

# Sistema Multimodal para un asistente robótico por medio de aprendizaje por refuerzo

Jorge Sebastián Mora Lara

Universidad de los Andes  
Bogotá D.C.

js.moral@uniandes.edu.co

**Asesor:** Fernando Enrique Lozano Martínez

flozano@uniandes.edu.co

**Co-asesor:** Carolina Higuera Arias

c.higuera10@uniandes.edu.co

## Abstract

*En este trabajo se presentara un sistema multimodal construido por medio de aprendizaje por refuerzo, que le permitirá al robot tipo Pepper ser un apoyo para los profesores de laboratorio, en donde por medio de sus sensores infrarrojos, sus cámaras 2D y 3D obtiene información de su entorno. Esta información es posteriormente procesada por múltiples modelos de aprendizaje automático, para obtener así una mayor información del ambiente, como una detección y reconocimiento de los objetos que el robot ve y la capacidad de reconstruir y clasificar las poses de los estudiantes. Este sistema al ser producto de una política obtenida por medio aprendizaje por refuerzo en ambientes continuos le permite una mayor adaptabilidad a ambientes similares, a diferencia de otros métodos como programación por maquinas de estados, las cuales están confinadas a los ambientes a las cuales fueron diseñadas.*

## 1. Introducción

La robótica social es un campo que ha crecido bastante en los últimos años, demostrando su potencial en la realización de tareas cotidianas, desde el apoyo como maestro de actividades físicas [1] a asesor de ventas [17], entre otras actividades. Uno de los robots más utilizados en este campo es el robot tipo Pepper [19], fabricado por la empresa Softbank, principalmente diseñado para que visualmente sea agradable, ya que tiene forma humanoide del torso hacia arriba, mientras que del torso hacia abajo tiene un sistema de movimiento omnidireccional basado en tres ruedas en su

base, como se puede ver en la figura 1. Cuenta con una tableta en su pecho, que le permite aún mayor interacción, así como con cámaras 2D y 3D, ubicadas en la frente, boca y ojo derecho, micrófonos y sensores touch ubicados en la parte superior de la cabeza, sensores infrarrojos y ultrasonidos ubicados en la parte inferior del cuerpo.

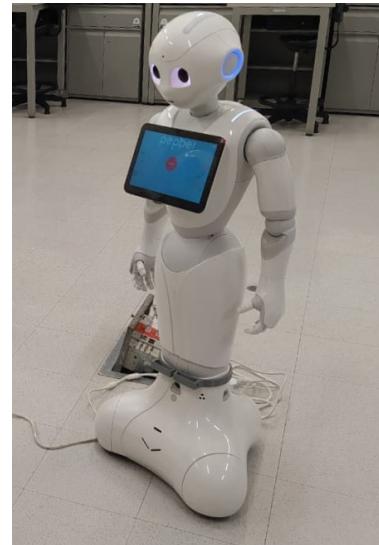


Figura 1: Robot tipo Pepper de la universidad de los Andes.

En este trabajo se diseñó e implementó un sistema multimodal por medio de aprendizaje por refuerzo, el cual utilizará la información de los sensores láser, la cámara 2D, 3D y múltiples modelos de aprendizaje automático, para crear un sistema que le permita al robot Pepper ser un asistente

para un profesor de laboratorio. Esto implica, primero, que tenga la capacidad de navegar de forma segura en un laboratorio, evitando chocar con las personas u objetos; segundo, debe tener la capacidad de reconocer quien tiene dudas o preguntas y acercarse a ellos para resolverlas; por ultimo, debe ser capaz de hacer cumplir las normas del laboratorio, como son: no ingerir alimentos o tomar bebidas y no utilizar aparatos electrónicos como reproductores o celulares durante las clases.

Para lograr este sistema multimodal se combinará una serie de algoritmos de aprendizaje de máquina y procesamiento de datos, como son el reconocimiento de objetos por medio de redes convolucionales profundas, la reconstrucción de poses por medio de una arquitectura basada en redes convolucionales y la clasificación de las mismas por medio de una red neuronal. Finalmente se obtuvo una política por medio de aprendizaje por refuerzo a través del algoritmo Proximal Policy Optimization [20], la cual utiliza la información proporcionada por los algoritmos previamente mencionados además de la información de los sensores infrarrojos y de la cámara 3D. Esta política le permite al robot Pepper navegar de forma segura en el entorno, reconocer y acercarse a las personas que tienen preguntas y vigilar que se cumpla las normas de seguridad.

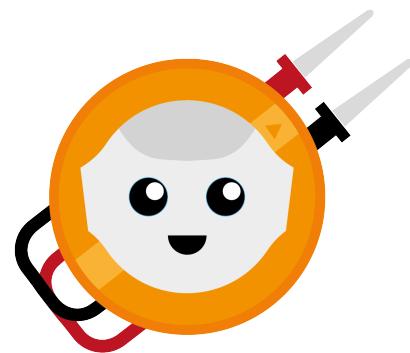
## 2. Simulador

Se diseño y construyó un simulador a partir del motor gráfico Unity 5 [27], el cual fue nombrado P.E.R Lab (Pepper Electronic Robot Laboratory), cuyo logo se puede ver en la figura 2. Este simulador está disponible en el repositorio<sup>1</sup>. En este simulador también se diseñó un modelo del robot Pepper, el cual se muestra en la figura 3, a este modelo se le programó y configuró los diferentes sensores de acuerdo a las especificaciones del robot Pepper [19] suministradas por Softbank.

Posteriormente se diseñó un ambiente basado en el laboratorio de ingeniería ML009 de la universidad de los Andes. En la figura 4 se puede apreciar el diseño final del ambiente.

Con el fin de que se pueda realizar un entrenamiento robusto en donde el robot se enfrente a diferentes variaciones del ambiente, durante cada episodio se cambia el numero de estudiantes, el numero de objetos en cada puesto de trabajo, la posición del tablero dentro del ambiente, la posición inicial del robot y la ruta por donde camina el profesor. En la tabla 1 se resume la cantidad de elementos que existen y con que probabilidad cada uno puede aparecer en un episodio según una distribución uniforme.

Adicionalmente durante la simulación los modelos de las personas son capaces de hacer diferentes acciones, como son teclear, manipular ciertos equipos de laboratorio, preguntar, o pedir que se aleje de ellos. Estas acciones se real-



**P.E.R. Lab**

Figura 2: Simulador P.E.R. Lab.



Figura 3: Modelo 3D del robot Pepper usado en el simulador.



Figura 4: Ambiente de simulación

<sup>1</sup><https://github.com/JorgeSebastianML/P.E.R.-Lab>

	Cantidad	Probabilidad
Estudiantes	46	0.2
Objetos	45	0.2
Posición del tablero	1350	1/1350
Dirección del profesor	2520	1/2520
Posición inicial del robot	1350	1/1350

Table 1: Cambios en el ambiente.

izan de forma aleatoria, según una distribución uniforme, con una probabilidad del 10% cada 30 segundos. En el vídeo <https://youtu.be/M485jqBbsXA> se puede apreciar el funcionamiento y características del simulador.

### 3. Metodología

#### 3.1. Enfoque de la solución

Para diseñar y construir este sistema se dividió el problema en cinco partes, los cuales son: El reconocimiento de objetos, la reconstrucción de poses, la clasificación de poses y la integración de los diferentes modelos, junto con la navegación e interacción del robot en el ambiente. Para cada subproblema se revisó el estado del arte para encontrar herramientas y algoritmos que permitieran resolverlo con el mejor desempeño posible. En la figura 5 se muestra el esquema utilizado para resolver este problema por medio de aprendizaje por refuerzo. En el vídeo <https://youtu.be/m1rBkZ3Yyu4> se puede observar el funcionamiento del sistema una vez implementado en el simulador.

#### 3.2. Reconocimiento de objetos

Para el reconocimiento de objetos se realizó una investigación del estado del arte en el tema, en donde se encontró que los mejores modelos se basan en redes convolucionales profundas, como Yolo v3 [18], resnet50 [12], VGG [21], Mask R-CNN [11], Yolo v4 [4] y Yolo v5 [26]. Estos fueron comparados bajo el dataset COCO [15], uno de los dataset actuales más utilizados para la comparación de modelos de detección de objetos, bajo la métrica Average Precision (AP), la cual calcula la precisión de la intersección sobre la unión de los bounding box predichos por el modelo contra los bounding box reales del dataset, como se ve en la ecuación 1. Adicionalmente se tomo en cuenta el numero de FPS que lograban bajo el mismo hardware (o similar), esto con el fin de tener en cuenta su desempeño y si es capaz de trabajar en tiempo real. Al revisar los resultados de los pappers Yolov4: Optimal speed and accuracy of object detection [4], Comparison of faster R-CNN models for object detection [14] y A review: Comparison of performance metrics of pretrained models for object detection using the TensorFlow framework [25], esta comparación se resume

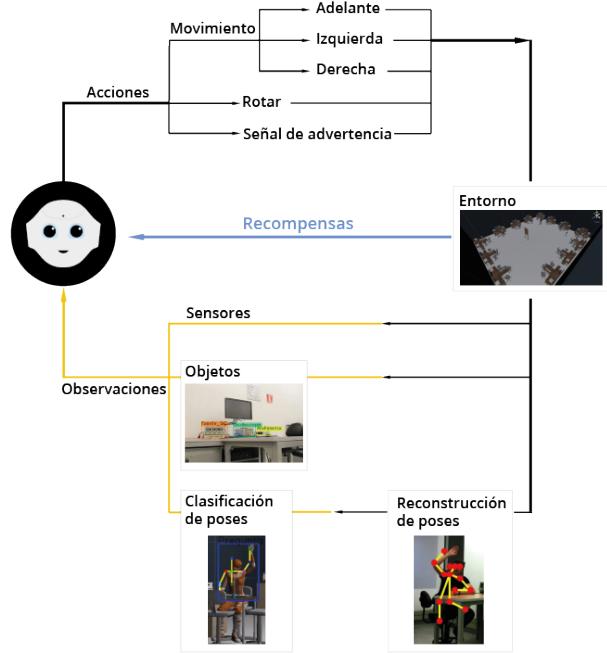


Figura 5: Diagrama de la solución

en la tabla 2. Por lo que se determinó que la mejor arquitectura para este proyecto es Yolo V4 [4], la cual tiene 43.5% AP a 65 FPS utilizando una GPU Tesla V100, teniendo un buen desempeño bastante bueno.

$$IoU = \frac{|A \cap B|}{|A \cup B|} \quad (1)$$

A: ground Truth Bounding Box.

B: bounding Box Prediction.

Arquitectura	AP en COCO
Yolo v3 - Darknet	33%
<b>Yolo v4 - Darknet</b>	<b>43.5%</b>
Yolo v5 - UltraAnalytics	42%
Mask R-CNN	36.4%
Faster R-CNN con Interception-ResVet v2	36.8%

Table 2: Comparación de modelos de reconocimiento de objetos.

#### 3.3. Reconstrucción de Poses

Para la reconstrucción de poses al igual que para el reconocimiento de objetos, se realizó una investigación del estado del arte en donde se tomaba en cuenta su precisión

en la reconstrucción de poses en un subdataset de COCO [15]. Se utilizó la métrica Average Precision (AP) descrita en la ecuación 1, para evaluar y comparar las arquitecturas. También se tuvo en cuenta los Frames Per Seconds (FPS), debido a que se busca que tenga la capacidad de ejecutarse en tiempo real junto con otros modelos de machine learning. A partir de esta investigación se encontraron las siguientes arquitecturas: AlphaPose [9], Mask R-CNN [11] y OpenPose [6], entre ellas las que mejor desempeño tiene en la métrica Average Precision es AlphaPose [9], la cual tiene 71%, mientras que OpenPose [6] tiene un 64.2%, pero este tiene mejor rendimiento en FPS, según el paper MVOR: A Multi-view RGB-D Operating Room Dataset for 2D and 3D Human Pose Estimation [23]. Finalmente se decidió utilizar la arquitectura de OpenPose [6], debido a que tiene mejor relación entre precisión y FPS, segun reportado en el paper mencionado, en donde lo comparan contra otros modelos incluyendo AlphaPose [9]. Esto permite una buena reconstrucción en tiempo real en paralelo a los demás modelos. En las figuras 6, 7 se puede ver las dos poses de interés reconstruidas con el modelo y en el vídeo <https://youtu.be/BWQLKSDDbq4> se puede apreciar el modelo implementado.

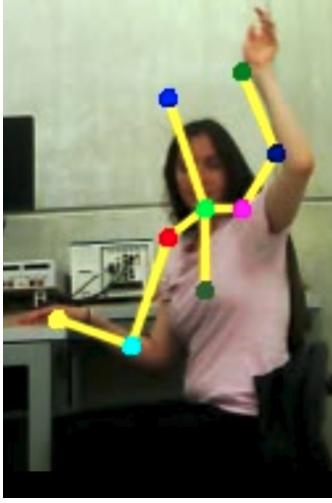


Figura 6: Pose de pregunta reconstruida con OpenPose.

### 3.4. Clasificación de poses

Se realizo un clasificador para reconocer tres diferentes poses: Alto, pregunta y ninguna. Para ello se exploraron diferentes algoritmos, como redes neuronales totalmente conectadas, redes convolucionales de pocas capas y redes convolucionales profundas. Para comparar estos métodos se creo un dataset a partir de vídeos procesados con la red OpenPose [6], en donde participaron más de 20 personas. Se compararon los modelos por medio de la métrica Accuracy, definida por la ecuación 2.

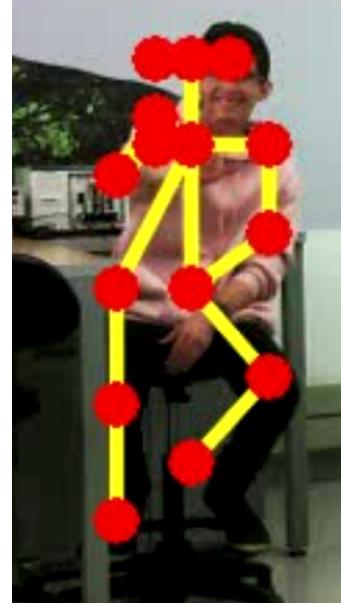


Figura 7: Pose de alto reconstruida con OpenPose.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \quad (2)$$

TP: True Positive.

TN: True Negative.

FP: False Positive.

FN: False Negative.

### 3.5. Navegación e interacción

Para realizar la navegación, interacción e integración se utilizaran algoritmos de aprendizaje por refuerzo, ya que estos permiten una mejor adaptabilidad a diferentes ambientes y han demostrado una gran capacidad para resolver problemas complejos, como se muestra en el paper Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research [16]. Con el fin de determinar cual de los diferentes algoritmos se utilizará se realizó una investigación en donde se revisaron: proximal policy optimization (PPO) [20], soft actor-critic (SAC) [10], Deep Q-Learning. Con base en los resultados presentados en el paper A Comparison of Action Spaces for Learning Manipulation Tasks [28] se decidió utilizar proximal policy optimization (PPO) [20], en donde se demuestra que obtiene muy buenos resultados en el ámbito de la robótica igualando a soft actor-critic, pero obteniendo políticas más estables y menos ruidosas, adicionalmente este algoritmo ha mostrado muy buenos resultados en otros trabajos como Emergent Tool Use From Multi-Agent Autocurricula [2], Intelligent Control of a Quadrotor with Proximal Policy Optimization Reinforcement Learning [5] y Learning Humanoid Robot Running Skills through Proximal Policy Optimization [7].

en donde el espacio de estados es muy grande y es necesario trabajar en ambientes continuos, lo cual pasa en este problema, ya que el entorno no se puede discretizar fácilmente debido a la interacción que el robot debe hacer con las personas.

## 4. Trabajo desarrollado

### 4.1. Reconocimiento de objetos

#### 4.1.1 Yolo v4

Como se mencionó en la sección 3.2, se seleccionó la arquitectura de Yolo v4[4], debido a su precisión y desempeño frente a otras arquitecturas. Esta fue implementada por medio del repositorio oficial de darknet<sup>2</sup>, los creadores de la red, y la librería de OpenCV.

Se construyó un dataset con los objetos de interés. Las imágenes de este dataset fueron recolectadas de los equipos de los laboratorios de ingeniería eléctrica y electrónica de la universidad de los Andes, también se obtuvieron imágenes del dataset Imagenet [8], y de internet. En la tabla 3 está el resumen de las clases y la cantidad de imágenes en cada una.

Clase	# de imágenes	% del dataset
Bebidas	497	19.5%
Osciloscopio	457	17.93%
Multímetro	405	15.89%
Generador de señales	323	12.67%
Snack	275	10.79%
Aparatos electrónicos	269	10.55%
Fuente de alimentación	323	12.67%
Total	2549	100%

Table 3: Dataset de objetos.

Posteriormente, se modificó la arquitectura de la red para que tuviera 7 neuronas de salida y fue entrenada utilizando los pesos pre-entrenados en el dataset COCO [15]. Con el fin de validar el desempeño de este modelo, se dividió el dataset utilizando el 90% para entrenar y 10% restante para validar. Durante el proceso de entrenamiento se verificaba el mean average precision (mAP) que obtenía esta red sobre los datos de validación y el resultado del proceso de entrenamiento se puede apreciar en la figura 8 y se puede observar que se logra obtener resultados bastante buenos, obteniendo un mAP de 80.8%.

Al ejecutarse sobre una GPU Nvidia RTX 2070, se obtiene un rendimiento en tiempo real, logrando un reconocimiento de 30 FPS, usando el 20% de la capacidad de la GPU, dejando suficiente capacidad para los demás modelos.

<sup>2</sup><https://github.com/AlexeyAB/darknet>

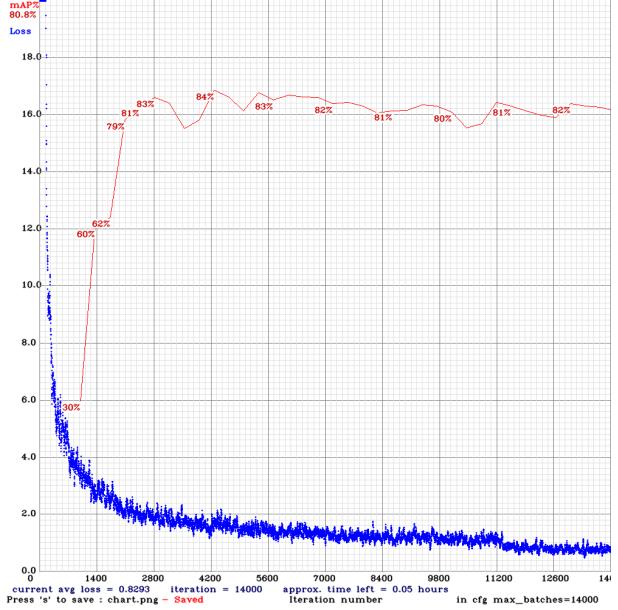


Figura 8: Curvas de entrenamiento y validación de la red convolucional Yolo v4.

#### 4.1.2 Seguimiento de objetos

Debido a que Yolo v4 [4] es una arquitectura que trabaja sobre una imagen individual, no toma en cuenta información de los frames anteriores, lo que genera inconsistencias entre ellos como son la no detección de un objeto que un frame anterior pudo detectar. Esto se debe principalmente al movimiento de la cámara y occlusiones temporales. Para solucionar este problema se implementó un filtro de kalman, el cual es un algoritmo utilizado para actualizar la salida del modelo, tomando como base información de observaciones previas, permitiendo en este caso realizar tracking de los objetos observados.

### 4.2. Reconstrucción de poses

Partiendo del repositorio oficial de OpenPose<sup>3</sup>, se utilizó la arquitectura y los pesos pre-entrenados, los cuales fueron implementados por medio de la librería de OpenCV. Posteriormente se modificó la salida de la red para solo tener en cuenta aquellos puntos que correspondan a la parte media y superior del cuerpo, puesto que las poses de interés son representadas principalmente por los brazos y su relación con el torso, además de que por lo general las personas permanecen sentadas en sus puestos de trabajo.

Al ejecutarse sobre una GPU Nvidia RTX 2070, se obtiene un rendimiento en tiempo real, logrando una reconstrucción de las poses a 30 FPS usando el 15% de la ca-

<sup>3</sup><https://github.com/CMU-Perceptual-Computing-Lab/openpose>

pacidad de la GPU, en el vídeo se puede apreciar el funcionamiento de la red.

### 4.3. Clasificación de poses

#### 4.3.1 Dataset

Con el fin de entrenar los modelos dedicados a la clasificación de poses, se creó un dataset con las poses de interés. Este dataset se realizó tomando vídeo desde las cámaras 2D del robot tipo Pepper de múltiples personas realizando diferentes variaciones de las poses de interés, las cuales son: Pregunta y Alto como se ven en las figuras 6 y 7. Posteriormente se pasaron los vídeos por la red de reconstrucción de poses OpenPose [6], finalmente los vídeos resultantes se descomponieron en frames y a partir de estos se construyó el dataset. En la tabla 4, se puede apreciar el estado final del dataset obtenido.

Clase	# Imágenes	% del Dataset
Pregunta	330	26.44%
Alto	251	20.11%
Ninguna	667	53.45%
Total	1248	100%

Table 4: Dataset de poses.

#### 4.3.2 Exploración y selección del modelo de clasificación

Se exploraron múltiples modelos para la clasificación de las imágenes en las tres poses de interés, entre los que destacan: Redes totalmente conectadas, Redes convolucionales de pocas capas y redes convolucionales profundas. Para hacer un análisis fiable, se dividió el dataset en 80% entrenamiento y 20% validación, utilizando los mismos datos para los modelos probados. A continuación se mostrarán los resultados obtenidos en esta exploración:

**Redes Totalmente conectadas:** Se realizó una búsqueda de grilla en donde se variaron el número de capas ocultas y las neuronas por capa. En la figura 9 se puede ver los resultados obtenidos.

Al analizar estos resultados se concluyó que esta aproximación en términos generales es bastante buena para clasificar las poses, obteniendo resultados superiores al 80% de accuracy en datos de validación con la mayoría de modelos, pero entre estos el que destaca posee tres capas ocultas con 300 neuronas por capa, obteniendo un accuracy del 88.99%.

**Redes Convolucionales de pocas capas:** Se construyó una red convolucional de tres capas, en donde se varió la tasa de aprendizaje. A diferencia de la arquitectura anterior, esta obtiene resultados regulares, en donde el mejor modelo

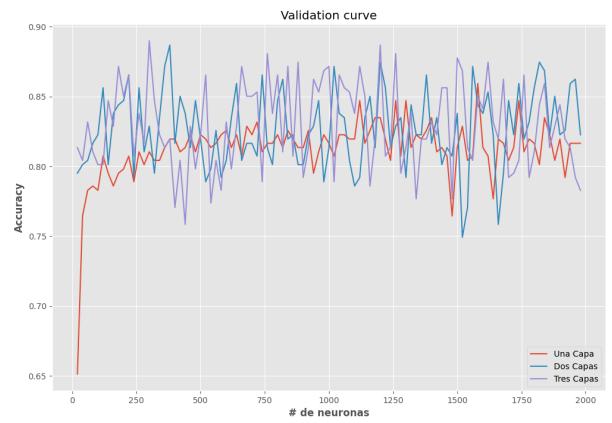


Figura 9: Curva de Accuracy vs de neuronas en datos de validación

obtiene un accuracy del 51.56%, demostrando que esta arquitectura no soluciona bien este problema.

**Redes Convolucionales profundas, con pesos pre-entrenados:** Se partió de la arquitectura de Yolo V4 [4] para realizar el modelo de clasificación, modificando los parámetros de la red para la cantidad de clases utilizadas. En la figura 10 se puede ver la curva de aprendizaje obtenida durante el entrenamiento y al evaluarlo sobre los datos de validación se obtuvo un accuracy del 89.84%.

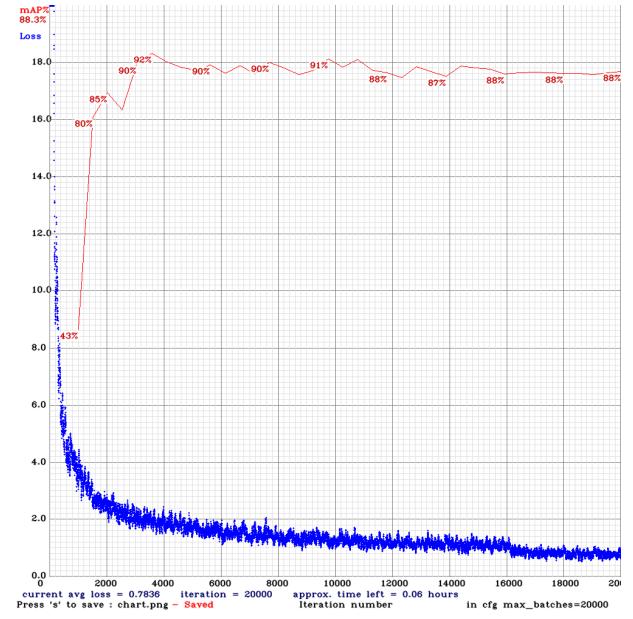


Figura 10: Curvas de entrenamiento y validación de la red convolucional profunda Yolo V4, para la clasificación de poses.

En la tabla 5 se resume la comparación de los mejores modelos obtenidos de las diferentes arquitecturas. En esta comparación se utilizaron múltiples métricas para realizar un mejor análisis, en donde se concluyó que los mejores resultados se obtuvieron con la red convolucional profunda con pesos pre-entrenados, por lo que esta fue la implementada en el sistema.

Modelo	Clase	Precisión	Recall	F1-Score	Accuracy
Red totalmente conectada	Pregunta	99%	93%	96%	88.99%
	Ninguna	85%	99%	92%	
	Alto	84%	58%	69%	
Red convolucional de pocas capas	Pregunta	74%	43%	54%	51.56%
	Ninguna	41%	82%	55%	
	Alto	66%	33%	44%	
Red convolucional profunda	Pregunta	76%	100%	86%	89.84%
	Ninguna	100%	82%	90%	
	Alto	100%	89%	94%	

Table 5: Resultados de los mejores modelos por arquitectura para la clasificación de poses.

#### 4.4. Navegación e Interacción, utilizando aprendizaje por refuerzo

Como se mencionó en la sección 3.5, se utilizó aprendizaje por refuerzo, específicamente proximal policy optimization (PPO) [20] para generar una política la cual le permite al robot Pepper realizar la navegación e interacción dentro del laboratorio, partiendo de la información procedida por los modelos previamente descritos así como la información de sensores infrarrojos y de la cámara de profundidad.

Para lograr esto, se utilizó el simulador P.E.R. Lab descrito en la sección 2, que por medio del paquete Unity ML-Agents Toolkit [13], permite la comunicación con el lenguaje de programación Python y proporciona varias herramientas para aplicar los diferentes algoritmos de aprendizaje por refuerzo, tanto en ambientes discretos, como en ambientes continuos.

##### 4.4.1 Acciones del robot

Se configuró el simulador para que el robot pueda realizar las siguientes acciones: Avanzar hacia adelante, avanzar hacia la derecha, avanzar hacia la izquierda, rotar sobre su eje en sentido horario y antihorario, mover verticalmente la cabeza y mandar una señal de advertencia.

Estas acciones fueron definidas ya que le permiten tener la suficiente movilidad para interactuar con este ambiente, sin complicar demasiado el modelo, ya que se evitaron usar acciones como movimientos en diagonal o movimientos más complejos de la cabeza. También se evitó usar la posibilidad del movimiento hacia atrás, debido a que es un movimiento peligroso para el robot ya que no posee sen-

sores apuntando a esa dirección y por lo tanto andaría a ciegas.

##### 4.4.2 Observaciones del ambiente

Para que el robot Pepper pueda tomar decisiones debe obtener información de su ambiente por medio de sus sensores y los modelos previamente descritos. Las observaciones consisten en:

- Velocidad lineal del robot en el eje X.
- Velocidad lineal del robot en el eje Z.
- Velocidad angular de rotación del robot sobre su eje.
- Ángulo de la cabeza.
- Información de distancia de los sensores láseres del robot.
- Información de profundidad de la cámara 3D del robot.
- Información de las imágenes captadas por las cámaras 2D del robot después de ser procesadas por los modelos de reconstrucción de poses, reconocimiento de poses y de objetos.

##### 4.4.3 Definición de recompensas

Una parte fundamental del aprendizaje por refuerzo son las recompensas que se le darán al agente cuando realice ciertos comportamientos u obtenga ciertos resultados, ya que a través de estas el algoritmo ajusta la política a seguir. Teniendo en cuenta esto se realizaron múltiples pruebas para ajustar las recompensas, con el fin de obtener una política que describa un comportamiento deseable para resolver este problema. Teniendo en cuenta la literatura y los proyectos que han utilizado este algoritmo, las recompensas fueron ajustadas en un rango de -1 a 1. A continuación se mostrarán las diferentes recompensa, el porque se definieron así y como se modificaron a lo largo de las pruebas, en donde la recompensa total es la suma de las ocho descritas:

**Primera recompensa:** Tiene un valor negativo de -1 y se da cuando los sensores infrarrojos del robot detecta algo a menos de 40 centímetros, esto con el fin de enseñarle a evitar chocarse, adicionalmente esto es una señal de terminación del episodio.

**Segunda recompensa:** Tiene un valor positivo que se da cuando el robot se acercaba a una persona que pregunta, inicialmente fue fijada en un valor de 0.5, pero posteriormente se modificó para que fuera una recompensa dinámica de acuerdo a la ecuación 3. Se decidió que fuera dinámica, para recompensar más entre más se acerque a la persona que pregunta.

$$Recompensa = 1 - \frac{DRS}{MDS} \quad (3)$$

DRS: Distancia Retornada por el Sensor.

MDS: Distancia Máxima del Sensor.

**Tercera recompensa:** Tiene un valor negativo y esta se da cuando el robot se acerca a una persona que no quiere que se le acerque, inicialmente fue definido con un valor de -0.5, pero posteriormente se decidió que fuera dinámica de acuerdo a la ecuación 4, se puede apreciar que es similar a la ecuación 3, con la diferencia que esta multiplicada por -0.9, esto se decidió de esta forma para que aprendiera a alejarse de las personas que no quieren que se les acerque tan pronto las detecte y evite acercarse más, pero para evitar que el robot no explore, se decidió que fuera menor en relación a encontrar a una persona preguntando.

$$Recompensa = -0.9 * (1 - \frac{DRS}{MDS}) \quad (4)$$

DRS: Distancia Retornada por el Sensor.

MDS: Distancia Máxima del Sensor.

**Cuarta recompensa:** Tiene un valor positivo y esta se da cuando detecta un objeto prohibido y mandaba la señal de advertencia. Inicialmente fue ajustada en 0.5, pero posteriormente fue aumentada a 1, debido a que le costaba mandar la señal cuando era requerida ya que la mayoría de veces tendía a equivocarse y quería evitar obtener la recompensa negativa.

**Quinta recompensa:** Tiene un valor negativo y esta se da cuando no detecta un objeto prohibido pero manda la señal de advertencia, inicialmente se le dio un valor de -0.5, pero debido a la situación anteriormente descrita fue bajado a -0.1.

**Sexta recompensa:** Tiene un valor positivo de 0.01, que se da cada 10 segundos que el robot permanezca sin chocar. Esta recompensa fue planteada así para incentivar que se mantuviera en el ambiente sin chocarse, buscando una navegación segura tanto para el robot como para las personas que lo rodean.

**Séptima recompensa:** Tiene un valor negativo de -1, que se da si el robot no se movía más de 35cm en un minuto, adicionalmente es una condición de terminación del episodio. Esta recompensa fue creada para incentivar al agente a moverse y explorar más el ambiente.

**Octava recompensa:** Tiene un valor negativo de -0.005, que se da cuando el robot no encontraba a una persona que pregunta en 10 segundos. Esta recompensa fue creada con el motivo de incentivar la exploración en el ambiente y la búsqueda de personas con dudas o preguntas.

#### 4.4.4 Arquitectura de la red neuronal que estima la política

Para definir la arquitectura de la red neuronal que estimará la política, se realizaron múltiples pruebas en donde fueron entrenadas por más de 1.5 millones de pasos y se evaluaba la recompensa obtenida a lo largo del entrenamiento. En la figura 11 se muestra los resultados obtenidos cuando se variaron el número de capas y cantidad de neuronas por capa de la red neuronal, en donde finalmente el mejor resultado fue obtenido con una red de 3 capas ocultas con 128 neuronas por capa.

Posteriormente se exploraron múltiples tamaños de batch de entrenamiento y el tamaño del buffer de memoria, el cual es utilizado para almacenar información relevante de estados pasados para la toma futura de decisiones. En la figura 12 se muestra los resultados obtenidos, en donde el mejor modelo se obtuvo con un tamaño del batch de entrenamiento de 512 y un tamaño de memoria de 2048.

Se analizó si añadirle una red neuronal LSTM a la parte de memoria de la red mejoraría su rendimiento como sugería el paper Reinforcement Learning with Long Short-Term Memory [3], para ello se comparó con la red neuronal base encontrada en los experimentos anteriores, al mismo tiempo se exploraron diferentes arquitecturas de LSTM, en donde se variaba el tamaño de la memoria y la longitud de la secuencia de entrada. En la figura 13 se muestran los resultados obtenidos, en donde se puede concluir que la LSTM incrementa el rendimiento del modelo, específicamente una cuyo tamaño de memoria es de 256 y la longitud de la secuencia de entrada es de 32.

Finalmente, se utilizó una métrica la cual consiste en el porcentaje de personas que preguntan que el robot detectó y se acercó, también se tomó en cuenta el porcentaje de objetos prohibidos que el robot detectó, en la tabla 6 se muestra los resultados obtenidos tanto con la red neuronal LSTM en la memoria como sin ella. Estos resultados se obtuvieron después de correr los modelos en 28 episodios diferentes y promediando los resultados obtenidos. Adicionalmente se compara con un baseline el cual consiste en el porcentaje de preguntas que respondió un profesor en dos semestres en una clase virtual. Estos datos fueron obtenidos del paper Towards Understanding Online Question Answer Interactions and their effects on student performance in large-scale STEM classes [22].

#### 4.4.5 Implementación de Learning Options

Al analizar los resultados obtenidos se obtuvo que en promedio el robot es capaz de encontrar el 42% de las personas que preguntan y el 24% de los objetos prohibidos. Con el fin de mejorar estos resultados se implementó Learning options [24], el cual consiste en que el agente tenga la capacidad de utilizar acciones más complejas y de mayor

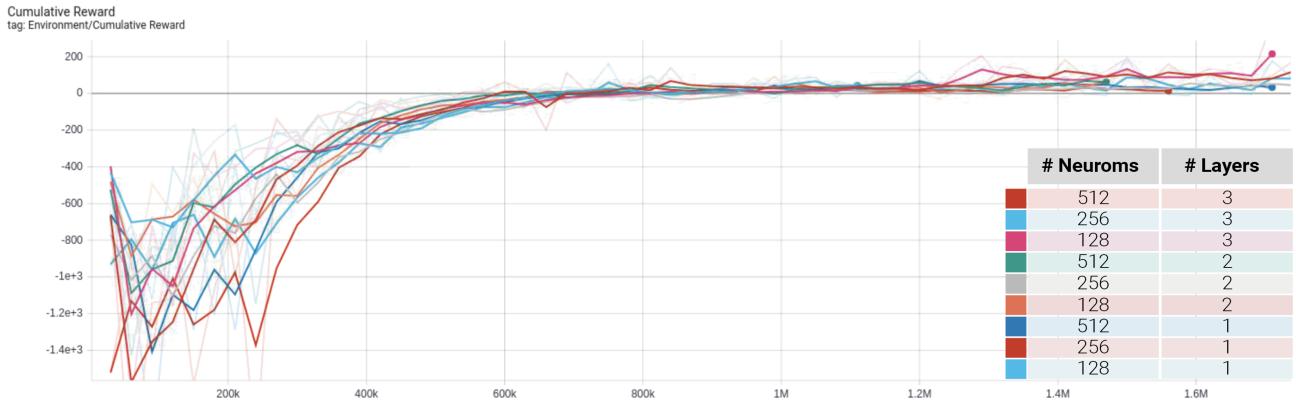


Figura 11: Recompensa vs Épocas, variando el numero de capas y neuronas.

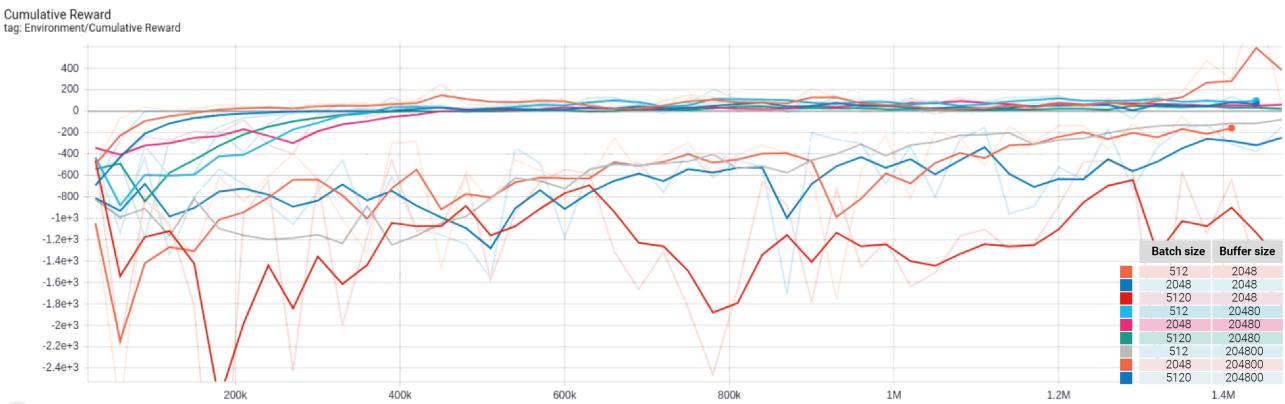


Figura 12: Recompensa vs Épocas, variando el tamaño del batch de entrenamiento y el tamaño del buffer de memoria.

Modelo		Promedio	STD
Humano	Persona	73.5%	-
Mejor modelo con LSTM	Persona	42.32%	34.08%
	Objeto	24.53%	13.98%
Mejor modelo sin LSTM	Persona	38.1%	34.42%
	Objeto	29.39%	16.64%

Table 6: Resultados de los diferentes modelos con y sin LSTM en la memoria.

duración en vez de muchas acciones sencillas, en este caso se decidió que dichas acciones se enfocaran en facilitar la navegación, para ello se modificó las acciones para que el robot pueda tomar una única dirección a la vez, en una línea recta que experimentalmente se determinó que fuera de un metro, tanto para adelante, izquierda y derecha. Otra acción es que rotara sobre su eje sin desplazarse, tanto en sentido horario como antihorario, en vez de que continuamente estuviera escogiendo una dirección, en donde también podía

combinar direcciones. En la tabla 7 se muestra una comparación de los resultados obtenidos con Learning options y el mejor modelo previamente encontrado y se puede notar como la implementación de este método mejoró notablemente el rendimiento del modelo, acercándose bastante al rendimiento humano. En el vídeo <https://youtu.be/ZEGzEKfldMg> se muestra una comparación entre la política encontrada con las acciones normales vs la política obtenida con Learning Options.

Modelo		Promedio	STD
Humano	Persona	73.5%	-
Learning Options	Persona	68.75%	29.23%
	Objeto	36.82%	16.22%
Mejor Modelo con LSTM	Persona	42.32%	34.08%
	Objeto	24.53%	13.98%

Table 7: Resultados de Learning options.

Debido al que el reconocimiento de objetos prohibidos

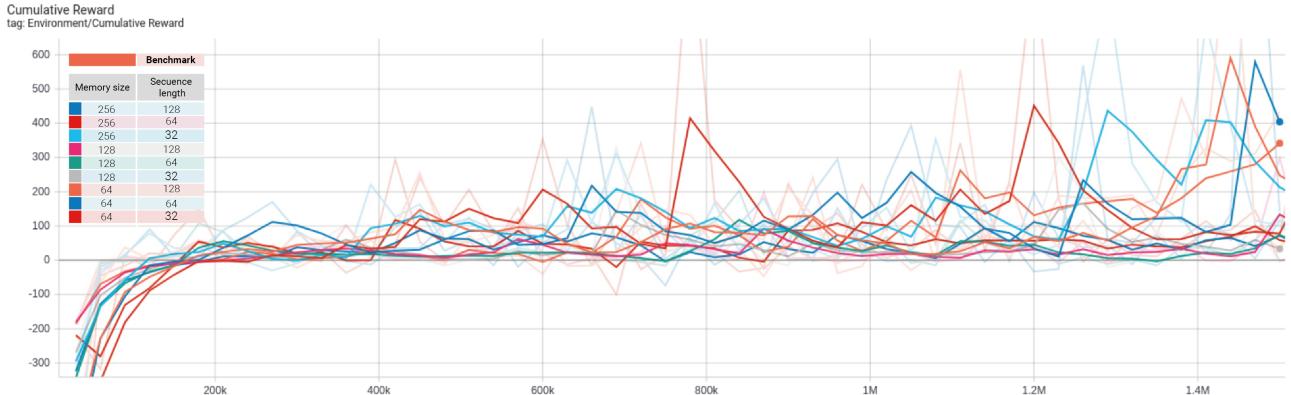


Figura 13: Recompensa vs Épocas, variando el tamaño de la memoria y la longitud de la secuencia de memoria de la LSTM.

por parte del robot es bajo a comparación al reconocimiento de personas con preguntas, se realizó un análisis de que objetos le costaba al robot encontrar. Para ello se sacó una lista de los objetos que no encontraba en cada episodio y después de 28 episodios se encontró que los celulares y las botellas de agua son los objetos que más se le dificulta. En la tabla 8 se muestra el promedio de la cantidad de objetos que no fueron encontrados a lo largo de los 28 episodios. Esto se debe, en el caso de los celulares a su tamaño, que solo se pueden ver desde ciertos ángulos y distancias, por otro lado las botellas de agua se le complica debido al color que es similar al de las paredes.

Objeto	Promedio	STD
Celular	1.61	1.07
Coca-Cola	0.32	0.48
Manzanas	0.82	0.95
Botella de agua	1.86	1.3
Papas Pringles	0.75	1

Table 8: Análisis de reconocimiento de objetos.

## 5. Conclusión

Sistema multimodal para un asistente robótico por medio de aprendizaje por refuerzo, tiene la capacidad de ejecutar labores sencillas dentro de los laboratorios, como son el seguimiento del cumplimiento de las normas establecidas para su uso, la navegación segura y la capacidad para reconocer a las personas que tienen preguntas.

La implementación e integración de múltiples sistemas basados en aprendizaje de máquina por medio de aprendizaje por refuerzo permite una gran flexibilidad al momento de resolver un problema complejo como es este, ya que permite la integración de múltiples fuentes de información como el reconocimiento de una gran variedad

de objetos, la caracterización y clasificación de las personas por medio de sus poses, la información suministrada por múltiples sensores, adicionalmente consigue una navegación reactiva bastante buena, en donde evita chocarse mientras interactúa con el ambiente.

Para la selección de la arquitectura de LSTM usada en la memoria se consideraron las tres que mejores resultados dieron, pero uno de los factores que se tuvieron en cuenta fue la complejidad de esta red, ya que se buscaba la mejor relación entre desempeño y coste computacional, esta misma lógica se aplicó a la selección del batch de entrenamiento y la longitud de la memoria, ya que habían arquitecturas con desempeño similar.

Learning options ha demostrado ser una poderosa herramienta para mejorar la política en un ambiente continuo, ya que al permitirle hacer acciones más complejas por cada decisión que toma, hace que el problema se simplifique.

Los códigos y modelos realizados en este proyecto se encuentran en el repositorio <sup>4</sup>.

## References

- [1] Miguel Cazorla Angelo Costa, Ester Martinez-Martin and Vicente Julian. Pharos—physical assistant robot system. 2018. 1
- [2] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula, 2020. 4
- [3] Bram Bakker. Reinforcement learning with long short-term memory. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2002. 8
- [4] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020. 3, 5, 6

<sup>4</sup>[https://github.com/JorgeSebastianML/Asistente\\_de\\_clase\\_robot\\_tipo\\_pepper](https://github.com/JorgeSebastianML/Asistente_de_clase_robot_tipo_pepper)

- [5] G. Cano Lopes, M. Ferreira, A. da Silva Simões, and E. Luna Colombini. Intelligent control of a quadrotor with proximal policy optimization reinforcement learning. In *2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE)*, pages 503–508, 2018. 4
- [6] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. 4, 6
- [7] L. Carvalho Melo and M. R. Omena Albuquerque Máximo. Learning humanoid robot running skills through proximal policy optimization. In *2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education (WRE)*, pages 37–42, 2019. 4
- [8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 5
- [9] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. Rmpe: Regional multi-person pose estimation. In *ICCV*, 2017. 4
- [10] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018. 4
- [11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2018. 3, 4
- [12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 3
- [13] Arthur Juliani, Vincent-Pierre Berges, Esh Vckay, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. Unity: A general platform for intelligent agents. *CoRR*, abs/1809.02627, 2018. 7
- [14] C. Lee, H. J. Kim, and K. W. Oh. Comparison of faster r-cnn models for object detection. In *2016 16th International Conference on Control, Automation and Systems (ICCAS)*, pages 107–110, 2016. 3
- [15] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015. 3, 4, 5
- [16] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, Vikash Kumar, and Wojciech Zaremba. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *CoRR*, abs/1802.09464, 2018. 4
- [17] Andrea Katherín Pérez Hernández, Mario Fernando de la Rosa Rosero, José Tiberio Hernández Peñaloza, and Fabio Augusto González Osorio. *Identificación de señales multimodales para reconocimiento de emociones en el contexto de interacción humano-robot*. Uniandes, 2017. 1
- [18] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018. 3
- [19] SoftBank Robotics. Pepper - documentation. [http://doc.aldebaran.com/2-4/home\\_pepper.html](http://doc.aldebaran.com/2-4/home_pepper.html). 1, 2
- [20] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. 2, 4, 7
- [21] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. 3
- [22] David H Smith IV, Qiang Hao, Vanessa Dennen, Michail Tsikerdekis, Bradly Barnes, Lilu Martin, and Nathan Tresham. Towards understanding online question answer interactions and their effects on student performance in large-scale stem classes. *International Journal of Educational Technology in Higher Education*, 17(1):1 – 15, 2020. 8
- [23] Vinkle Srivastav, Thibaut Issenhuth, Abdolrahim Kadkhodamohammadi, Michel de Mathelin, Afshin Gangi, and Nicolas Padoy. Mvor: A multi-view rgb-d operating room dataset for 2d and 3d human pose estimation, 2019. 4
- [24] Martin Stolle and Doina Precup. Learning options in reinforcement learning. volume 2371, pages 212–223, 08 2002. 8
- [25] Sergio Sánchez Hernández, H Romero, and A Morales. A review: Comparison of performance metrics of pretrained models for object detection using the tensorflow framework. *IOP Conference Series: Materials Science and Engineering*, 844:012024, 06 2020. 3
- [26] ultralytics. ultralytics/yolo v5 - github. <https://github.com/ultralytics/yolov5>, 2020. 3
- [27] Unity. Unity - homepage. <https://unity.com/>. 2
- [28] Patrick Varin, Lev Grossman, and Scott Kuindersma. A comparison of action spaces for learning manipulation tasks, 2019. 4