and durations of each action

2) ***Weighted Metrics****:* The productivity and wellness scores are calculated as time-weighted averages (e.g., sum(productivity * duration) / total_time).

3) ***Energy Tracking****:* It is about the cumulative energy impact based on action durations.

4) ***Eating Times****:* Timestamps of break-related activities for wellness insights.

For example, if a person's (ID: P0) action includes "Working (Focused)": 400s, "Lunch Break": 150s, "Distracted Work": 50s, then the productivity score (0.9 × 400 + 0.3 × 150 + 0.5 * 50) / 600 = 0.73 and (-0.5 × 400 + 0.7 × 150 + -0.6 × 50 = -125) energy scores would be 0.73 and -125, respectively. Additionally, timestamps are converted to human-readable formats (e.g., "12:34:56") for eating times.

E. Overall Performance and Conclusion

In the last step, the system creates an aggregated analysis report by combining all the individual reports into one. It includes variables like average productivity/wellness (weighted by each person's total time), collaboration time (sum of "collaborating" durations across all individuals), and issues (counts of low-rated actions, e.g., rating less than 4, e.g., "TV Break").

As discussed earlier, the final output includes a JSON file (report_detailed.json) for structured data, a text file (report_detailed.txt) with a human-readable summary, including individual breakdowns and recommendations, and an annotated video (output_monitor_detailed.mp4) showing object labels with bounding boxes.

VII. Challenges and Limitations

YOLOv8's Inherent Limitation in Contextual Understanding: The model can only detect predefined object classes (e.g., 'laptop', 'person', 'cell phone') and cannot classify complex actions or contextual behaviours (e.g., "Working (Focused)" vs "Distracted Work") [16].

Lack of Temporal Dynamics in YOLOv8: The model operates on a per-frame basis without the understanding of temporal relationships between objects or actions over time. Example: A person holding a pizza in one frame might be classified as "Lunch Break", but if they pick up a laptop in the next frame, the model alone cannot classify it as "Working While Eating" without external temporal smoothing.

YOLOv8 and COCO Dataset Limitations: The model, trained on the subset of the COCO dataset, can detect object classes only (e.g., 'person', 'laptop', 'pizza'). It works very well for localisation and classification tasks but cannot interpret actions. Example: If the model detects a person and a book, it will label them as two separate objects rather than classifying them as a single task ("Reading the book"). 40+ custom logics were implemented to counter this problem. However, these logics alone are not sufficient, making the model not suitable for real-world corporate applications.

A. Quota Completion

In a previous experimental approach, an equal number of images were selected from each class (200 images per class) to balance the subset. But for frequently occurring object classes like 'person' and 'laptop', the images were insufficient, making it less accurate over the validation set. To tackle this problem, we implemented quota enforcement where we allocated different quotas to different classes so that the model can train more accurately over frequently occurring classes. However, even after quota enforcement, many object classes out of the 26 only reached an average of 85% completion due to COCO's bias toward everyday objects rather than niche corporate items. For example, the classes 'laptop' and 'person' had ample samples, but 'dining table' or 'cup' (used for "Mental Reset") had fewer instances, leading to uneven detection performance.

B. No Real-Time Live Feed Support:

The current implementation has the ability to take pre-recorded .mp4 files (e.g., input1.mp4) as input rather than capturing it live through a camera. For example, to monitor a workspace in real time, the code would need to replace cv2.VideoCapture("input1.mp4") with cv2.VideoCapture(0) for webcam capturing, requiring additional buffering and latency management. This kind of approach is not suitable for real-world corporate spaces.

C. Minute Challenges in Implementation

1) ***Tracking Instability****:* The IoU-based tracking (IOU_THRESHOLD=0.3) faced challenges like occlusion or rapid movements, leading to ID switches or duplicate IDs.

For example, if two employees cross paths, their bounding boxes might overlap, causing P0 and P1 to swap IDs mid-video, messing individual reports.

2) ***Energy Metric Subjectivity****:* The energy values (e.g., -0.5 for "Working (Focused)") are assigned arbitrarily and may not reflect real fatigue levels.

3) ***Scalability****:* As the number of people increases in the frame, associating objects with the correct person becomes error-prone due to its simplistic approach (MAX_DISTANCE=200). For example, in an overcrowded room, 'pizza' might be incorrectly linked to some other person, but in actuality, someone else might be eating it.

*4) **Misclassification of Similar Classes**:* Similarly looking object classes are often misclassified, causing errors during evaluation. For example, an iMac (Apple's monitor) might be misclassified as a 'tv' due to its larger screen causing errors in the reports.

The primary challenge we faced was during the filtration and optimising stage of the COCO dataset for workspace monitoring. COCO's single-file structure (e.g., instances_train2017.json) combines annotations for all 80 classes in one single file, making it hard to filter out pointers to images and required classes. Additionally, all the images belonging to different classes were dumped in a single image folder, 'Train2017', again making it difficult to figure out the image classes. However, as discussed above, we applied several custom filtration logics that successfully mapped annotations to their respective images.

VIII. Model Performance Evaluation

We evaluated the model's performance using:

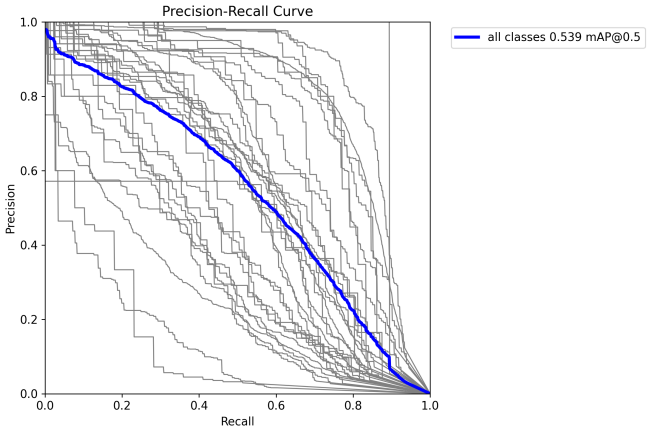

Fig. 3. Precision-Recall Curve

*1) **Number of epochs**:* We stopped training the model at epoch 60 since there was a negligible increase in mAP50 metrics (our aim of scoring mAP50 greater than 0.5 was already achieved).

*2) **Overall Detection Accuracy (mAP Metrics)**:* At epoch 60, the model achieves a mean Average Precision (mAP) of 0.53906 at IoU=0.5 (mAP50) and 0.38639 at IoU=0.5:0.95 (mAP50-95). This indicates that the model is moderately performing over all classes. The precision-recall curve confirms this with an mAP@0.5 of 0.539, suggesting a balanced trade-off between precision and recall.

*3) **Class Imbalance Impact**:* Due to quota enforcement, several lower-priority classes (less frequently occurring classes) were underperforming when compared to high-priority classes. The class distribution plot shows a significant imbalance, with 'person' having 30,000 instances, while classes like 'suitcase' or 'toothbrush' have fewer than 1,000. Additionally, in the confusion matrix, we can observe that frequently

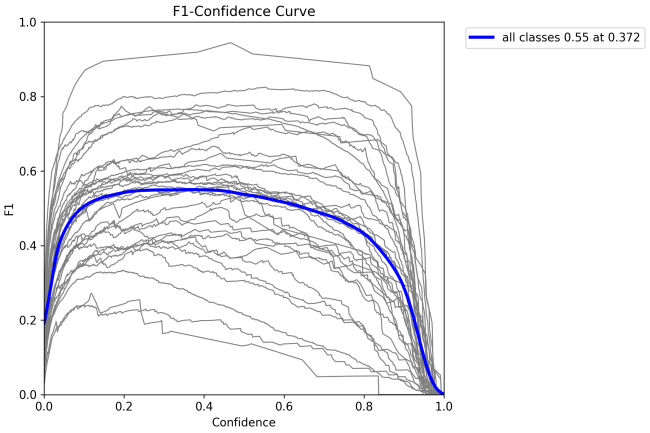

Fig. 4. F1 Curve

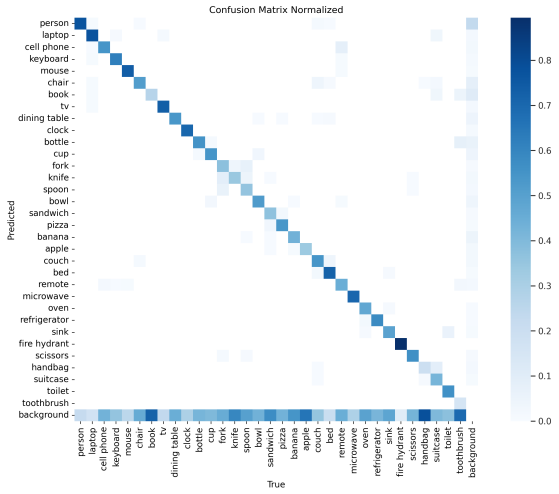

Fig. 5. Normalised Confusion matrix

occurring classes like 'person' achieved high accuracy (0.9), but rarer classes like 'suitcase' are often misclassified (e.g., predicted as 'handbag' with 0.2 probability).

*4) **F1 Score and Confidence Threshold**:* The F1 – confidence curve indicates an optimal F1 score of 0.55 at a confidence threshold of 0.372. This indicates that the model is performing well and balancing precision and recall at this threshold, but performance drops for higher confidence levels, indicating overconfidence for some predictions.

*5) **Precision and Recall Trade-Off**:* The precision-recall curve shows a precision of 0.6 at a recall of 0.5, dropping as recall increases. This indicates that the model is struggling to maintain high precision for less frequently occurring classes.

*6) **Validation losses**:* At epoch 60, validation losses are box_loss=1.0547, cls_loss=1.14603, and dfl_loss=1.29446. This explains why the model is misclassifying similarly looking objects ('tv' and 'laptop') due to relatively high cls_loss.

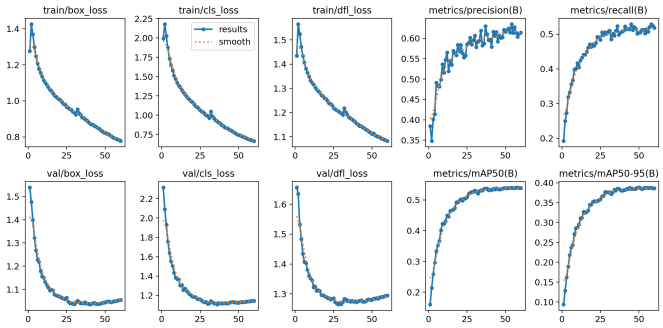

Fig. 6. Training progress report till epoch 60

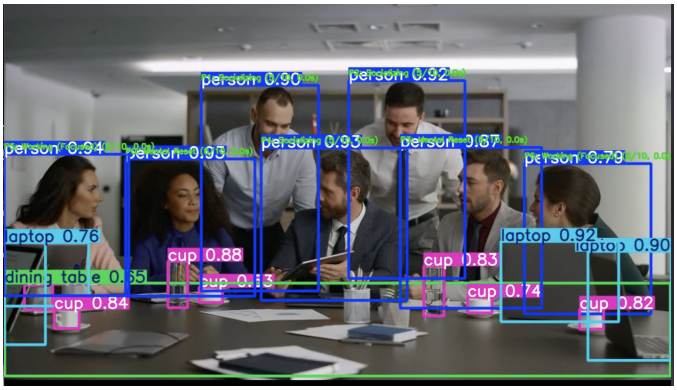

Fig. 7. Snapshot of the test video with object classification (video designed by freepik)

IX. Results and Discussion

The model started training from epoch 0 and lasted till epoch 60. The average time taken to complete one epoch was 16 minutes and 54 seconds. The total time spent on model training was 16 hours and 54 minutes. At epoch 60, the model achieved an mAP50 of 0.53906 and mAP50-95 of 0.38639, indicating consistent performance over all classes. Training losses were low (box_loss=0.61437, cls_loss=0.38937, dfl_loss=0.87537), but validation losses were higher (box_loss=1.0547, cls_loss=1.14603, dfl_loss=1.29446), which suggested overfitting. The confusion matrix shows high accuracy for 'person' (0.9) but confusion for 'cup' was 0.1(e.g., misclassified as 'bottle' with 0.1 probability), reflecting class imbalance from the distribution plot ('person': 30,000 instances, 'cup': 5,000). To verify its effectiveness, we tested the model on more than 20 samples (designed by Freepik). Let us take one of the test samples into consideration. The 14-second video (1280x720, 25 FPS) consisted of 7 people, 8 cups, 1 pen holder, 1 dining table, 1 tablet, and 4 laptops (based on ground truth). The model was able to predict all 7 people correctly with a confidence of 0.92–0.95, 9 cups with a confidence of 0.69–0.89, 1 book with a confidence of 0.43, 4 laptops with a confidence of 0.76–0.94, and 1 dining table with a confidence of 0.39–0.65.

The variability in the confidence score is due to the video's cinematic nature (a zoom-out shot where initially most of the objects were not clear due to their partial appearance). This testing suggested that the model struggled to detect similarly looking objects (1 pen holder was misclassified as a cup, making it a total of 9 cups, and the tablet was misclassified as a book due to the tablet's sleek and thin design). The optimal F1 threshold maintains reliable detections, but inconsistent detection of smaller objects suggests the need for balanced training data, fine-tuning, or temporal smoothing to improve performance in workspace monitoring. The final report (in .txt format) showed varied productivity (0.20–0.84) and wellness (0.62–0.88) scores across 7 individuals (P0–P6), probably due to the nature of the video (Zoom out shot).

From the report, we infered that P6 had the highest productivity (0.84) with 6.2s of focused work, while P1 and P2 had the lowest (0.20), spending all 13.9s socialising (more reward was given to P6 since the person was using a laptop). The average wellness level was 5.25, but P6's low energy level (-2.8) suggests fatigue. Socialising was prevalent (e.g., P1, P2: 13.9s), but focused work varied (P5: 7.9s, P6: 6.2s). P0 and P4 reported "Unhealthy Breaks" (5x and 28x), contributing to the 33 total issues noted. P4 was flagged with frequent eating (28 instances) due to the presence of cups. However, if we refer to ground truth, there was no eating action performed during the entire timeline, which indicated a lack of conditional logic applied (the model may flag eating even if the cups are empty). Additionally, P6 was flagged with the lowest energy metric, which is not true if compared to ground truth. This might be due to the partial appearance of P6 throughout the video. Overall metrics concluded that the video represents a balanced workspace (average productivity (2.49) and wellness (5.25)), but the lack of collaboration time (0s) and 33 unhealthy breaks suggest areas for improvement.

```
Overall Analysis
--------------------------------------------------
Avg productivity: 2.49
Avg wellness: 5.25
Collab time: 0
Issues:
  - Unhealthy Break: 33x

```

Fig. 8. A snippet of report_detailed.txt

X. Conclusion

This study mainly focused on the YOLOv8-based workspace monitoring system, evaluated through the training process and testing process, which has revealed both its strengths and weaknesses. From our experiments we could infer that the YOLOv8-based workspace monitoring system demonstrates promising capabilities when it comes to object detection if trained on more samples. Training results at epoch 60 (mAP50=0.53906, mAP50-95=0.38639) indicate moderate accuracy, though overfitting and class imbalance lead to inconsistent detection of rarer objects like 'spoon' and 'book'. As seen in the confusion matrix and precision-recall curve, it performed well on frequently occurring classes.

Despite these strengths, the model lacks action classification capabilities unless action logic is applied with a rewarding system. The experiments revealed that the system's overfitting, class imbalance, and inconsistent detection of rarer objects have limited its reliability for fine-grained monitoring in real-world corporate settings. While the model is greatly suited for detecting classes like 'person' and 'laptop', providing high-level insights on it, it still lacks capabilities to classify similarly looking objects. Since every action is interdependent in a real workspace, even small misclassifications can lead to major changes in reports. Hence, we can conclude that the current implementation is not suitable for real-world deployment. The future work should be focused on increasing the sample size, using videos as input for training instead of images for granular action classification, implementing temporal smoothing to improve consistency, and exploring advanced techniques like transfer learning to enhance robustness.

REFERENCES

[1] M. Desai, H. Mewada, I. M. Pires, and S. Roy, "Evaluating the performance of the yolo object detection framework on coco dataset and real-world scenarios," in *Proceedings of the 15th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2024)*, EUSPN. Leuven, Belgium: TBD, October 2024, p. TBD.

[2] P. G. Gopal, N. S. Subhash, and R. A. Anil, "Analysis of employee surveillance system using deep learning models," in *2022 6th International Conference On Computing, Communication, Control And Automation (ICCUBEA*, 2022, pp. 1–6.

[3] Y. Liu, Z. Zhou, and Y. Wang, "A tracking method of multi-workers on-site with kalman filter and openpose," in *Proceedings of the International Conference on Construction and Real Estate Management (ICCREM 2021)*. TBD: ASCE, 2021.

[4] H. Cheng *et al.*, "Deep learning system for worker re-identification and ppe classification in construction safety," *Proceedings of the 2022 IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[5] J. Wu, Z. Kuang, L. Wang, W. Zhang, and G. Wu, "Context-aware rcnn: A baseline for action detection in videos," *ArXiv*, vol. abs/2007.09861, 2020.

[6] L. Yang, D. Holtz, S. Jaffe, S. Suri, S. Sinha, J. Weston, C. Joyce, N. Shah, K. Sherman, B. Hecht, and J. Teevan, "The effects of remote work on collaboration among information workers," *Nature Human Behaviour*, vol. 6, no. 1, pp. 43–54, 2021.

[7] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," *ArXiv*, vol. abs/1405.0312, 2015.

[8] R. V. Kulkarni, A. Thakur, S. Nalbalwar, S. Shah, and S. Chordia, "Exploring scalable and efficient deployment of machine learning models: A comparative analysis of amazon sagemaker and heroku," in *2023 International Conference on Information Technology (ICIT)*, 2023, pp. 746–751.

[9] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "Slowfast networks for video recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 6202–6211.

[10] M. Garland, "Parallel computing with cuda," in *2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS)*. IEEE, 2010, p. 1.

[11] M. Sohan, T. Ram, and V. Ch, *A Review on YOLOv8 and Its Advancements*, 01 2024, pp. 529–545.

[12] W.-S. Hsu, G.-T. Liu, S.-J. Chen, S.-Y. Wei, and W.-H. Wang, "An automated clubbed fingers detection system based on yolov8 and u-net: A tool for early prediction of lung and cardiovascular diseases," *Diagnostics*, vol. 14, no. 19, p. 2234, 2024. [Online]. Available: https://www.mdpi.com/2075-4418/14/19/2234

[13] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "Yolox: Exceeding yolo series in 2021," *ArXiv*, vol. abs/2107.08430, 2021.

[14] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Scaled-yolov4: Scaling cross stage partial network," *ArXiv*, vol. abs/2011.08036, 2021.

[15] M. Yaseen, "What is yolov8: An in-depth exploration of the internal features of the next-generation object detector," *arXiv preprint arXiv:2408.15857*, 2024. [Online]. Available: https://arxiv.org/abs/2408.15857

[16] Y. Cao, Q. Cao, C. Qian, and D. Chen, "YOLO-AMM: A real-time classroom behavior detection algorithm based on multi-dimensional feature optimization," *PMC*, 2024.