Scalable Object Detection in Mixed Reality using Incremental Re-training and One-shot 3D Annotation

Alireza Taheritajar* Augusta University
Jeffrey Benson† Augusta University
Anthony Gibson‡ Augusta University
Brandon Wilburn§ Augusta University
Jieqiong Zhao¶ Augusta University, Purdue University
Jason Orlosky‖ Augusta University, Osaka University

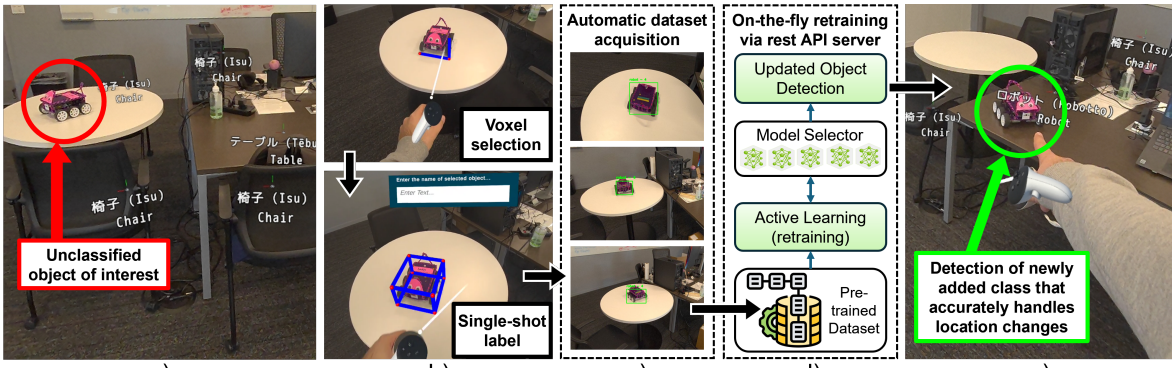

Figure 1: Images showing our one-shot labeling and active incremental learning approach for object detection, including a) a new object class that the end-user wants to be added to the object detector (YOLOv11), b) the one-shot labeling process in which the user boxes and labels the target, c) automatic acquisition of frames for dataset generation, d) the high-level architecture of our rest API server for model re-training, and e) practical use of the newly detected object at a new location for a language study application.

Abstract

While object detection can be incredibly useful for a variety of augmented and mixed reality applications, achieving a large number of classifiable objects with high accuracy without extremely large deep learning (DL) or object recognition models is still difficult. More importantly, object recognition frameworks are often rigid in that they don't provide a direct means to add new classes to pre-trained models in real time.

In this paper, we introduce a novel approach that enables on-demand training of new object classes for consistent detection of in-situ objects for virtual labeling and interaction. By leveraging knowledge of the 3D location of an object in the scene taken from a mixed reality (MR) display's environment mesh, we are able to automate the labeling of subsequent 2D images taken from the front-facing camera, which requires only a single, initial labeling interaction from an end-user. In addition, we have developed a continual learning approach that allows for on-the-fly retraining of the classifier and provides accurate classification quickly enough for the model to be practically usable in MR applications. We validate this approach by measuring the re-training time required for various object configurations, provide a comparison to other classification strategies, and analyze how the addition of object classes affect detection continuity across 3D scenes. We also demonstrate that labeling interactions work for practical applications in AR that are dependent on object detection, such as language learning, procedural instruction, or manufacturing guidance.

Index Terms: Mixed reality, augmented reality, incremental learning, object detection, 3D annotation.

1 Introduction

Recent advancements in computer vision, augmented, and mixed reality (AR/MR) have begun to enable various applications in the fields of education [48], medical visualization [39, 30], maintenance and repair [40], disability assistance [38] and self-autonomous driving [2, 1]. Many of these applications, especially in the fields of education and training, require the seamless detection of physical objects for virtual labeling and interaction from the end user. However, object detection frameworks designed for MR headsets have not been able to handle large numbers of classes [58] or the real-time addition of new classes for on-demand or scalable use with MR applications. One reason is that most state-of-the-art object detection models primarily operate on manually labeled 2D images [7]. Additionally, they need a large number of annotated images, and the process to generate these datasets can be labor-intensive, time-consuming, and costly [45]. More importantly, classifiers and detector frameworks are structured such that the addition of new classes require extensive retraining of the model.

To alleviate these problems, we propose the integration of **active incremental learning via one-shot 3D annotation** into mixed reality (MR) workflows, as outlined in Figure 1. To do so, an end-user first places a 3D bounding box on an object of interest, as shown in a) and b) of Figure 1, which triggers the automatic generation of a labeled dataset from 2D images taken at different viewpoints. Secondly, we have designed an active incremental learning framework for the purpose of retraining a detection model on-the-fly. Our end-to-end architecture extends the state-of-the-art YOLOv11 [32]

*e-mail: ataheritajar@augusta.edu
†e-mail: jbenson@augusta.edu
‡e-mail: antgibson@augusta.edu
§e-mail: bwilburn@augusta.edu
¶e-mail: jzhao@augusta.edu
‖e-mail: jorlosky@augusta.edu

2025 IEEE International Symposium on Mixed and Augmented Reality (ISMAR) | 979-8-3315-8761-1/25/$31.00 ©2025 IEEE | DOI: 10.1109/ISMAR67309.2025.00040

object detection model by incrementally integrating new classes using replay techniques [26]. Not only does this allow for the addition of new object classes that are quickly usable in practice, but it avoids catastrophic forgetting, i.e., the loss of accuracy in previously trained classes [13, 12]. Newly labeled objects can be immediately used for MR applications without disruption to the end-user's application, and the process of acquiring training images is fully automated after an initial label is placed.

Through a server-side active incremental learning framework, we employ a fixed set of exemplars from both new and existing classes using transfer learning. Furthermore, we include a representational state transfer API (REST-APIs) [53] to ensure seamless communication with AR/MR headsets, along with efficient dataset organization and data format conversions. This technique can be applied in various scenarios, offering a fast, precise, and scalable alternative to traditional, labor-intensive human labeling. Our goal is to improve the model's effectiveness in handling new objects across diverse settings to better support individuals in real-world situations that want to make use of object labels for MR applications.

To demonstrate this approach in practice, we have also implemented a language-learning application in which end-users can both label objects and make use of the resulting annotations for vocabulary study. For this purpose, we present several pre-trained locational-contextual models tailored for an individual's personal environment. In addition, we test the addition of newly added object classes for immediate use during vocabulary study.

This capability is particularly useful for AR/MR technologies that rely on object detection for immersive experiences that support real-world interactions, which supports learner engagement and information retention [29]. We present four primary contributions, including:

- A one-shot 3D annotation and dataset acquisition system that automatically acquires images for retraining object detection models,
- An end-to-end active incremental learning framework using an inference engine to detect objects and provide timely on-the-fly retraining for practical use in MR applications,
- A series of system evaluations that highlight the effectiveness of transfer learning at producing usable detection results in a real-world context, and
- Demonstration of the benefits of this approach in practice through a secondary language learning application.

2 Related Work

Related work primarily falls into two categories, including 1) continual learning and replay methods for improving the practical use of object detection, and 2) the use of labeling and object detection for applications in MR. Our work is focused on the intersection of these two fields, with the goal of improving object detection for real-world use.

2.1 Overview of Continual Learning

As previously mentioned, one difficulty for MR is to be able to make use of large numbers of detectable classes with an efficient or mobile object detection framework. The typical training process of a deep network involves preparing large-scale datasets, such as images [15] or texts [9], through which the network goes through multiple epochs [67]. However, in practical settings, training data is frequently obtained in a streaming format [23], and we need to actively retrain models based on new data or new classes to prevent low accuracy due to data drift [22, 20]. Continual learning, also known as incremental learning, refers to the ability of a model to learn continuously from a stream of data, integrating new information while retaining previously acquired knowledge [46].

This approach is particularly relevant in dynamic environments where new data continuously emerges, such as natural language learning contexts, which are discrete, compositional, and context-dependent [9]. Various methods have been proposed to address the challenges associated with continual learning, including catastrophic forgetting, where the model forgets previously learned information when trained on new data [31]. Techniques like regularization [37, 33], replay [55], context-specific components [42], knowledge distillation [25], and dynamic architectures [64] have been developed to mitigate these issues [14, 63].

2.2 Replay Methods as a Means for Re-training

Another restrictive issue with most detection models is that the detection accuracy of previously learned classes falls drastically when new training data is added, which is often referred to as catastrophic forgetting. This is a major problem for MR applications, since most previously detected classes need to maintain a high enough accuracy for to generate a virtual label or augmentation in real time.

To alleviate this problem, it is possible to shuffle all the training datasets and re-train with them over multiple epochs to prevent the model from forgetting previously learned classes. This approach is called transfer learning and produces a baseline model as the upper bound of class-incremental learning [67].

In contrast, without access to all data, incremental learning is prone to catastrophic forgetting. Various methods seek to prevent this and match the baseline model's accuracy. The data replay method is one of the applicable methods with higher accuracy in this area. Replay plays a crucial role in the human cognitive process [54]. For instance, a student preparing for final exams should review the textbooks and notes to retrieve past memories and knowledge. Similarly, in machine learning, the replay method involves storing a subset of past training data and replaying this data while training new tasks. This allows the model to rehearse previously learned information, thereby retaining knowledge from earlier tasks while incorporating new information. Replay methods can be more efficient than other continual learning techniques requiring complex model architecture modifications. It is applicable to a wide range of neural network architectures and learning scenarios. The choice of samples to store and replay is crucial. Random sampling may not always be optimal; strategies like importance sampling [11] can be used to improve performance. There are primarily two types of replay methods:

- **Direct Replay (Memory Replay):** In this approach, we must store a buffer of past experiences and somehow take samples from this buffer during training on new tasks. It is straightforward to implement, but it can be memory-intensive since it requires storing actual data samples [6].
- **Generative Replay:** Instead of storing raw data, it uses generative models (like GANs or VAEs) to generate synthetic samples of past experiences. This reduces the memory requirement by storing model parameters rather than data samples, but it can be computationally intensive [56].

The YOLO (You Only Look Once) models are well-known for their real-time object detection capabilities [52], and the latest version, YOLOv11 [32], enhances this foundation with improvements in both accuracy and speed. However, these models cannot efficiently add new object classes without a complete re-training of the network, which is extremely time-consuming.

However, to the best of our knowledge, as of the time of writing, no research has explored the integration of incremental learning configurations into YOLOv11, which is critical for enabling interactions with in-situ labels for applications like language learning, where learners continuously encounter new objects and vocabulary [4]. By integrating incremental learning with YOLOv11, we

can dynamically add new objects both for immediate interaction and recall in future contexts. Our application uses replay methods to retrain the YOLOv11 model incrementally (i.e., upon real-time requests from our semi-automatic labeling), ensuring robust performance across all learned classes.

2.3 Image Labeling and Annotation Technologies

As artificial intelligence continues to progress, the need for extensively labeled datasets has risen dramatically. Labeling data in different formats, including text, images, and videos, is a challenging task across all fields. Various studies suggest using manual annotation tools such as ImageTagger [18], BRIMA [34], VIA [17] and LabelImg [61]. Nonetheless, these tools demand a significant amount of human labor and are frequently vulnerable to various mistakes, such as inconsistencies in labeling due to different annotators. Some researchers employed semi-automatic tools that use various algorithms for the annotation of new images, such as iVAT [8], ByLabel [49], and techniques based on machine learning, such as SAM [41, 57, 47]. However, these methods often require well-trained models (e.g., deep learning models) or highly accurate algorithms, along with a substantial amount of processing power. In this paper, we address these issues with a novel method for one-frame automatic annotation.

2.4 Use of Object Detection in Augmented and Mixed Reality Learning and Training

Object detection has often been proposed as an enabling technology for many AR and MR platforms in the fields of manufacturing, education, training, repair, and others. For example, Abdi et al. proposed the use of computer vision for detecting road signs for the purpose of providing supportive augmentations to improve driver safety [1]. Fuchs et al. used a similar approach to track dietary activities to support health by detecting foods and providing MR-based interventions [19]. Advani also used a similar food-based detection strategy, but their application of the technology was to provide grocery assistance for blind or visually impaired people [3]. Many other object-based detection strategies have been built for AR and MR [21, 62, 44], but they all rely on object detection frameworks that have a fixed or limited number of classes available for detection.

Furthermore, MR technologies have transformed learning experiences by providing immersive, interactive environments that engage learners in ways traditional methods cannot. These technologies enable learners to experience real-world scenarios in a controlled setting [65], enhancing their understanding and retention [43, 27] of new information. In language learning, computer vision can create immersive experiences by recognizing objects in the learner's environment and displaying their names in the target language [10, 16]. Another example is the system developed by Lee et al., a language learning system that can recognize several objects in the environment using computer vision and teach the end-user new vocabulary in a different language [36]. However, new words cannot readily be added to their object detection framework. This contextual learning approach helps learners associate words with real-world objects, improving retention and comprehension [4].

Our work introduces a novel end-to-end object detection system within an augmented reality platform that allows for scalable use of object detection and addition of new classes on-the-fly. It includes both a server-side framework and a client-side application, which we have deployed on the Meta Quest 3. In this system, people can easily create a new dataset by annotating just a single frame. These annotations enable our locational-contextual model to be retrained with newly generated datasets or class labels, allowing end-users to quickly make use of new object classes. The significance of this research lies in demonstrating how augmented reality and computer vision can be effectively applied to real-world scenarios in practice.

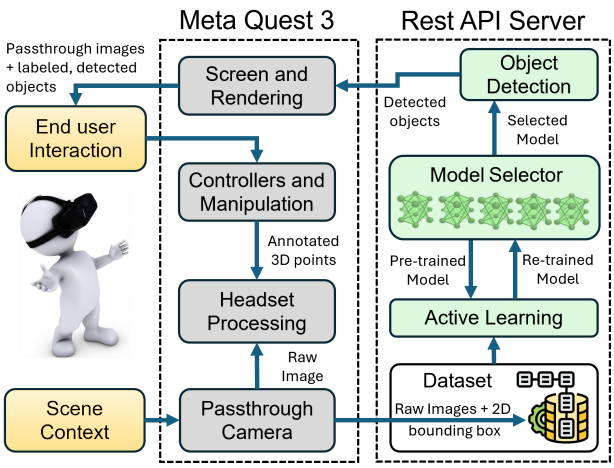

Figure 2: Flow diagram of our proposed end-to-end interaction and retraining framework, including human immersive interaction (left), components, software, and processing for the MR headset (middle), and the server-side component, which is responsible for object detection, model selection, and active incremental learning (right). This system enables real-time object detection and classification while supporting model updates with newly annotated data.

3 Framework and Methodology

In this section, we describe the hardware and end-to-end software architecture we built, including 1) the one-shot labeling process, 2) automated acquisition of an image dataset for a target object, 3) on-the-fly re-training of the detection model, i.e. incremental learning, and 4) details of the design and workflow that enables perceptually seamless interaction with newly classified objects.

Figure 2 illustrates the software architecture, which resides on two main hardware components, including a local server to process the object detection results and active learning algorithm, and the *Meta Quest 3* to handle interaction with target objects and rendering of indicators and labels. The server-side components, developed in Python, handles REST API requests using the FastAPI framework. It also manages model training and inference, leveraging the Ultralytics library[1] to train the YOLOv11 model and execute the inference engine in a subprocess. In this section, we provide a detailed explanation of the functionality of each component.

3.1 One-shot Annotation and Automated Dataset Acquisition

Our first goal was to design an interactive interface that enables users to annotate real-world objects in a physical environment using the MR display's passthrough mode. The primarily functional requirement of this system is for the user to assign a rectangular solid bounding box that accurately encompasses a majority of the voxels of the target. This 3D information is used later to compute 2D bounds from different viewpoints that are fed back into the retraining algorithm for the detection model. This process is outlined in Figure 3, which shows an example of the boxing and labeling, as well as two sample images acquired as part of the dataset retraining.

3.1.1 Labeling Interactions

To enable fast bounding and labeling of different objects, we designed two labeling techniques to allow for flexibility when labeling objects of different sizes and distances.

[1] https://github.com/ultralytics/ultralytics

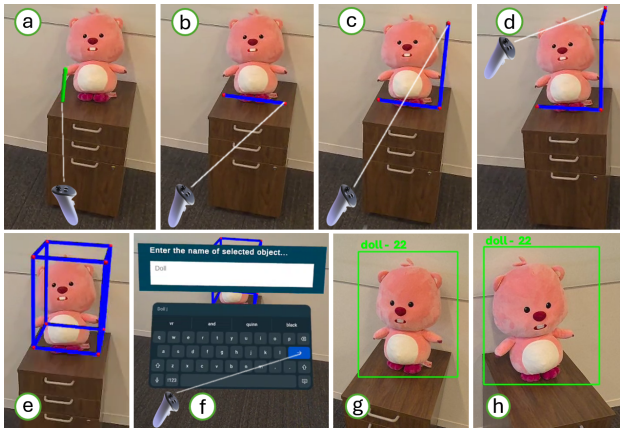

Figure 3: Overview of the annotation process, including a-d) the user drawing a 3D bounding box to specify the volume containing the object, e) the finalized bounding box, f) entering the annotation to send to the machine learning server (also possible by voice input), and g-h) sample images of the resulting recognition results passed to the server for training. (Note that the 2D bounding boxes are only rendered for reference).

Multi-point Bounding. The process of placing a bounding box involves setting the vertices necessary to define its boundaries in 3D. In our first implementation, the user is able to select points in 3D space using the collision of a ray cast against the mesh data generated by the Depth API from their surroundings, as shown in the top row, a-e), of Figure 4.

After setting the initial vertex, the user then selects a second point, simultaneously determining an axis perpendicular to the ground and the width of the bounding box. Then, the user sequentially sets a third point to determine the height and a fourth point to simultaneously determine the depth and finalize the position of the box. This method allows for precise control of the boundaries and is ideal for smaller objects at close distances.

Point and Scale. Our second implementation allows the user to select a position in a 3D space using the intersection of the two rays projected outwards from each controller. Using these rays (i.e. the direction vectors) we compute the point of intersection between the two lines originating from the controllers, which allows us to select any point in 3D space (with the point visualized by a small 3D sphere), as shown in the bottom row, f-i), of Figure 4.

Using this intersection, which is also filtered for stability, the user places the small sphere at a desired center point for the bounding box and then anchors that point with a press of the index trigger on the left controller. After that, they can then press and hold the same trigger to scale one corner of the bounding box to a desired size using the same intersection of the two rays, with the other seven corners simultaneously mirroring its offset from the center across all axes. Upon finalizing the size of the bounding box with another trigger press, the user can rotate the box along its vertical axis to precisely match the orientation of the target object.

By adopting the second approach to annotate objects in the 3D environment, we eliminate the need for a depth API and the calculation of an environment mesh. This results in faster and more practical annotations for objects that are labeled at a distance or for headsets that may not provide direct access to the environment mesh. This also allows for easy labeling of objects that are at a distance, partially occluded, or out of reach.

During testing, one of the first limitations we noticed with the bounding technique was that intersections of the controller ray with far-away mesh points were difficult to control with high precision, especially when an edge of the bounding box was parallel to the

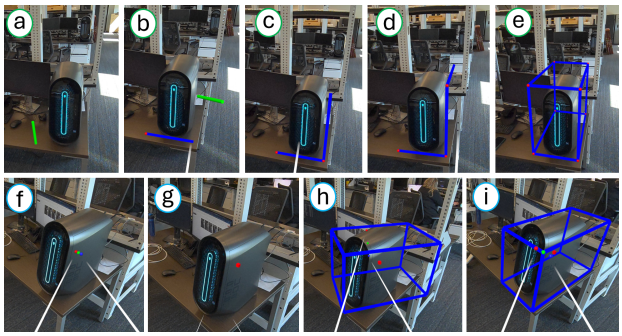

Figure 4: Images of the two different selection methods we implemented for drawing object bounds. The first method (top row) allows for sequential selection of points that form the bounding box in 3D space using the absolute position of the controller. The user a) sets an initial point, b) sets another point to determine the width of their selection boundary, c) sets a third point, constrained to a single vertical axis, to determine height, and d) the final point, constrained to a horizontal axis, to determine depth. The second selection method (bottom row) in which the user controls a 3D cursor f) using the intersection of two rays in 3D space to g) set the center point for their selection boundary and h) drag one corner of the boundary box to expand it, and i) rotate the boundary box's final position by rotating the controller.

user's gaze point. This resulted in the point-and-scale technique, in which bounding box edges are not dependent on intersection with a single axis. Secondly, the addition of a rotation to the point-and-scale was added after we realized that users had no way to specify the bounding boxes' orientation without an additional rotation operation. Lastly, the scene mesh provided by a depth camera or simultaneous localization and mapping (SLAM) can have inconsistencies for complex objects, so our point-and-scale technique further reduced reliance on the mesh.

3.1.2 Automated Dataset Acquisition

Once the four points are identified, our system generates a transparent 3D cube that encloses the annotated object (Figure 3 e). We compute the projected positions of the cube's eight corners by transforming their 3D coordinates into UV coordinates in 2D space, factoring in the camera's 3D perspective at the moment of manual labeling. This transformation ensures that people can move around the object and view it from different angles while maintaining accurate alignment.

Next, our system calculates a 2D bounding box that tightly encloses the projected corners and captures an image of the object in real-time (Figure 3 g and h). A key advantage of this approach is that a single annotation in 3D space automatically generates multiple, accurate 2D bounding boxes of annotated images from different viewpoints, facilitating benchmark dataset creation with minimal manual effort. Our final saved 2D images have a resolution of 1280×960 pixels, and we account for the perspective distortion introduced by this resolution in our calculations.

3.2 Active Incremental Learning Framework

In order to incorporate new data or classes into an already trained model, we had to carefully design an active learning approach that could maintain high enough accuracy to be usable for AR and MR applications while being efficient enough for the model to undergo retraining in a short amount of time. Most network retraining approaches are either extremely time-consuming or result in a decrease in accuracy in other classes.

Simply put, our strategy is to freeze a number of layers in the network to preserve previously trained classes and to re-train the head such that new classes can be added with minimal retraining time. To both efficiently retrain a usable model and deploy that model on a local PC, we developed a REST-API framework to manage all necessary processes and facilitate communication with MR headsets. This framework is capable of receiving images and sending predicted results, as well as accepting new datasets and training the corresponding locational-contextual models.

In the retraining phase, we use the previously trained weights, which were trained on existing classes, as our pretrained weights. We freeze the first 23 layers of YOLOv11s and then proceed with transfer learning. For each retraining session, we retain 100 instances (based on our GPU capacity and desired training time) of the old classes as replay exemplars along with the new class for the training and validation sets. Though more training would result in higher accuracy, logical testing revealed that these settings were sufficient when combined with our object detection approach, which aggregates results over multiple frames. We set the patience parameter to 15 to prevent overfitting and ensure we save the best-trained weights. Since we use a pretrained model that was initially trained on existing classes, the model tends to focus more on new classes, which often results in it stopping training sooner than scheduled. This approach ensures a short and limited training time of approximately 30 minutes on our NVIDIA 3080 GPU.

To improve the generality of our image dataset, we make use of the built-in mosaic image and augmentation techniques that are part of the YOLO framework, such as cropping, flipping, scaling, and copy-pasting, to place objects in different backgrounds. This helps reduce background distractions and environmental dependencies. We set the related parameters ($mosaic = 0.9, copy\_paste = 0.9$) to high values to maximize the use of these capabilities, ensuring that the model can accurately predict the desired objects in diverse environments.

We tested this approach multiple times and achieved accuracy levels comparable to the baselines (detailed results shown in Section 4.1). Training a model with a limited number of images for new classes can improve detection accuracy for the specific objects annotated by the user. This approach allows us to reduce the generality of the model, resulting in fewer false detections. Also, using the aggregated detection result from multiple images (Section 3.3.2), we can ensure that we display accurate results to the user.

However, given the base model's constrained ability to handle different classes, particularly if it was trained on a limited dataset and struggles with unseen objects or environmental variations, it may be necessary to use distinct models tailored to each location and context based on the environment in which the headset operates. To address this, we implemented a model selector feature that utilizes multiple trained models tailored for different contexts and positions of the AR headset. For example, an equipment room will likely have static equipment that only needs to be added during installation or replacement. By adopting this method, the model can operate with fewer classes per context. Consequently, the reduced need for training images ultimately leads to improved accuracy with a shorter training period while mitigating model capacity issues. Additionally, in the application context of second language learning, we can dynamically remove classes that learners have already mastered, allowing for a more targeted and efficient learning experience.

3.3 Object Detection System

Visualizing detected objects in 3D space is more challenging than in 2D space [44, 62]. The six degrees of freedom offered by augmented reality headsets add to this complexity. Rapid movements can alter the scene in front of the camera, leading to the misplacement of detected objects due to detection latency. In this paper, we use various techniques to transform YOLOv11's 2D UV coordinate locations into 3D space, using an aggregate of points projected onto the environment mesh to determine depth, which helps us track detected objects, reduce false positives, and address latency issues.

3.3.1 UV to 3D Conversion

To correctly compute 3D world locations, we have to convert 2D UV coordinates obtained from a camera image into corresponding 3D world points, while compensating for head movements. The core idea is to map points detected in a webcam texture onto real-world positions using a raycasting approach in a Unity environment. To accurately locate detected objects while minimizing latency- and motion-related effects, we buffer the complete `cameraPoseInWorld` at the time of capture of the snapshot from the front camera. This image is sent to the server, which runs the image through the current object detection model and returns detection results in 2D UV coordinate space, which includes: (1) object labels (text), (2) bounding box corner points (four points per object), and (3) confidence scores.

These results are then passed back to the headset via the network for further processing and 3D visualization. Once the results arrive in the headset's coordinate system, we convert detection results containing 2D pixel coordinates (UV) that represent the corners of a detected object via the following steps. For each detection, we perform:

- **UV Adjustment:** The code converts raw pixel values into UV coordinates, adjusting the Y-axis to accommodate Unity's inverted coordinate system.
- **Raycasting:** Each adjusted UV point is transformed into a ray in world space by using the buffered camera's pose. The helper function computes a ray in the camera frame and then converts it using the camera's position and orientation. The environment raycast manager then tests these rays to determine intersections with real-world surfaces.
- **Detection Processing:** If all four corners of the detection yield valid raycast intersections in that frame, the corresponding 3D world points are recorded for further processing. Otherwise, we wait for valid intersections in subsequent frames before determining a 3D world point.

After converting the UV coordinates:

- **Integrity Checks:** The implementation verifies that none of the computed 3D points contain invalid values (NaN or infinity). This is critical to ensure the reliability of the mapping.
- **Geometric Calculations:** The center of the quadrilateral defined by the corners is calculated by averaging the 3D corner points. The average distance from the center to each corner is also computed, providing a measure of the detection's spatial extent to remove double detections and overlapping.

This conversion approach is essential for applications in augmented reality and object tracking. Precise spatial localization directly impacts system performance and user experience. The results are used by the object tracker to consolidate outcomes, handle false positives, and display anchors.

3.3.2 Object Tracker Mechanism

As shown in Figure 5, in order to effectively display detected objects, it is crucial to minimize flickering labels associated with each detection per frame, eliminate duplicate detections, and address false positives. To achieve a robust tracking system, it is necessary to monitor various attributes of each detected object, including its

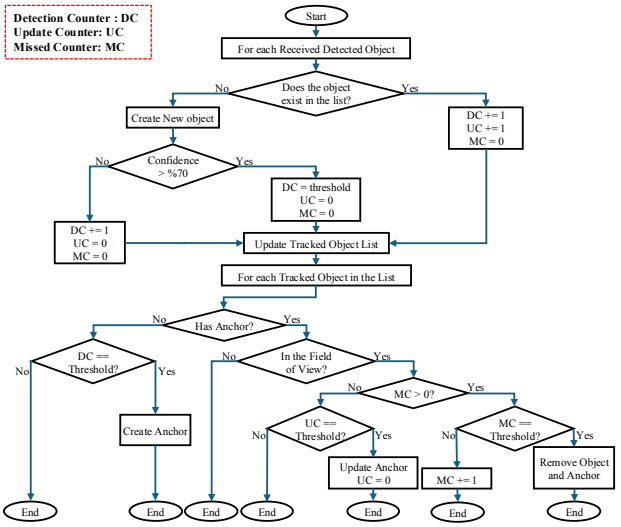

Figure 5: Flow diagram showing the process of tracking detected objects through multiple frames based on detection results.

location, size, name, and confidence score. Confidence scores from YOLO detections are retained to assess the reliability of the displayed objects in the tracking system. The tracker prioritizes high-confidence detections for rendering, ensuring accurate and relevant visual feedback.

Our implementation is similar to that of Huynh et al. [28], incorporating confidence scores and adapting to accommodate newly added classes to the detector. It integrates with a 3D conversion module to maintain and update the state of detected objects over time. We define three counters: the detection counter, the miss counter, and the update counter.

- The detection counter increases when an identified object with the same label and location is detected per frame, which helps verify its certainty prior to rendering.
- The miss counter keeps track of instances of existing objects that are not detected within the field of view. If an object is not detected after a specified number of misses, it will be removed from the screen.
- The update counter adjusts the center location of an object after receiving a specific number of new location updates. However, the new location must be within the object's area; otherwise, we consider it a new object.

When detection results are calculated in 3D space (i.e., VR headset), the tracker searches for matching objects using both the object's label and its spatial proximity. This helps determine whether an existing tracked object should be updated. We consider a distance threshold based on the calculated object size to avoid locating two identical objects inside each other. If a match is found within a calculated distance threshold, the system resets the miss counter and increments both detection and update counts. Once a sufficient number of updates have accumulated, the object's position and associated anchor are refreshed.

If no matching object is found, a new tracked object is instantiated with an initial detection count that varies based on its confidence score. The tracker also monitors objects that move out of people's field of view, incrementing a miss counter for each undetected object and subsequently removing those that exceed a predefined miss threshold. Additionally, people have the option to manually clear all tracked objects using a designated controller input. This dynamic tracking mechanism, along with configurable thresholds for object creation, update, and removal, ensures that the system reliably maintains accurate, real-time spatial representations of objects in the headset virtual environment.

4 System Evaluation

With any label-based system used for training or education, one primary concern is the time taken to add new classes to a detection framework such that they are practically usable. Though our system allows for immediate use of a label once placed, we primarily sought to evaluate how long it takes for our re-training algorithm to produce useful results in the event that an object changes locations.

4.1 Active Incremental Learning Training Times

To evaluate the effectiveness of our approach, we first compared how long it would take to add new individual classes to a pre-trained model, both with our incremental learning framework and with currently available methods for re-training. In other words, we needed to determine how quickly new classes would be usable to an end-user's MR application for each method.

To do so, we conducted a series of experiments using YOLOv11, with our proposed active incremental learning based on a replay method on transfer learning. For this test, the first 23 layers of the model were frozen during training to retain low-level feature representations. Initially, the model was trained on four object classes—*Person*, *Chair*, *Monitor*, and *Table*—using official pre-trained YOLOv11 weights. In subsequent stages, new classes were added incrementally while continuing training from the previously learned weights. The times taken to train the initial four classes (simultaneously) as well as the times taken to add each individual class are shown in the top chart of Figure 6, which are labeled respectively.

For comparison, we conducted the same test without the first 23 layers frozen. In other words, we measured the time required to train the initial four classes as well as each additional class when all layers must be re-trained to add the new classes, a common approach in current detection systems. These results are shown at the bottom of Figure 6, which highlight the significant time (50+ epochs) necessary even to add single classes in a traditional object detection workflow. In comparison, our approach only takes between 3 and 20 epochs, significantly reducing the time it takes before added classes are practically usable by the detector for the purposes of generating a label automatically.

4.2 Analysis of Accuracy by Class Addition

Secondly, we sought to demonstrate 1) the relative time scales necessary for training the initial set of four images from our contextual model, 2) subsequent training times required for adding individual classes, and 3) the relative classification performance, measured in mAP, to demonstrate that the detection accuracy of previously trained classes is still preserved without catastrophic forgetting.

More simply, the key objective of this test is to demonstrate that adding new classes incrementally can be done in comparatively little time without degrading the performance of the already-learned classes. Results are shown in Table 1, which demonstrates that the accuracy of the initial classes remains relatively stable across incremental stages, highlighting the strength of our approach in preserving knowledge and effectively mitigating the catastrophic forgetting problem. All models were set to train up to 500 epochs.

Although the training schedule is fixed, in our use cases, training typically involves early stopping based on a patience threshold of 15 epochs due to the low number of images and using a previously trained model on existing classes as pretrained weights to initiate the training process. Because we have the 3D positions of the objects in world-space, we can aggregate data over frames despite

Table 1: Performance results of our proposed incremental learning method using YOLOv11. The table reports the mean Average Precision (mAP) at IoU thresholds ranging from 0.50 to 0.95 (%) across different object categories. Each stage of training involves incrementally adding new object classes while preserving the detection performance of previously learned ones. All the training requires a patience of 15 epochs to determine an optimal stop point, i.e., when loss no longer decreases and precision (mAP) no longer increases on the validation dataset, to avoid overfitting. The results are evaluated on the test split of the PASCAL VOC validation dataset. The number of test images per class is as follows: Person (2232), Chair (642), Monitor (296), Table (323), Car (608), Dog (661), Train (275), Airplane (348), Sofa (336), and Bottle (369), with a total of 4195 images. Training time (in minutes) is reported to reflect the addition of each new class. An extension of this table that includes the results of 27 trained classes is included as supplementary material.

Train Time (min)	Epochs (Optimal Stopping)	mAP-All %	Person %	Chair %	Monitor %	Table %	Car %	Dog %	Train %	Airplane %	Sofa %	Bottle %
81	500*	54.7	39.3	48.2	69.5	61.7	-	-	-	-	-	-
23	240	58.3	56.2	43.0	71.5	62.7	-	-	-	-	-	-
18	113	61.2	55.5	45.4	71.8	64.8	**68.3**	-	-	-	-	-
21	105	64.5	52.8	50.2	71.3	64.2	69.7	**78.6**	-	-	-	-
28	132	65.9	49.8	47.5	74.2	62.8	69.4	77.5	**80.1**	-	-	-
29	122	69.0	52.6	50.9	73.8	64.8	69.9	79.8	81.4	**78.5**	-	-
18	63	67.3	48.7	51.1	71.8	65.1	69.4	75.4	81.3	79.1	**63.3**	-
29	115	66.3	48.8	50.2	72.3	63.8	69.9	73.5	81.7	79.6	66.9	**56.2**

*As a baseline, the layers for initial training were not frozen during the first 500 epochs, and this training was not stopped early.

lower recognition rates. For testing, we set the minimum average detection threshold at 70% accuracy, as shown in Figure 6.

As a result, most training sessions terminate in less than 30 minutes, making the approach suitable for low-latency applications such as MR-based object detection. For this benchmark evaluation, we used the PASCAL VOC validation dataset, selecting 100 images per class group for training and 100 for validation, while the remaining samples were reserved for testing. We limit the number of images to simulate our dataset generation deployment via the Meta Quest 3 and use a replay method with fixed exemplar size.

This fixed sample size setup serves two main purposes: 1) In the early stages, only the head of the model is trained, requiring fewer data samples. 2) In AR-based object detection scenarios, minimizing training time and the effort required for data collection to enhance usability and scalability.

Additionally, we adopt a dynamic class management strategy in our language learning use case, where object recognition assists vocabulary acquisition. Once a user has sufficiently learned a word (object), its class can be forgotten in the model, ensuring each contextual-locational model maintains an appropriate number of active classes (typically 10) for efficient processing.

The test dataset consisted of 4195 images, with per-class distribution as follows: Person (2232), Chair (642), Monitor (296), Table (323), Car (608), Dog (661), Train (275), Airplane (348), Sofa (336), and Bottle (369). As shown in Table 1, our method retains consistent high performance even as additional classes are incrementally added. Even after adding 17 new classes to the last trained model of Table 1 (on top of the initial 10 classes), we only observed a decrease in mAP for the previous classes as follows: Person -1.1%, Chair -8.4%, Monitor -6.2%, Table -7.8%, Car -4.7%, Dog -15.8%, Train -3.7%, Airplane -5.3%, Sofa -6.9%, Bottle -2.6%. Though some loss of accuracy is expected, the reductions are small enough that our object recognition approach can still handle aggregation of multiple frames of data without major issues. This confirms the effectiveness of our transfer learning and replay-based incremental learning strategy for real-world, scalable, and user-friendly MR applications.

4.3 Dataset Generation and End-to-end Latency

Next, we sought to provide general metrics as to the times required for labeling frames. We calculated the average total time for processing and uploading annotated frames to be approximately 13 seconds, which includes both the capture and annotation of the frames. The end-to-end process includes the capture and automated annotation of 100 frames after the one-shot label, which takes roughly 12 seconds, resulting in an average of 120 milliseconds per frame. Additionally, the upload time, which refers to transmitting the annotated frames to the server, took approximately 1.5 seconds. For reference, the average time required to annotate a single image across three different methods, SAMAL [47], manual annotation [47], and our proposed approach, is shown in Table 2.

SAMAL requires 435 ms per image, whereas manual annotation is significantly slower at roughly 9.4 seconds (72 minutes for all 459) per image. In contrast, our method significantly reduces the annotation time to 120 ms per image. This comparison highlights both the inefficiency of manual annotation and the notable improvements achieved by our approach over existing automated solutions like SAMAL. A comprehensive breakdown of the system detection latency and annotation time is provided in Table 2.

Table 2: Annotation Performance Comparison.

Per-Image Annotation Time	Time (ms)
SAMAL [47]	435
Manual Annotation [47]	9,411
Our Approach	**130**

To evaluate the responsiveness of our system, we measured the end-to-end latency from image capture on the Meta Quest 3 device to the final visualization of detection results in the augmented reality environment. The system comprises two key latency components: server-side processing and client-side (Meta Quest 3) processing. On the server side, the latency was measured from the moment a request (image) is received to the moment the processed detection results are sent back. The average latency recorded on the server, running YOLOv11 object detection, was approximately 40 milliseconds. On the client side, the total latency includes image capture, transmission to the server, waiting for server response, and rendering of results in the AR interface.

The overall average latency measured on the Meta Quest 3 device was approximately 200 milliseconds. This includes network communication time (100 ms), server processing delay (40 ms), and local post-processing tasks (60 ms) such as 3D visualization and spatial placement of detected objects. The results show that while server-side processing adds only a small amount to the total latency, client-side tasks—such as transmission, reception, and augmented reality visualization—contribute significantly to delays.

We developed end-to-end asynchronous processing to minimize the effects of latency on the main thread. Implementing optimization strategies at both the communication and rendering levels could

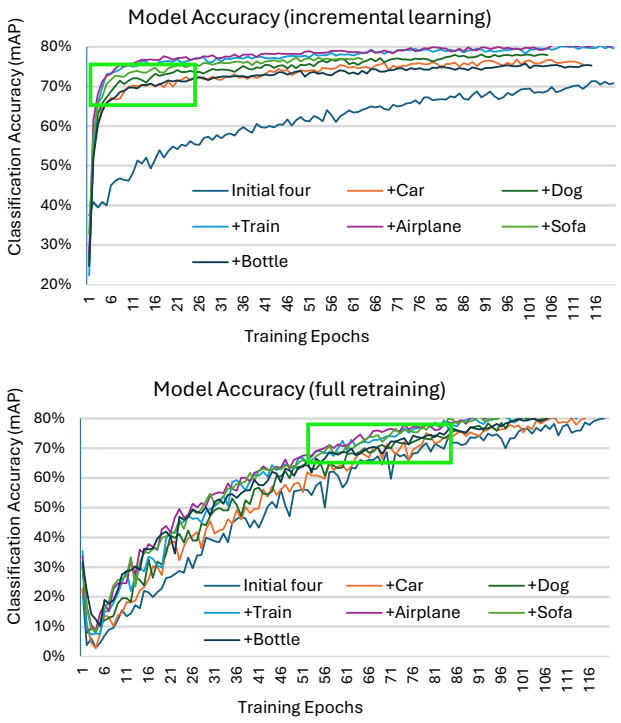

Figure 6: Graphs showing the time necessary to achieve 70% accuracy when adding an additional class to the YOLOv11 model. The top graph shows our approach, with the green box highlighting the maximum number of epochs needed to add an individual class and retrain the model, roughly 3 to 20 epochs. The bottom graph shows traditional training (without any frozen layers), which takes between 50 and 90 epochs to reach similar accuracy.

further reduce the overall latency of the system. Despite some end-to-end system latency, end-users would not likely experience any perceptible delay in the rendering of newly detected objects once they are recognized by the model. Since the Meta Quest 3's latency is extremely low for rendering anchored, world-relative content, affixed labels are also perceived as near-instantaneous (i.e. no jitter) when displayed in-situ.

4.4 Practical Detection Testing and Implementation of a Language Learning Application

Finally, we wanted to make sure that our system worked well in practice, specifically by testing it in multiple environments with different lighting conditions and varied object orientations. Several of these environments are shown in Figure 7, which visually demonstrates the versatility of both the underlying object detection and our 3D object positioning algorithm. While some labels are still erroneous, these images show that we can seamlessly detect and display object labels. Moreover, the system continuously tracks the existing and incoming results to reduce duplicate detections, missed information, and false positives.

Secondly, we implemented an in-situ second language learning system that allows for real-time addition of new words (objects) for vocabulary learning. The top row of Figure 7 shows the corresponding 3D labels in both a target and a destination language, enabling in-situ vocabulary study for the user. The addition of new objects to the classifier and subsequent rendering in real environments are highlighted in our supplementary video. By incorporating a second language acquisition scenario into our augmented reality system, we demonstrate the benefits of on-demand labeling and real-time interaction with a learner's surroundings.

Additionally, to evaluate the stability of our proposed method in detecting more objects with the same trained model, we tested our approach using 27 different classes, the results of which are shown in Figure 8. The model selector we developed can also be used to train a context- or location-specific model, which effectively minimizes training time while maximizing accuracy.

5 Discussion

In practical testing with our language learning application (see the video in supplementary detail for visual results), we learned much about the potential use cases for this framework as well as other opportunities for exploration.

5.1 Design Considerations

One of the strengths of our design is the ability to immediately use new annotations in-situ for the purposes of learning or instruction. Especially in the field of manufacturing, businesses often struggle with the inability to use object detection frameworks such as Yolo because they are trained on such a limited dataset of everyday objects. Our framework helps address this problem by making new classes accessible with relatively little interaction.

However, there are still some scenarios in which we need to employ a different strategy for training. For example, when the image dataset used for retraining is passed to the rest API, there is a small window of time (on the order of 3-20 epochs, about 10-25 minutes on a pc with an NVIDIA RTX3080) when the object cannot be tracked if it is moved to a new location. In other words, if the end user moved the object immediately after labeling it, the system would take some time before the object in the new location is recognized. As a potential solution, one strategy would be to have a phased approach to the training, where new classes are trained using as small or "mini" model that retrains in a matter of seconds to handle recently moved objects, incremental learning for the mid term, and a complete re-training process over the course of several days that fine tune the frozen layers of the head. Optical flow may also be able to track objects for a few minutes until the optimized model can add the new object.

Despite some loss of accuracy in older classes due to catastrophic forgetting in the replay method, our method still effectively detects previous classes. This is accomplished through our proposed strategies within the object tracker mechanism, which concatenates the results of detected objects across multiple frames. Additionally, to reduce the number of training images in a dataset, it may be beneficial to use more complex strategies for selecting an exemplar, such as random selection, herding [51], entropy-based selection [5], and influence-based selection [60]. Another strategy is to slowly phase out older classes that are unused or no longer needed, for example, classes that correspond to already learned words or concepts in learning applications.

5.1.1 Other Applications and Virtual Assistants

In addition to language learning, scalable object detection has tremendous potential for use in manufacturing and training applications. The manufacturing industry, in particular, faces significant hurdles with training new employees due to high turnover every year. Because new or custom machinery is not generally available in object recognition frameworks, instruction is not scalable to shop floor or assembly line work without sufficient retraining. To interact with instructions, manuals, or training materials for new equipment, managers or trainers can easily add these new objects to a detection model, which provides immediate access to training materials for new employees.

One other potential way to further enhance our framework is by integrating large language models (LLMs). Integration with LLMs

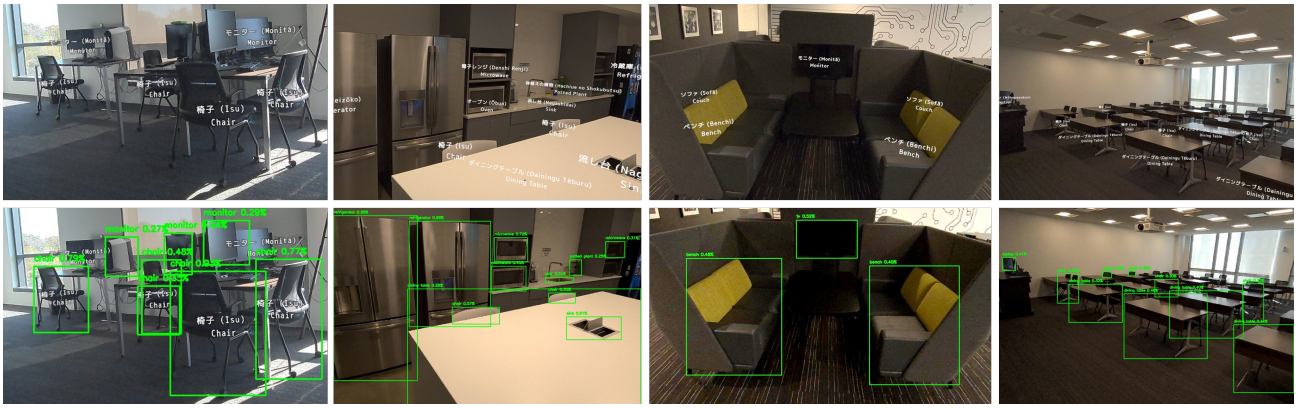

Figure 7: Demonstration of detection results (bottom row) and corresponding 3D-positioned labels for our language learning application (top row) using YOLOv11 pretrained on the COCO dataset. The positioning and density of the labels demonstrate that our custom detection workflow not only avoids duplicates and false positives but provides adequate 3D positioning in world-space.

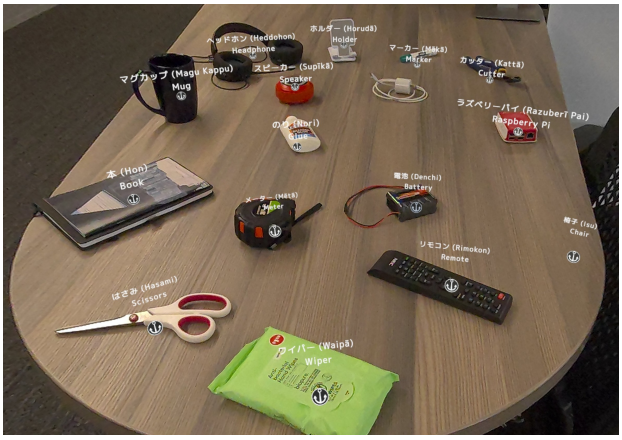

Figure 8: Detection results after training the model on 27 different classes, demonstrating the scalability of the approach to a larger number and more diverse set of items, including those not present in the pretrained Yolov11 dataset.

can contextualize labeled objects, allowing for interactive captions in addition to direct labeling. This, in combination with LLMs that specialize in multiple languages or a specific context like manufacturing, could generate rich, interactive content that is specific to the end-user's application. For example, Lee et al. implemented an AR system that combines basic object detection and generic LLMs. Their paper goes on to show that there is promise in AR-based language learning immersion systems, but still has yet to implement a customized virtual assistant that can dynamically provide language-specific lessons to objects in-situ [35].

LLM-powered virtual assistants that make use of spatial object anchors and are tailored to the application context via retrieval augmented generation (RAG) [66] frameworks would allow for more personalized and immersive learning experiences, leading to more effective learning platforms that reduce the burden of manually creating personalized learning experiences.

5.2 Future Work

For future work, we plan to enhance our current methodology and explore new ideas to develop a more robust and generalizable solution. Specifically, we aim to deploy the inference engine directly on the Meta Quest device, incorporate SAM2 [50] for object annotation in segmentation tasks, and address occlusion issues [59] both to minimize the inclusion of occluded images as training data and to make label visualization more realistic. Additionally, we intend to support multiple annotation formats as output, integrate object tracking for constantly moving targets, and adopt socket-based communication methods (such as ZeroMQ [24]) to enable faster and more efficient data transmission.

We still have several other minor limitations with our approach. For example, if an end-user moves a physical object immediately after completing the label and dataset acquisition, there will be a small amount of time when the object is not re-acquired (re-detected) while the retraining algorithm runs in the background. We plan to work on further optimization or temporary object motion tracking as a potential solution to this issue.

6 Conclusion

In this paper, we introduced a new method for scalable object detection that enables the practical detection and labeling of objects in AR/MR applications. To accomplish this, we made use of one-shot labeling in conjunction with an end-to-end active learning framework that allows for automatic dataset creation and on-the-fly retraining of object detection models. By significantly reducing the overhead required to train image classifiers, this framework can help enable a wide array of applications dependent on object detection such as language learning, procedural instruction, and assembly guidance, while ensuring that newly labeled objects can be accurately classified without compromising the accuracy of previously learned objects.

We hope that this work and the scalability of the approach will be a significant step forward in the adoption of MR applications.

Acknowledgments

This work was funded in part by the National Science Foundation, grant number 2223035.

References

[1] L. Abdi and A. Meddeb. Deep learning traffic sign detection, recognition and augmentation. In *Proceedings of the Symposium on Applied Computing*, pp. 131–136, 2017. 1, 3

[2] H. Abu Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother. Augmented reality meets computer vision: Efficient data generation for urban driving scenes. *International Journal of Computer Vision*, 126:961–972, 2018. 1

[3] S. Advani, P. Zientara, N. Shukla, I. Okafor, K. Irick, J. Sampson, S. Datta, and V. Narayanan. A multitask grocery assist system for the visually impaired: Smart glasses, gloves, and shopping carts provide auditory and tactile feedback. *IEEE Consumer Electronics Magazine*, 6(1):73–81, 2016. 3

[4] G. C. Agbo and P. A. Agbo. The role of computer vision in the development of knowledge-based systems for teaching and learning of English language education. *ACCENTS Transactions on Image Processing and Computer Vision*, 6(19):42–47, May 2020. 2, 3

[5] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio. Gradient based sample selection for online continual learning. *Advances in Neural Information Processing Systems*, 32, 2019. 8

[6] J. Bang, H. Kim, Y. Yoo, J.-W. Ha, and J. Choi. Rainbow memory: Continual learning with a memory of diverse samples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8218–8227, 2021. 2

[7] S. A. Bello, S. Yu, C. Wang, J. M. Adam, and J. Li. Deep learning on 3D point clouds. *Remote Sensing*, 12(11):1729, 2020. 1

[8] S. Bianco, G. Ciocca, P. Napoletano, and R. Schettini. An interactive tool for manual, semi-automatic and automatic video annotation. *Computer Vision and Image Understanding*, 131:88–99, 2015. 3

[9] M. Biesialska, K. Biesialska, and M. R. Costa-jussà. Continual lifelong learning in natural language processing: A survey. In D. Scott, N. Bel, and C. Zong, eds., *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6523–6541. International Committee on Computational Linguistics, Barcelona, Spain (Online), Dec. 2020. doi: 10.18653/v1/2020.coling-main.574 2

[10] A. Caetano, A. Lawson, Y. Liu, and M. Sra. ARLang: An outdoor augmented reality application for portuguese vocabulary learning. In *Proceedings of the ACM Designing Interactive Systems Conference*, DIS '23, p. 1224–1235. ACM, July 2023. doi: 10.1145/3563657.3596090 3

[11] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision*, ECCV 2018, pp. 532–547, 2018. 2

[12] Z. Chen and B. Liu. *Lifelong Machine Learning*. Morgan & Claypool Publishers, 2018. 2

[13] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, 2021. 2

[14] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 2

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848 2

[16] M. Dunleavy, C. Dede, and R. Mitchell. Affordances and limitations of immersive participatory augmented reality simulations for teaching and learning. *Journal of science Education and Technology*, 18:7–22, 2009. 3

[17] A. Dutta and A. Zisserman. The VIA annotation software for images, audio and video. In *Proceedings of the ACM International Conference on Multimedia*, pp. 2276–2279, 2019. 3

[18] N. Fiedler, M. Bestmann, and N. Hendrich. ImageTagger: An open source online platform for collaborative image labeling. In *Proceedings of the RoboCup 2018: Robot World Cup XXII 22*, pp. 162–169. Springer, 2019. doi: 10.1007/978-3-030-27544-0_13 3

[19] K. Fuchs, M. Haldimann, T. Grundmann, and E. Fleisch. Supporting food choices in the internet of people: Automatic detection of diet-related activities and display of real-time interventions via mixed reality headsets. *Future Generation Computer Systems*, 113:343–362, 2020. 3

[20] J. a. Gama, I. Žliobaitundefined, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, 46(4):44:1–44:37, Apr. 2014. doi: 10.1145/2523813 2

[21] Y. Ghasemi, H. Jeong, S. H. Choi, K.-B. Park, and J. Y. Lee. Deep learning-based object detection in augmented reality: A systematic review. *Computers in Industry*, 139:103661:1–103661:15, 2022. doi: 10.1016/j.compind.2022.103661 3

[22] L. Golab and M. T. Özsu. Issues in data stream management. *ACM Sigmod Record*, 32(2):5–14, Mar. 2003. 2

[23] H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet. A survey on ensemble learning for data stream classification. *ACM Computing Surveys (CSUR)*, 50(2):23:1–23:36, 2017. 2

[24] P. Hintjens. *ZeroMQ: Messaging for Many Applications*. O'Reilly Media, Inc., Sebastopol, CA, 2013. 9

[25] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2

[26] S. Hou, X. Pan, C. C. Loy, Z. Wang, and D. Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 831–839, 2019. 2

[27] H. Hunaepi, A. I. A. R. Sudiatmika, and W. Fadly. Augmented reality in education: A meta-analysis of (quasi-) experimental studies to investigate the impact. *Reflection Journal*, 3(2):74–87, 2023. 3

[28] B. Huynh, J. Orlosky, and T. Höllerer. Semantic labeling and object registration for augmented reality language learning. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 986–987. IEEE, 2019. 6

[29] A. Ibrahim, B. Huynh, J. Downey, T. Höllerer, D. Chun, and J. O'donovan. Arbis pictus: A study of vocabulary learning with augmented reality. *IEEE Transactions on Visualization and Computer Graphics*, 24(11):2867–2874, 2018. 2

[30] A. Karambakhsh, A. Kamel, B. Sheng, P. Li, P. Yang, and D. D. Feng. Deep gesture interaction for augmented anatomy learning. *International Journal of Information Management*, 45:328–336, 2019. 1

[31] R. Kemker, M. McClure, A. Abitino, T. L. Hayes, and C. Kanan. Measuring catastrophic forgetting in neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32, 2018. 2

[32] R. Khanam and M. Hussain. YOLOv11: An overview of the key architectural enhancements. *arXiv preprint arXiv:2410.17725*, Oct. 2024. doi: 10.48550/arXiv.2410.17725 1, 2

[33] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. 2

[34] T. Lahtinen, H. Turtiainen, and A. Costin. Brima: Low-overhead browser-only image annotation tool (preprint). In *Proceedings of the IEEE International Conference on Image Processing*, ICIP 2021, pp. 2633–2637, 2021. doi: 10.1109/ICIP42928.2021.9506683 3

[35] H. Lee, C.-C. Hsia, A. Tsoy, S. Choi, H. Hou, and S. Ni. VisionARy: Exploratory research on contextual language learning using AR glasses with ChatGPT. In *Proceedings of the Biannual Conference of the Italian SIGCHI Chapter*. ACM, New York, 2023. doi: 10.1145/3605390.3605400 9

[36] J. Lee, S. Kim, M. Park, C. L. Rasgaitis, and J. E. Froehlich. Embodied AR language learning through everyday object interactions: A demonstration of EARLL. In *Adjunct Proceedings of the Annual ACM Symposium on User Interface Software and Technology*, UIST Adjunct '24, pp. 52:1–52:3, 2024. doi: 10.1145/3672539.3686746 3

[37] Z. Li and D. Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2017. 2

[38] K. Liu, Y. Yu, Y. Liu, J. Tang, X. Liang, X. Chu, and Z. Zhou. A novel brain-controlled wheelchair combined with computer vision and augmented reality. *Biomedical Engineering Online*, 21:50:1–50:20, 2022. doi: 10.1186/s12938-022-01020-8 1

[39] N. Mahmud, J. Cohen, K. Tsourides, and T. M. Berzin. Computer vision and augmented reality in gastrointestinal endoscopy. *Gastroenterology Report*, 3(3):179–184, 2015. 1

[40] A. Malta, M. Mendes, and T. Farinha. Augmented reality maintenance assistant using YOLOv5. *Applied Sciences*, 11(11):4758, 2021. 1

[41] O. Marques and N. Barman. Semi-automatic semantic annotation of images using machine learning techniques. In *Proceedings of The*

Semantic Web: International Semantic Web Conference, ISWC 2003, pp. 550–565. Springer, Sanibel Island, FL, USA, Oct. 2003. doi: 10.1007/978-3-540-39718-2_35 3

[42] N. Y. Masse, G. D. Grant, and D. J. Freedman. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *Proceedings of the National Academy of Sciences*, 115(44):E10467–E10475, 2018. 2

[43] Z. Merchant, E. T. Goetz, L. Cifuentes, W. Keeney-Kennicutt, and T. J. Davis. Effectiveness of virtual reality-based instruction on students' learning outcomes in k-12 and higher education: A meta-analysis. *Computers & Education*, 70:29–40, 2014. 3

[44] M. M. Nafea, S. Y. Tan, M. A. Jubair, and T. Abd Mustafa. A review of lightweight object detection algorithms for mobile augmented reality. *International Journal of Advanced Computer Science and Applications*, 13(11), 2022. 3, 5

[45] B. Pande, K. Padamwar, S. Bhattacharya, S. Roshan, and M. Bhamare. A review of image annotation tools for object detection. In *Proceedings of the International Conference on Applied Artificial Intelligence and Computing*, ICAAIC 2022, pp. 976–982. IEEE, 2022. doi: 10.1109/ICAAIC53929.2022.9792665 1

[46] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019. 2

[47] Á. Patrício, J. Valente, A. Dehban, I. Cadilha, D. Reis, and R. Ventura. AI-powered augmented reality for satellite assembly, integration and test. In *Proceedings of the IEEE International Conference on Artificial Intelligence and eXtended and Virtual Reality*, AIxVR 2025, p. 9 pages. IEEE, 2025. 3, 7

[48] D. A. Plecher, C. Eichhorn, K. M. Seyam, and G. Klinker. Arsinoë - learning Egyptian hieroglyphs with augmented reality and machine learning. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality Adjunct*, ISMAR 2020 Adjunct, pp. 326–332. IEEE, 2020. doi: 10.1109/ISMAR-Adjunct51615.2020.00092 1

[49] X. Qin, S. He, Z. Zhang, M. Dehghan, and M. Jagersand. ByLabel: A boundary based semi-automatic image annotation tool. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, WACV 2018, pp. 1804–1813. IEEE, 2018. doi: 10.1109/WACV.2018.00200 3

[50] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, et al. SAM 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. 9

[51] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. iCaRL: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, CVPR 2017, pp. 2001–2010, 2017. 8

[52] J. Redmon and A. Farhadi. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, p. 6 pages, 2018. 2

[53] L. Richardson and S. Ruby. *RESTful Web Services*. O'Reilly Media, Inc., Sebastopol, CA, 2008. 2

[54] A. Robins. Catastrophic forgetting in neural networks: The role of rehearsal mechanisms. In *Proceedings of the New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems*, pp. 65–68. IEEE, Nov. 1993. doi: 10.1109/ANNES.1993.323080 2

[55] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne. Experience replay for continual learning. In *Proceedings of the International Conference on Neural Information Processing Systems*, NeurIPS 2019, p. 11 pages. Curran Associates, Inc., 2019. 2

[56] H. Shin, J. K. Lee, J. Kim, and J. Kim. Continual learning with deep generative replay. In *Proceedings of the International Conference on Neural Information Processing Systems*, NIPS 2017, p. 10 pages, 2017. 2

[57] E. Song, D. Oh, and B.-S. Oh. Visual prompt selection framework for real-time object detection and interactive segmentation in augmented reality applications. *Applied Sciences*, 14(22):10502:1–10502:16, 2024. doi: 10.3390/app142210502 3

[58] A. Taheri Tajar, A. Ramazani, and M. Mansoorizadeh. A lightweight Tiny-YOLOv3 vehicle detection approach. *Journal of Real-Time Image Processing*, 18(6):2389–2401, May 2021. doi: 10.1007/s11554-021-01131-w 1

[59] Y. Tian, T. Guan, and C. Wang. Real-time occlusion handling in augmented reality based on an object tracking approach. *Sensors*, 10(4):2885–2900, Mar. 2010. doi: 10.3390/s100402885 9

[60] M. Toneva, A. Sordoni, R. T. d. Combes, A. Trischler, Y. Bengio, and G. J. Gordon. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018. 8

[61] D. Tzutalin. Labelimg: Label image bounding boxes for object detection. `https://github.com/tzutalin/labelImg`, 2015. Accessed: 2025-07-29. 3

[62] M. Uma, S. Abirami, M. Ambika, M. Kavitha, S. Sureshkumar, and K. R. A review on augmented reality and YOLO. In *Proceedings of the International Conference on Smart Electronics and Communication*, ICOSEC 2023, pp. 1025–1030, Sept. 2023. doi: 10.1109/ICOSEC58147.2023.10275842 3, 5

[63] G. M. Van de Ven, T. Tuytelaars, and A. S. Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, 2022. 2

[64] J. Yoon, E. Yang, J. Lee, and S. J. Hwang. Lifelong learning with dynamically expandable networks. In *Proceedings of the International Conference on Learning Representations*, ICLR 2018, p. 11 pages, Apr. 2018. 2

[65] H. Zhang, Y. Jiao, Y. Yuan, Y. Li, Y. Wang, W. Lu, J. Fuh, and B. Li. Object detection and text recognition for immersive augmented reality training in laser powder bed fusion. *Procedia Computer Science*, 232:913–923, 2024. doi: 10.1016/j.procs.2024.01.091 3

[66] P. Zhao, H. Zhang, Q. Yu, Z. Wang, Y. Geng, F. Fu, L. Yang, W. Zhang, J. Jiang, and B. Cui. Retrieval-augmented generation for AI-generated content: A survey. *arXiv preprint arXiv:2402.19473*, p. 22 pages, 2024. 9

[67] D.-W. Zhou, Q.-W. Wang, Z.-H. Qi, H.-J. Ye, D.-C. Zhan, and Z. Liu. Class-incremental learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):9851–9873, July 2024. doi: 10.1109/TPAMI.2024.3429383 2