Relevancia del contexto y la tarea para la flexibilidad conductual: una arquitectura computacional para la Robótica cognitiva

16 de junio de 2019

Índice

Resumen

El presente proyecto busca inspirarse en el marco explicativo del procesamiento predictivo (PP) para modelar una arquitectura computacional cognitiva que codifique el contexto y la tarea para la ejecución de acciones en un agente artificial. El proyecto ha sido motivado por el PP debido a su interesante propuesta de la inversión del sentido en el que se pensaba que la información sensorial es procesada y a sus estrategias basadas en conocimiento estructurado dependiente del contexto y la tarea a realizar. Esta teoría propone que el cerebro se encuentra constantemente anticipándose a las entradas sensoriales que recibe mediante la generación de predicciones, enfatizando la importancia del flujo de información descendente o *top-down* de la percepción y la acción. El PP implica la minimización del error predictivo, mismo que se genera de la comparación de las predicciones con las entradas sensoriales y que se procesa vía ascendente o *bottom-up* (Clark, 2015). Este proyecto está enmarcado dentro de la robótica cognitiva, un área de investigación que se inspira en teorías y modelos de disciplinas como la psicología y las neurociencias para investigar modelos computacionales biológicamente plausibles en agentes artificiales. Para realizar esta investigación se tomará como base el aprendizaje de máquina, en especial el aprendizaje profundo. Además, se retomarán los principios propuestos por la arquitectura cognitiva computacional denominada SOIMA (*Self-Organized Internal Models Architecture*, por sus siglas en inglés). La SOIMA integra los modelos internos -inverso y directo- en una sola arquitectura mediante el uso de mapas autoorganizables (SOMs, por sus siglas en inglés *Self-Organizing Maps*) que generan *clusters* o grupos de información de distintas modalidades sensoriales (visual, propioceptiva, táctil) o motoras. De esta forma, esta arquitectura permite modelar esquemas sensorimotrices multimodales en agentes (Escobar-Juárez et al., 2016). Mediante este proyecto se desea investigar e implementar el contexto y la tarea en un agente artificial. Esto se logrará basándose en la SOIMA y otras herramientas de aprendizaje profundo, buscando estudiar la importancia del contexto y la tarea en la planeación y ejecución de una acción.

1. Introducción

La robótica cognitiva es una disciplina que se interesa por el estudio de modelos computacionales biológicamente plausibles en agentes artificiales. Inspirada en teorías y modelos propuestos en áreas como la psicología cognitiva y las neurociencias, la robótica cognitiva busca estudiar las implicaciones de estos modelos bajo condiciones controladas. Al mismo tiempo, este campo de estudio posee el objetivo de dotar con capacidades cognitivas básicas a dichos agentes para que sean capaces de actuar en un mundo complejo de manera flexible (Pfeifer and Scheier, 2001).

Por lo anterior, el diseño e implementación de arquitecturas computacionales cognitivas es un área de investigación activa y de gran interés dentro de este campo. Inspiradas en distintos aspectos cognitivos y basadas en principios teóricos en ocasiones divergentes, las arquitecturas computacionales que han sido reportadas hasta ahora forman un espacio multivariante dentro de la robótica cognitiva. De igual modo, las herramientas computacionales implementadas también varían entre ellas, así como las tareas cognitivas a las cuales están enfocadas (Ye et al., 2018).

Recientemente, la teoría del PP ha generado gran interés dentro de la robótica cognitiva debido a que proporciona una base epistemológica para estudiar flujos de información distintos en las arquitecturas computacionales. El PP permite generar predicciones *top-down* de las salidas motoras basadas en la experiencia sensorimotora previamente acumulada. De acuerdo a este marco explicativo la información fluye jerárquicamente, de tal forma que las capas de más alto nivel generan predicciones de las capas de menor jerarquía. Asimismo, el error que se genera de la comparación de las predicciones con las entradas sensoriales es la única información que fluye *bottom-up* (Choi et al., 2018).

El marco del PP ha permitido generar habilidades como la coordinación visuomotora (Wijesinghe et al., 2018), la manipulación de objetos (Noda et al., 2013) y la navegación autónoma en agentes artificiales (Zhong et al., 2018). Además, debido a su propuesta de procesamiento jerárquico, ha permitido implementar herramientas de aprendizaje profundo, especialmente de redes neuronales convolucionales (CNNs, por sus siglas en inglés). Sin embargo, la principal aplicación del PP dentro de la robótica cognitiva se ha limitado a predicciones de salidas motoras a corto plazo (Choi et al., 2018).

Pese a esta gran diversidad de arquitecturas cognitivas computacionales, existen pocas que codifiquen la tarea y el contexto de manera explícita en su estructura. La información acerca de la tarea es especificada por el programador en la mayoría de los casos, mostrando al agente artificial el objetivo deseado de la tarea de manera externa (Escobar-Juárez et al., 2016). No obstante, dotar a agentes artificiales con la capacidad de codificar el contexto y la tarea que deben realizar podría significativamente incrementar sus potencialidades y su versatilidad de comportamientos. La principal aportación de esta investigación está centrada en el estudio del contexto y la tarea en la planeación y ejecución de acciones de un agente artificial.

1.1. Objetivo

Modelar una arquitectura computacional que codifique el contexto para brindar a un agente artificial la flexibilidad de elegir la tarea adecuada y ejecutar las acciones necesarias para lograr un objetivo, tomando como inspiración el marco teórico del procesamiento predictivo.

1.2. Preguntas de investigación

- ¿Qué papel juega el contexto en la elección de una tarea?
- ¿Cuál es la importancia del contexto y la tarea en la planificación de una acción para lograr un objetivo específico?

1.3. Hipótesis

- Aprender y actualizar las predicciones sobre las consecuencias sensoriales de las acciones en un contexto específico optimizará la conducta de un agente artificial al brindar flexibilidad en la elección de la tarea.
- Las asociaciones multimodales aprendidas dentro de un contexto específico indicarán la secuencias de acciones necesarias para lograr un objetivo final, el cual es dependiente de la tarea y del contexto.

2. Procesamiento predictivo

Recientemente, una propuesta que desafía los marcos conceptuales de las teorías clásicas del procesamiento de la información del sistema nervioso ha surgido desde la neurociencia computacional. Esta teoría se denomina procesamiento predictivo y ha generado gran interés dentro de la robótica cognitiva debido a que enuncia una manera unificada de entender los procesos de la percepción y la acción, así como distintos procesos cognitivos (Clark, 2015).

De acuerdo a las posturas clásicas del funcionamiento del sistema nervioso, el principal flujo de información perceptual que se procesa se transmite de manera ascendente o *bottom-up*, desde los receptores sensoriales hasta el cerebro. A través de este proceso, la información sensorial entrante se integra en una representación cada vez más compleja conforme la información fluye hacia el cerebro. Estas posturas han delegado un papel únicamente de modulación y atencional al flujo descendente o *top-down* (Marr, 1982).

En contraste, la teoría del procesamiento predictivo propone que el cerebro se encuentra anticipándose constantemente al flujo de las entradas sensoriales generando predicciones de lo que detectará antes de que esta información sea procesada. De esta forma, el flujo *top-down* constará de predicciones. De la comparación de estas predicciones con las entradas sensoriales se generará un error de predicción. El PP propone que este error es la única información que será procesado vía *bottom-up*. De esta forma, el PP invierte el sentido en el que las teorías clásicas explican el procesamiento de la información, enfatizando que el mayor flujo de información será *top-down*.

El PP postula que las predicciones generadas vía *top-down* fluyen en una cascada jerárquica, de tal manera que las capas de mayor jerarquía generan predicciones de las capas de menor nivel. La cascada de predicciones es producida mediante uno o varios modelos generativos integrados en el cerebro. Estos modelos resultan capaces de extraer las propiedades invariantes de las entradas sensoriales. Las predicciones son generadas por los modelos mediante estimaciones estadísticas dado el conocimiento previo obtenido a través de la experiencia (Rao and Ballard, 1999).

De manera formal, existen diversas propuestas computacionales acerca de la manera

en la que el modelo generativo se actualiza en cuanto recibe información nueva. Existen propuestas de modelo generativo basados en una función gaussiana o en una secuencia de muestras previas de datos de una serie de tiempo, dependiendo de la perspectiva del estudio. Asismiso, durante el proceso de optimización se han implementado procesos como la minimización de la energía libre, la suma del error cuadrático, entre otros. A pesar de que todos estos algoritmos buscan ajustar el modelo a los datos, los procesos de optimización varían, así como su plausibilidad biológica (Spratling, 2017).

Una de las posturas más radicales del PP plantea que el procesamiento basado en predicciones de igual manera incluye al sistema motor; esta propuesta ha sido denominada inferencia activa. Este mecanismo propone que las predicciones generadas en el sistema motor están basadas en información sensorial propioceptiva. El término propiocepión se refiere al sentido de la posición relativa de las partes cercanas del cuerpo, así como la fuerza aplicada en un movimiento. De acuerdo a la inferencia activa, el comportamiento se puede entender en términos de la inferencia de las causas de estados propioceptivos, en donde las capas de más alto nivel generan predicciones propioceptivas en lugar de comandos motores. Esta propuesta está basada en la observación de las caracteríticas similares entre las proyecciones descendentes de la corteza motora y las proyecciones *top-down* de la corteza visual (Adams et al., 2013). De esta forma, el marco explicativo del PP integra a la percepción y a la acción en un proceso basado en predicciones, ambos con la misma finalidad de minimizar el error predictivo.

Una característica relevante dentro del PP es que este marco considera la importancia del contexto y la tarea en la que se encuentra el agente, ya sea biológico o artificial. Esto lo hace proponiendo un mecanismo mediante el cual el impacto que tienen diferentes señales de errores de predición puede ser modulado sistemáticamente mediante la certeza estimada o precisión de cada uno de ellos. La alteración del peso o ganancia de los errores de predicción seleccionados varían de acuerdo a la tarea y al contexto, de tal forma que los errores relevantes para la tarea presentarán alta precisión y, por lo tanto, presentarán un mayor impacto en el procesamiento de la información. De esta forma, las variaciones en los pesos asignados a cada error tienen la función de elegir qué predecir en cada situación. Aunque el mecanismo no se detalla finamente, esta teoría resulta interesante debido a su propuesta sensible al contexto y a la tarea (Clark, 2013).

Buscar inspiración en la importancia del contexto y la tarea de acuerdo a las propuestas del PP resulta una idea interesante dentro del marco de la robótica cognitiva, ya que permitiría dotar a los agentes artificiales con una capacidad de abstracción más elevada y ser capaces de generar una conducta más versátil al brindarles flexibilidad en la elección de la tarea de acuerdo a su contexto.

3. Robótica cognitiva

La robótica cognitiva es una disciplina que se interesa por el estudio de modelos computacionales biológicamente plausibles en agentes artificiales. El fundamento teórico para modelar procesos cognitivos está centrado en teorías de las ciencias cognitivas que enfatizan la importancia de la interacción del agente con su entorno para el desarrollo de procesos cognitivos, como el aprendizaje y la memoria (Pfeifer and Scheier, 2001).

De manera clásica, distintas corrientes de las ciencias cognitivas han entendido a los procesos cognitivos como un procesamiento de información abstracta, considerándolos como resultado de un proceso simbólico basado en reglas e independiente de las modalidades cerebrales encargadas de la percepción y la acción. Por otro lado, existen alternativas que dan un rol central al cuerpo en el desarrollo de la cognición. La idea sustancial de estos enfoques es que la función principal del cerebro es funjir como el sistema de control de un cuerpo que se encuentra en constante interación con su entorno (Wilson, 2002).

La cognición cimentada y la cognición corporizada se enfocan en el rol del cuerpo para el desarrollo de la cognición. Sumado a ello, la cognición cimentada enfatiza el rol de la *simulación interna* y la *re-enacción* de los procesos cerebrales. De acuerdo a esta postura, los estados perceptuales producidos por la visión, la audición y las demás modalidades sensoriales, así como los estados internos del cuerpo y la acción poseen correlatos neuronales a nivel de corteza cerebral. Estos correlatos o patrones de activación son denominados representaciones y se encuentran almacenados en la memoria. Las representaciones pueden corresponder a una única modalidad sensorial o motora o pueden ser multimodales al integrar información de varias de ellas (Barsalou, 2008).

De esta manera, los flujos de activación de las modalidades sensoriales y motoras se integran en una representación cerebral interna del cuerpo, denominada esquema corporal.

Siguiendo la postura de la teoría de la cognición cimentada, estas representaciones se adquieren durante la experiencia y pueden ser *re-activadas*, generando simulaciones internas durante los procesos de percepción y acción. Las representaciones multimodales, el esquema corporal y la simulación interna son considerados fundamentales para la planeación de acciones y para interactuar de manera eficiente con el entorno. Estos conceptos han sido clave dentro de la robótica cognitiva (Escobar-Juárez et al., 2016).

3.1. Modelos internos

La incertidumbre del entorno es un problema central en robótica cognitiva (Nguyen-Tuong and Peters, 2011). Para que un agente artificial logre interactuar con el entorno necesita aprender y adaptarse continuamente a nuevas tareas. La mayoría de las investigaciones sobre aprendizaje en agentes artificiales se han enfocado en el aprendizaje de tareas únicas que se aprenden de manera aislada (Nguyen-Tuong and Peters, 2011). Sin embargo, los agentes artificiales necesitan aprender las regularidades de tareas individuales y del entorno, y subsecuentemente, explotar este conocimiento para aprender nuevas tareas. A la vez, es importante investigar cómo la similitud entre las tareas puede utilizarse para generalizar este conocimiento a nuevas tareas o en contextos diferentes.

Cualquier agente racional debe decidir cómo manipular el entorno basándose en sus observaciones y predicciones de la influencia que ejerce en el sistema. Por lo tanto, el agente requiere considerar dos problemas principales: (1) necesita deducir la conducta del sistema con base a las observaciones previas y, (2) al poder inferir esta información necesita determinar cómo manipular el sistema para obtener las consecuencias sensoriales deseadas con relación al entorno.

El primero de estos problemas implica aprender a predecir la información faltante para completar el conocimiento sobre las acciones y las respuestas del sistema. Para comprender la conducta del sistema y cómo éste reacciona en torno a las acciones del agente, se necesita información sobre los estados sensoriales y las acciones (del pasado, presente y del estado deseado). Sin embargo, el agente tiene acceso limitado a la cantidad de observaciones aprendidas, por lo que necesita predecir la información faltante dada la información previamente aprendida.

Estudios sobre el desarrollo cognitivo en humanos sugieren que el sistema nervioso central simula internamente conductas dinámicas del sistema motor mediante el uso de modelos internos (modelo directo y modelo inverso) para planear, controlar y aprender acciones (Blakemore et al., 2000; Kawato, 1999; Wolpert et al., 1995). Los modelos internos también le otorgan a un agente la posibilidad de anticipar, predecir y planear conductas motoras basándose en simulaciones internas (Schillaci et al., 2012), unificando de una manera natural la información sensorial y motora para crear representaciones multimodales (Wolpert and Kawato, 1998).

Tomando esta evidencia como inspiración la investigación en Robótica cognitiva ha intentado responder a la pregunta sobre cómo un agente artificial puede desarrollar y aprender modelos internos de tal forma que pueda interactuar con el entorno de una manera eficiente. Por lo anterior, los modelos internos forman parte de una de las principales teorías para el aprendizaje y el control motor dentro de las ciencias cognitivas, especialmente en las neurociencias y en la robótica cognitiva. Dependiendo de la información que se requiere predecir es posible distinguir entre distintas arquitecturas: modelo directo, modelo inverso, modelos mixtos, y modelos de predicción de varios pasos (Nguyen-Tuong and Peters, 2011)

Desarrollar los modelos internos es básicamente establecer una relación causal entre los movimientos y los estados sensoriales. Existen dos variedades de los modelos internos, el modelo directo y el modelo inverso (Figura 26). El modelo directo predice las consecuencias sensoriales de una acción dada una situación sensorial inicial; este modelo también se conoce como predictor. Por otro lado, el modelo inverso genera el comando motor necesario para lograr un cambio deseado en la información sensorial dada una situación sensorial actual. A este modelo también se le ha denominado controlador. Estos modelos han sido implementados de manera conjunta en la robótica cogntiva en tareas como el control sacádico, el reconocimiento de acciones y la planeación motora (Wolpert and Kawato, 1998).

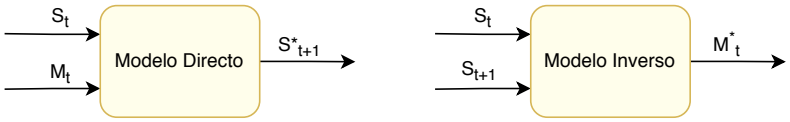

Figura 1: Representación del modelo directo e inverso.

Modelar el modelo directo y el modelo inverso de manera independiente puede llevar

a problemas sobre la redundancia cinemática en los agentes artificiales. En donde, según una configuración de la articulación q, la tarea de una posición en el espacio deseada x puede ser determinada con exactitud, pero puede haber muchas posibles configuraciones de q para ejecutar la tarea de la posición espacial deseada x. Por lo tanto, existen muchas configuraciones de las articulaciones q y el modelo inverso puede tener soluciones infinitas para llegar a la posición espacial x.

En este sentido, la relación entre el modelo inverso y el modelo directo puede contener soluciones inválidas que provocan un desempeño pobre en las predicciones. Para resolver este problema, (Jordan and Rumelhart, 1992) propusieron por primera vez, los modelos mixtos, en los cuales se combina el modelo directo y el modelo inverso de tal forma que la información codificada en el modelo directo puede resolver el problema de la redundancia cinemática del modelo inverso. Por lo tanto, cuando se combinan el modelo inverso y el modelo directo se genera una relación de identidad en donde el modelo inverso proporciona una solución única consistente con el modelo directo. Los modelos mixtos han causado gran interés y se han estudiado extensivamente en el campo de las neurociencias. (Kawato, 1999; Wolpert and Kawato, 1998).

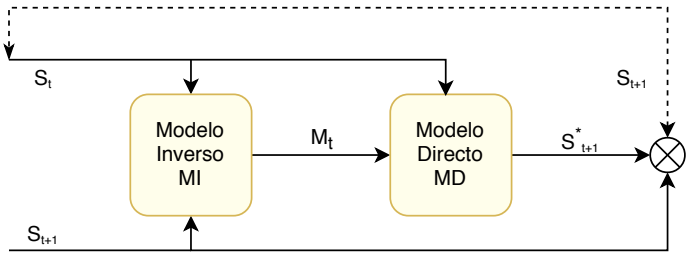

Figura 2: Modelos internos acoplados.

Por otra parte, se han propuesto modelos que permiten predecir varios estados futuros o acciones para tener información del sistema a largo plazo. De esta forma, el modelo de predicción de varios pasos genera una serie o secuencia de estados antes de ejecutarlos, mediante la predicción del comportamiento del sistema para los siguientes tiempos $t + n$ y no sólo para el siguiente paso en el tiempo $t + 1$. Lo anterior sin la disponibilidad de la retroalimentación del sistema en cada paso (Gaona et al., 2015; Möller and Schenck, 2008).

Los modelos internos son considerados idóneos para modelar procesos cognitivos dentro del marco de la cognición corporizada y cimentada. Por lo anterior, la robótica cognitiva ha buscado desarrollar arquitecturas computacionales capaces de integrar estos conceptos con la finalidad de permitir a los agentes artificiales adquirir sus propios esquemas sensorimotrices y llevar a cabo tareas cognitivas como el control sensorimotor de manera autónoma. El diseño de arquitecturas computacionales que integren estos conceptos no es trivial, ya que existen limitaciones en cuanto a las herramientas computacionales y la implementación de éstas en un mismo modelo. A continuación se presentan distintas arquitecturas que modelan estos conceptos desde distintas perspectivas.

3.1.1. Modelos internos autoorganizados: SOIMA

Una arquitectura cognitiva capaz de codificar los modelos internos, tanto el directo como el inverso, es la SOIMA (Escobar-Juárez et al., 2016). Su estructura está basada en una red de mapas autoorganizables (SOMs) en donde cada mapa codifica información de una modalidad dada, ya sea sensorial o motora. A su vez, estos mapas se conectan entre sí de tal forma que permiten integrar la información sensorial y motora en esquemas multimodales. La arquitectura implementa dos tipos de aprendizaje: (1) aprendizaje no supervisado basado en cuantización vectorial para cada SOM y (2) aprendizaje hebbiano para conectar los diferentes SOMs entre sí.

1. Aprendizaje no supervisado: Los SOMs son redes neuronales artificiales que generan *clusters* de información organizados topológicamente, revelando información similar entre los vectores de entrada. Estas redes implementan un mecanismo de aprendizaje no supervisado basado en la cuantización vectorial para generar dicha clasificación. La cuantización vectorial es un método que genera una aproximación a una distribución de probabilidad contínua de un vector variable dado un conjunto finito de entradas de dicho vector (Kohonen, 1990). En el caso de la SOIMA, los vectores son clasificados de acuerdo a la distancia euclidiana entre ellos y a su orientación, calculada a partir de su similitud coseno. Una vez evaluada la activación que genera la comparación de cada neurona con el vector de entrada se elige a la neurona ganadora, que será la que presente una menor activación. Posterior a este cálculo, los pesos de la red se modifican de acuerdo a una ecuación que está en función del producto de la distancia entre el vector de pesos y el vector de entrada por una tasa de aprendizaje que varía en función del tiempo, por una función de vecindad,

calculada a partir de la distancia entre cada nodo y la neurona ganadora.

2. Aprendizaje hebbiano: La asociación entre SOMs de distintas modalidades dentro de la arquitectura se establece de acuerdo a la regla de Hebb. Su postulado señala que *çuando el axón de una célula A se encuentra lo suficientemente cerca de la célula B para excitarla y participa persistentemente en hacerla disparar, un proceso de crecimiento o cambio metabólico en una o ambas células tiene lugar, de tal forma que la eficiencia de A como disparadora de B se incrementa"* (Hebb, 1949). La regla de Hebb que se implementa en la SOIMA tiene la finalidad de modificar los pesos de las neuronas ganadoras en cada SOM en función de una tasa de aprendizaje y la activación de cada uno de los nodos.

Como se observa en la Figura 2, los SOMs de cada modalidad, motora M y sensorial S en este caso, están conectados al SOM de mayor jerarquía MMR (*Multi-Modal Representation*, por sus siglas en inglés). Este SOM será el que codifique las representaciones multimodales. En un inicio, las conexiones entre los SOMs de cada modalidad y el de mayor jerarquía se establecen en cero. Posteriormente, se generan entradas en los SOMs de las modalidades S y M a partir de la interacción del agente con su entorno. Las coordenadas de los nodos ganadores dentro de cada SOM se almacenan en el SOM de mayor jerarquía. De esta forma, los esquemas sensorimotrices del agente artificial son codificados en el mapa MMR mediante el aprendizaje hebbiano.

Mediante la codificación de tripletas (Figura 3), la arquitectura es capaz de generar una predicción de una situación sensorial a partir de la entrada de la información sensorial actual y un comando motor que genera el cambio sensorial. Esto ocurre cuando es necesario implementar un modelo directo. Por el contrario, para generar un comando motor mediante el modelo inverso, las entradas de la red corresponderán a la situación sensorial actual y a la situación sensorial deseada.

La forma en la que se asocian los distintos SOMs dentro de la arquitectura le proporcionan a la SOIMA la interesante propiedad de ser bi-direccional. Su estructura facilita que el modelo inverso o el modelo directo puedan ser implementados de acuerdo a lo que sea necesario. Adicionalmente, la SOIMA permite implementar el modelo inverso y directo de forma acoplada, de tal forma que la salida de un SOM actúe como la entrada de otro. Las entradas a estos modelos pueden proceder del entorno o de la misma arquitectura. Además,

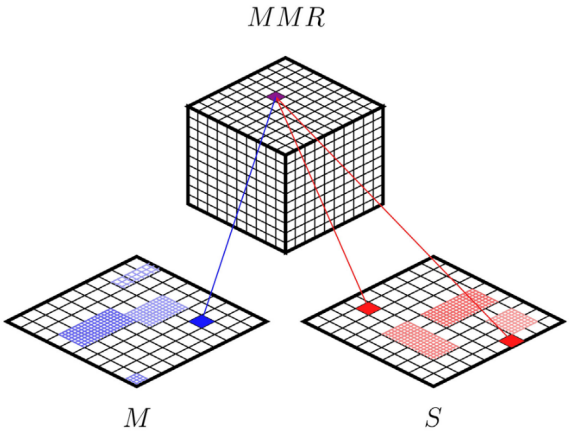

Figura 3: SOIMA: *Self-Organized Internal Models Architecture*. Arquitectura basada en SOMs (Escobar-Juárez et al., 2016).

los esquemas sensorimotrices pueden acoplarse mediante la integración de varios SOMs que codifican representaciones multimodales.

Esta arquitectura se ha implementado en tareas de control visuomotor y coordinación ojo-mano. El modelo debe predecir los cambios necesarios para alcanzar el objetivo final dado un contexto inicial que es indicado por la información sensorimotora actual. A pesar de ello, la tarea que debe ejecutar el agente artificial no se codifica dentro del modelo. El objetivo que debe alcanzar el agente es indicado por el experimentador. Sin embargo, la propiedad bi-direccional de la SOIMA hacen de ésta una arquitectura ideal para modelar procesos dentro del marco del procesamiento predictivo.

3.1.2. Modelos internos basados en propiocepción

Debido a que generalmente los seres humanos desarrollan habilidades nuevas en un corto plazo y que la habilidad adquirida es robusta en contextos muy versátiles, los modelos artificiales de aprendizaje inspirados en estos mecanismos de aprendizaje ofrecen una alternativa prometedora para la robótica cognitiva (Asada et al., 2001).

Considerando que en los humanos la propiocepción juega un papel importante para el desarrollo de los modelos internos para anticipar la posición del cuerpo y la fuerza aplicada durante cualquier conducta motora, es importante modelarla tomando en cuenta los aspectos fundamentales de esta modalidad sensorial (Zhang et al., 2018). De particular interés

para el estudio de la propiocepción en humanoides, Schillaci et al. (2012) propusieron una asociación de las articulaciones del brazo y del cuello con la posición estimada de la mano en 3D para establecer el modelo directo e inverso en un robot humanoide NAO. Sin embargo, para modelar la propiocepción en humanoides, no se ha prestado especial atención a los mecanismos subyacentes de la propiocepción en los seres humanos en términos de cómo estos modelos internos se desarrollan.

El término propiocepción fue introducido por Sherrington (1907) y se define como la sensación de la posición relativa del cuerpo, así como la fuerza y el esfuerzo aplicado durante el movimiento. La propiocepción es una modalidad sensorial interoceptiva debido a que informa los estados internos del cuerpo. Por lo tanto, la propiocepción es una modalidad sensorial que brinda información relevante a los organismos sobre la posición y la fuerza ejercida en cada parte de su cuerpo. La propiocepción involucra distintos receptores eferentes, llamados propioceptores, los cuales están conformados, principalmente, por receptores musculares y articulares.

A partir de investigaciones realizadas con infantes, se ha sugerido que el aprendizaje para alcanzar y agarrar un objeto es un proceso corporizado que requiere que los infantes adquieran la sensación de propiocepción antes de que sean capaces de relacionar causalmente sus acciones con los objetivos deseados. Corbetta et al. (2014), mediante el rastreo de movimientos oculares de infantes hacia los objetos, reportaron cómo aprenden a relacionar la sensación propioceptiva de la posición de su mano y la sensación táctil, junto con la mirada dirigida hacia el objeto deseado. Para lograr esto, primero es necesario que los infantes aprendan a mover voluntariamente su brazo en el espacio para obtener retroalimentación propioceptiva y táctil, de tal forma que puedan asociar las consecuencias sensoriales de sus acciones. Por lo tanto, los infantes relacionan la información visual del objeto deseado con la retroalimentación táctil en las experiencias propioceptivas.

En el estudio de Luo et al. (2016), se reporta una simulación de los resultados del trabajo de Corbetta et al. (2014) en un robot humanoide PKU-HR6.0, mostrando su efectividad dentro del campo de la robótica cognitiva. El agente artificial aprendió el modelo inverso relacionado la información visual de la posición de su brazo con la 'sensación de propiocepción'. En general, Luo et al. (2016) encontraron que modelar la 'sensación de propiocepción' ayudó a desarrollar un modelo inverso más eficiente en comparación con

los modelos tradicionales, que consideran únicamente los valores de los ángulos de las articulaciones del robot como entrada del modelo inverso. Estos resultados son consistentes con la manera en que los humanos aprenden los modelos internos basados en propiocepción.

Retomando las ideas de Luo et al. (2016), Zhang et al. (2018) propusieron modelos internos basados en propiocepción para el control motor del brazo de un robot humanoide PKU-HR6. Para modelar la propiocepción se consideraron dos características importantes de la propiocepción en los humanos. La primera consideración es que la información propioceptiva en términos de ángulos de las articulaciones es desconocida por los humanos a nivel consciente (no se piensa el movimiento en términos de la fuerza necesaria aplicada a cada músculo), lo que implica que el aprendizaje propioceptivo es no supervisado. Aunado a ello, la propiocepción debe ser capaz de codificar el estado sensorial del brazo; además, en los robots la información propioceptiva debe ser transferible a la información de las uniones de los servomotores.

Basándose en ambas consideraciones, generan el modelo directo e inverso usando una red neuronal profunda tipo *autoencoder*. Esta red neuronal aprende a producir como salida la información que recibe de entrada. De esta forma, los ángulos de las articulaciones están representados en la entrada y la salida de la red, mientras que la información propioceptiva es codificada en la capa oculta de la misma. Así, la información propioceptiva puede reconstruir nuevamente la información de las articulaciones (Figura 4). Al extraer las características de los estados de las articulaciones, la capa oculta del *autoencoder* debería ayudar a reducir la influencia del ruido y, en este sentido, podría considerarse una alternativa más eficiente para modelar propiocepción en agentes artificiales.

El modelo directo propuesto en Zhang et al. (2018) asocia la información propioceptiva de las articulaciones del brazo S_q con la posición de la mano p, en donde S_q considera la información propioceptiva del total de las articulaciones. Este modelo se denominó Modelo Directo Neuronal basado en Propiocepción (PNFM, por sus siglas en inglés). En contraste, en el Modelo Directo Neuronal (NFM, por sus siglas en inglés) la asociación se realiza utilizando directamente la información de los ángulos de las articulaciones q (Figura 5).

Por otro lado, el modelo inverso implica una inversión en dichas asociaciones. En el modelo inverso tradicional, llamado Modelo Inverso Neuronal (NIM, por sus siglas en inglés),

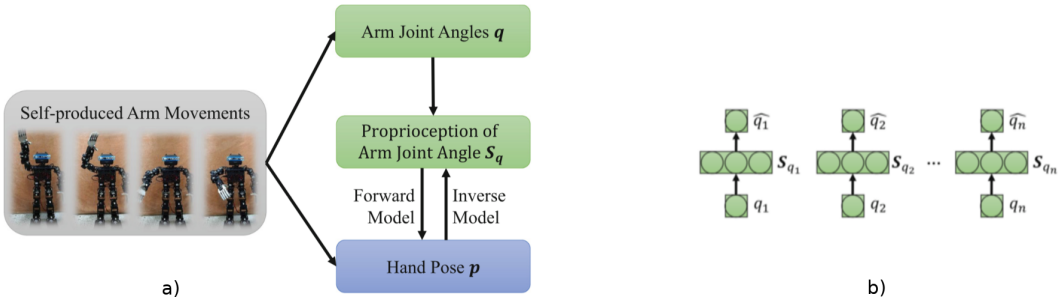

Figura 4: Arquitectura propuesta para modelos internos basados en propiocepción. a) Flujo de información del modelo directo e inverso de la arquitectura b)esquema representativo de la estructura *autoencoders* para codificar propiocepción para cada articulación (Zhang et al., 2018).

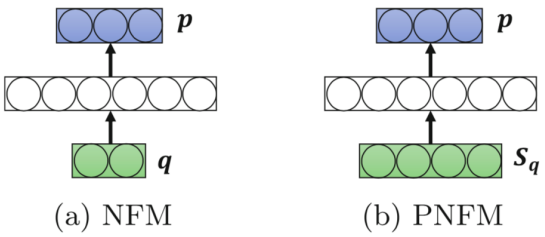

Figura 5: Representación de dos modelos directos utilizando autoencoders (Zhang et al., 2018).

el aprendizaje implica asociar la posición de la mano p con los ángulos de las articulaciones del brazo q. En Zhang et al. (2018) propusieron relacionar la posición de la mano p con la propiocepción de las articulaciones Sq para reemplazar los ángulos q de las articulaciones. A este modelo lo llamaron Modelo Inverso Neuronal basado en Propiocepción (PNIM). Como se mencionó anteriormente, la redundancia cinemática hace que el modelo inverso sea más complicado y, por lo tanto, más difícil de aprender.

Inspirados en Riemann and Lephart (2002), quienes propusieron que el movimiento de una articulación induce el movimiento de otra, y que diferentes articulaciones juegan un papel dominante en distintas direccione, Zhang et al. (2018) sugirieron un Modelo Inverso de cascada basado en Propiocepción (cPNIM, por sus siglas en inglés). En este modelo, la propiocepción de las articulaciones S_{q1} es controlado directamente por la posición de la mano p, y en consecuencia S_{q2} es controlado tanto por la posición de la mano p como por S_{q1}, y así sucesivamente para cada S_{qn}.

En el estudio presentado por Zhang et al. (2018) se probaron las distintas arquitecturas basadas en propiocepción. Primero, compararon el Modelo Directo Neuronal basado en Propiocepción (PNFM) con el clásico Modelo Directo Neuronal (NFM) para verificar los beneficios de usar la información propioceptiva en lugar de los valores de los ángulos de las articulaciones. Encontraron que PNFM obtuvo errores significativamente menores en la predicción de la posición de la mano p en comparación de la NFM.

A la vez, pusieron a prueba las distintas arquitecturas del modelo inverso (Figura 6). La salida del modelo inverso implica predecir los ángulos de las articulaciones necesarios para llevar a la mano a la posición deseada. En este sentido, para evaluar el desempeño del modelo inverso, las articulaciones del robot se establecen según la salida del modelo inverso, y la diferencia entre la posición de la mano actual y la deseada es calculada. Se observó que el Modelo Inverso Neuronal basado en propiocepción (PNIM) tuvo un mejor desempeño que el Modelo Inverso Neuronal (NIM), lo que indica que la propiocepción es efectiva para mejorar la precisión del modelo inverso. Adicionalmente, en el modelo de cascada (cPNIM) se observó una media del error menor que la obtenida con PNIM, lo que significa que el mapeo inverso de cada articulación en forma de 'cascada' mejora aún más el desempeño. En conjunto, estos resultados corroboran la propuesta de Riemann and Lephart (2002).

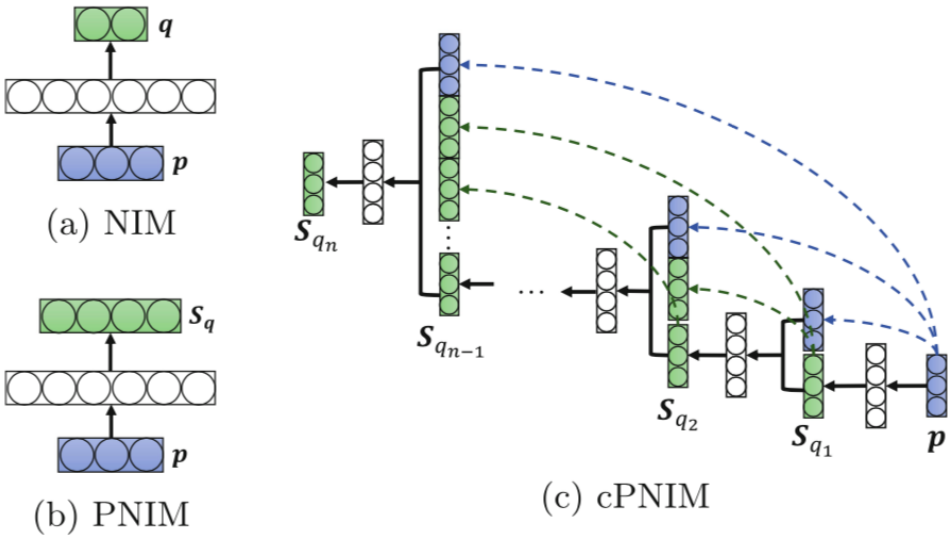

Figura 6: Representación de distintos modelos inversos utilizando autoencoders (Zhang et al., 2018).

Como conclusión, Zhang et al. (2018) sugieren utilizar la capa oculta de la red neuronal *autoencoder* para codificar propiocepción en agentes artificiales. En este sentido, señalan que utilizar la información propioceptiva en lugar de los comandos motores mejora el desempeño de las predicciones realizadas, tanto por el modelo directo como el inverso. Adicionalmente, sugieren utilizar el modelo inverso de cascada basado en propiocepción (cPNIM) para obtener un mejor desempeño en términos de la diferencia entre la posición actual y una posición deseada. Los resultados que obtuvieron indican que la red neuronal *autoencoder* funciona para aprender y modelar en un agente artificial la manera en la que se aprende la propiocepción en humanos. Finalmente, es importante mencionar que estos resultados concuerdan con el marco teórico propuesto por el PP al modelar la propiocepción considerándola como una modalidad sensorial más y no sólo como comandos motores, así como al proponer una arquitectura neuronal en cascada, la cual implica predecir las acciones necesarias en términos de los estados anteriores y de las consecuencias sensoriales deseadas.

4. Modelación computacional del contexto en la Robótica cognitiva

A pesar de ser uno de los conceptos más usados en distintas disciplinas, entender el significado de contexto no resulta trivial. Actualmente, el significado aceptado de manera general es que el contexto es el *conjunto de circunstancias que enmarcan o delimitan un evento o un objeto* (Bazire and Brézillon, 2005). Pese a ello, la definición usada en diferentes investigaciones puede variar de acuerdo al área o a la tarea de interés. En el campo de la robótica cognitiva, una definición de contexto que resulta de gran utilidad es la siguiente:

el contexto es una configuración distinguible de las características relacionadas con el entorno, la tarea y el agente que tiene un poder predictivo para el comportamiento (Turner, 1998).

Las características relacionadas con el entorno incluyen la información del espacio físico que rodea al agente artificial, como la ubicación de los objetos. Las propiedades relacionadas con la tarea incluyen el tipo de tarea que realiza el agente, así como las restricciones que ésta implica. Por otro lado, las características relacionadas con el agente incluyen el estado interno del robot, como los ángulos de sus articulaciones, la posición de su cabeza, entre otros. Aunado a ello, se ha considerado que la información contextual no es únicamente estática, sino que varía contínuamente en tiempo real, lo que resulta fundamental para la planeación y ejecución de acciones (Lee and Kim, 2018).

Pese a esta definición, el contexto es un tema relativamente reciente en la robótica cognitiva y no existe un consenso respecto a su definición. Dado que el contexto influye en casi todos aspectos del comportamiento, generar arquitecturas computacionales que modelen este concepto dotaría a los agentes artificiales de una mayor versatilidad y flexibilidad conductual. A continuación, se describirán dos estudios que han sido elegidos debido a su cercanía con el concepto de contexto mencionado previamente. Además, una de las arquitecturas que se describirá está inspirada en el marco del PP.

4.1. Modelación del contexto mediante SOMs

Recientemente, se ha reportado una arquitectura capaz de predecir el efecto de un conjunto de acciones realizadas con distintas herramientas (Mar et al., 2018). Este estudio

toma en cuenta la orientación de la herramienta, así como el agarre. La arquitectura consta de dos SOMs, cada uno para codificar la posición de la herramienta, así como los efectos de la acción ejecutada con ella, respectivamente.

Las diferentes orientaciones de la herramienta, así como la posición de la mano del agente artificial al sostenerla son codificadas en un mismo SOM. Por otro lado, la codificación de los efectos de la acción en el segundo SOM incluye a la acción y su efecto generado. La acción consta de arrastrar un objeto y se codifica con el ángulo de arrastre, mientras que su efecto está representado en términos del desplazamiento logrado del objeto. La asociación entre ambos mapas se realiza mediante un proceso de regresión, de tal forma que las coordenadas de cada nodo del SOM que codifica la posición de la herramienta se relacionan con las coordenadas de los nodos que especifican los efectos de la acción a través de una red neuronal de función radial (Figura 7).

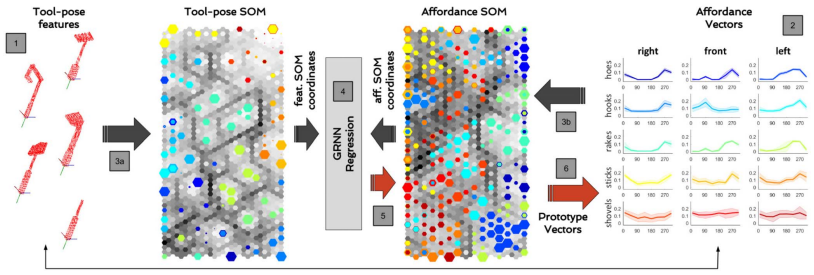

Figura 7: Diagrama propuesto para modelar posibilidades de acción de acuerdo a la orientación de la herramienta. Cada SOM codifica la posición de la herramienta y los efectos de la acción, respectivamente. Las coordenadas de ambos SOMs son relacionados mediante la implementación de una red neuronal de regresión generalizada (Mar et al., 2018).

El experimento consta de tres fases: recopilación de datos, entrenamiento y prueba del modelo. Durante la primera fase, el experimentador coloca la herramienta en la mano del agente, el cual está programado para que automáticamente detecte la orientación de la misma. Posteriormente, el agente realiza acciones exploratorias para observar los efectos de cada una de ellas sobre el objeto. Durante la fase de entrenamiento, todos los vectores de información recopilados son usados para que la red neuronal de función radial aprenda a predecir las coordenadas de los nodos de los SOMs que indican el efecto de una acción dada cierta posición de la herramienta. Finalmente, durante la fase de prueba el agente artificial es capaz de predecir las posibilidades de acción de cada posición de la herramienta y usar

esta asociación para elegir la mejor acción para una tarea dada.

Sin embargo, en este experimento la fase de aprendizaje no se realiza mediante la interacción del agente con la herramienta en línea. Además, las acciones están predeterminadas, así como la posición inicial, ya que para ejecutar el arrastre, la herramienta se coloca encima del objeto y posteriormente se dirige hacia el plano de la mesa donde se realiza la tarea, con la finalidad de que el agente no empuje el objeto. Aunado a ello, la posición de la herramienta se determina de manera automática.

4.2. Modelación del contexto utilizando aprendizaje profundo

De acuerdo a Choi et al. (2018), el PP permite generar modelos internos para diferentes tareas mediante la asociación de las intenciones de interacción del agente con su entorno con las secuencias perceptuales causadas por dichos estados intencionales. En la investigación citada, se propone un modelo de red neuronal recurrente profunda basada en la codificación predictiva capaz de generar un plan visuomotor dado un objetivo. El modelo es capaz de aprender, reconocer y generar secuencias visuomotoras mediante la minimización del error de predicción durante los tres procesos.

La red consta de dos vías de procesamiento, una designada para la visión y otra para la propiocepción. La entrada de la vía visual consiste en imágenes, mientras que la vía encargada de la propiocepción procesa información de los ángulos de las articulaciones. Cada vía de procesamiento consta de tres capas ordenadas de manera jerárquica y el patrón de conexión entre ellas se determina de tal forma que cada capa se conecta con sus capas vecinas, superior e inferior. La información procesada por ambas vías se integra en una última capa de mayor jerarquía; aunado a ello, ambas vías de procesamiento presentan conexiones laterales entre ellas, lo que permite generar información de una modalidad a partir de la otra. La información que fluye de manera descendente transmite predicciones de las capas de menor jerarquía, mientras que la información ascendente transmite el error de predicción (Figura 8).

No obstante, la generalización de la planeación que se logra con este modelo está limitada por el número de patrones sensorimotores utilizados durante el entrenamiento. Aunado a ello, durante la tarea de agarre el agente artificial no logra cumplir el objetivo debido a la falta de resolución de la imagen de cada uno de los patrones de entrenamiento. Asimismo,

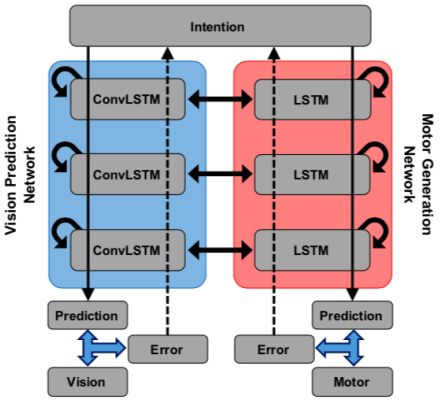

Figura 8: Modelo de red neuronal recurrente profunda para predecir secuencias visuomotoras (Choi et al., 2018).

el modelo no tiene la posibilidad de adaptarse a los cambios de contexto, ya que la configuración inicial del agente está predefinida(Figura 9) .

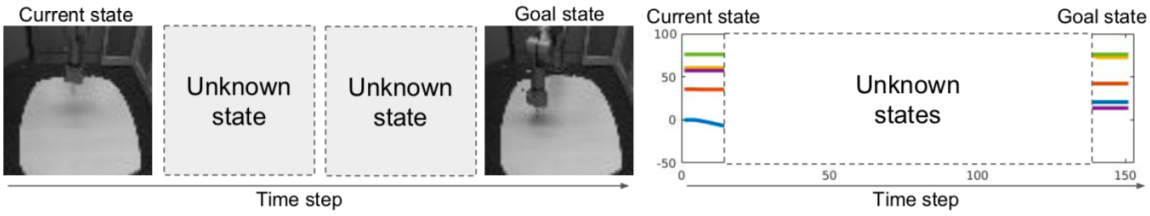

Figura 9: En la ilustración se muestran las condiciones que son necesarias para que el modelo pueda predecir la secuencia visuomotora

A pesar de que actualmente existen enfoques que han permitido dotar a agentes artificiales con la capacidad de realizar distintas tareas, el concepto de contexto no ha sido estudiado a profundidad en ellos. El presente proyecto busca modelar el contexto y la tarea en una arquitectura computacional. En la siguiente sección se propone un modelo que considera estos conceptos.

5. Propuesta de un modelo sobre el contexto y la tarea en Robótica cognitiva

Pese a la gran diversidad de arquitecturas cognitivas computacionales que han sido propuestas desde la robótica cognitiva, existen pocas que codifiquen la tarea y el contexto de manera explícita en su estructura. No obstante, dotar a agentes artificiales con la capacidad de codificar el contexto y la tarea que deben realizar podría significativamente incrementar sus potencialidades y su versatilidad de comportamientos. La principal aportación de esta investigación está centrada en el estudio del contexto y la tarea en la planeación y ejecución de acciones de un agente artificial, tomando como inspiración el marco del procesamiento predictivo. El modelo que se propone en este proyecto para el estudio de estos conceptos en la planeación y ejecución de acciones se ilustra en la Figura 10.

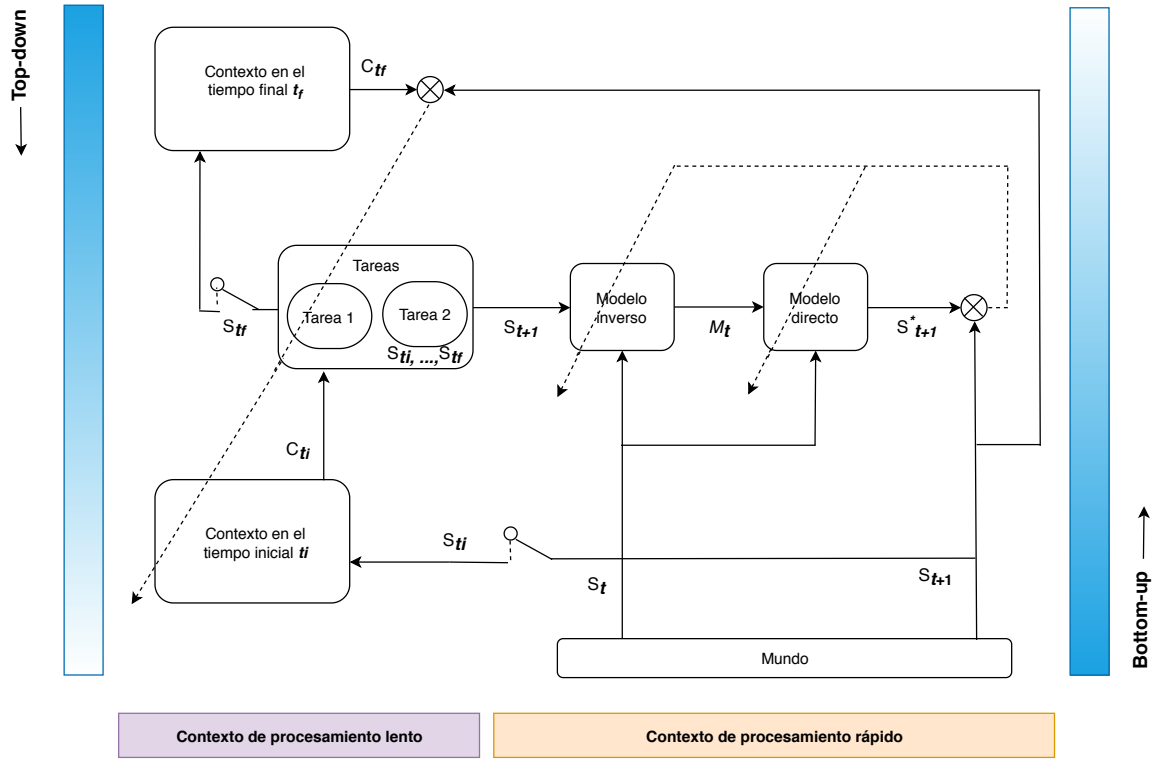

Figura 10: Diagrama del flujo de información del contexto y la tarea

Tomando inspiración del PP, en el modelo existe flujo de información vía top-down y bottom up. La información S_t que proviene del mundo comienza su proceso al convertir-

se en el estado inicial S_{ti} que entrará a la capa de Contexto inicial. La información que recibe esta capa corresponderá al estado inicial de la tarea. La información del contexto inicial se dirige a la capa de Tarea, para activar alguna de ellas de acuerdo al contexto. El módulo o capa de Tarea tiene representada las trayectorias o conjunto de estados necesarios para cada tarea. Una vez que la tarea ha sido activada por el contexto inicial, el estado final de la tarea sale del módulo Tareas y activa la capa superior de contexto final, indicando el estado final deseado de la tarea. El estado final deseado, que sale de la capa de Contexto en el tiempo t_f servirá como comparador de las predicciones realizadas a partir de los modelos internos acoplados, indicando si se ha llegado al estado final deseado o no.

A su vez, el modelo inverso recibe como entradas el siguiente estado deseado S_{t+1} asociado con la tarea, así como la información S_t que proviene del mundo. Este modelo genera una predicción del movimiento denominado M_t, necesario para cambiar de un estado al otro. Por su parte, el modelo directo recibe como entrada la predicción M_t del modelo inverso, así como el estado inicial de la tarea, proveniente del mundo. Con esta información, el modelo directo realiza una predicción del cambio de estado producido al ejecutar el movimiento M_t.

Una vez que se ejecuta el movimiento M_t en el mundo, el siguiente estado generado S_{t+1} se compara con la predicción $S*_{t+1}$ realizada por el modelo directo. Si el error es grande, los modelos internos se modifican para corregirlo. Si la predicción y el estado real son iguales, entonces el estado del mundo S_{t+1} se compara con el contexto final, para verificar que este estado se acerca al estado final deseado o para corroborar que ya se ha alcanzado el estado meta. En caso de que no se haya alcanzado el estado final deseado mediante las predicciones, se procede a evaluar el contexto inicial nuevamente para reiniciar el proceso.

El modelo propuesto considera la planeación en línea, comparando cada predicción con el estado final deseado. No obstante, el modelo puede adaptarse para realizar predicciones a largo plazo. El flujo de información dentro del modelo para generar predicciones a largo plazo se muestran en la Fig. 11. La estructura general sigue el mismo principio que el modelo de procesamiento en línea, a excepción del flujo de los comparadores. Como puede observarse, el estado S_{t+1} del mundo ya no es el que se comparará con las predicciones del modelo directo. En contraste, la comparación se realizará con el estado S_{t+1} que proviene de la capa de Tarea en un primer paso. A su vez, las siguientes predicciones estarán basadas en la primera predicción realizada y no en el mundo. Esto se logra convirtiendo la predicción del

modelo directo en una nueva entrada del modelo inverso, comenzando nuevamente el ciclo de modelos internos asociados. Por su parte, el sistema indica cuando ha llegado al estado meta realizando una comparación entre las predicciones que se generan con los modelos internos y no con la información que viene del mundo, como en el caso del modelo de procesamiento en línea.

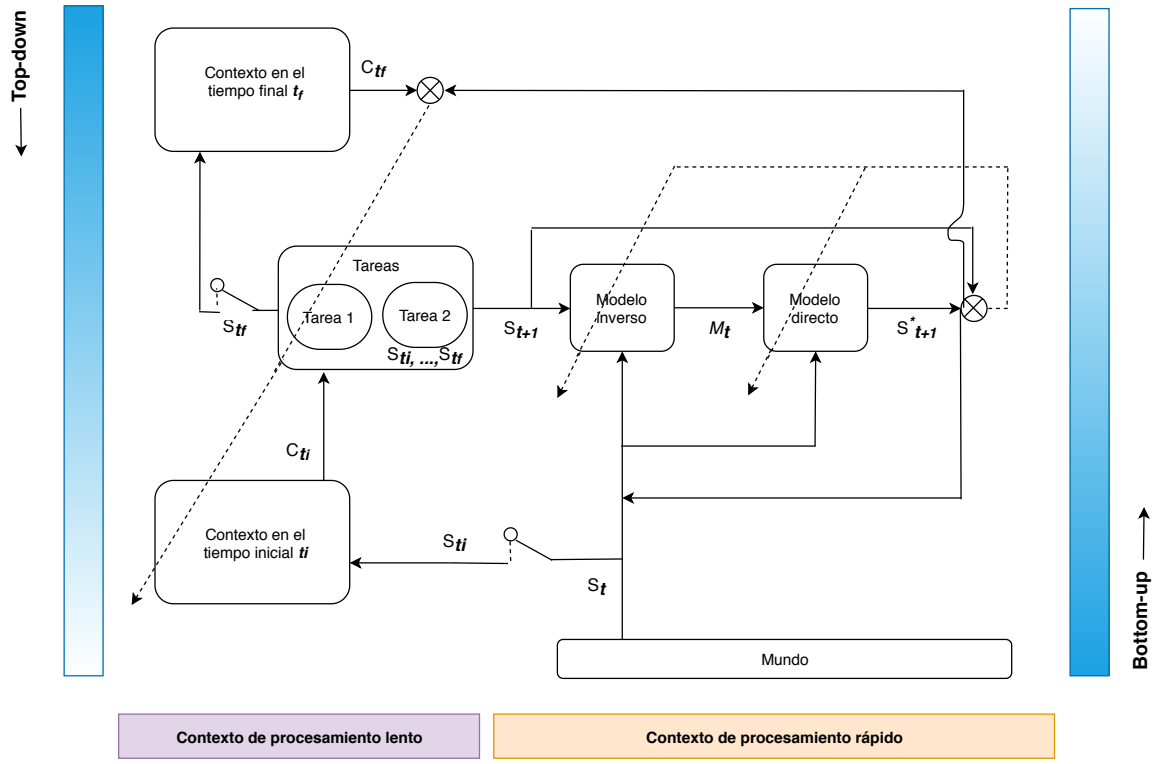

Figura 11: Diagrama del flujo de la información en el modelo de contexto y la tarea bajo predicciones a largo plazo.

En la siguiente sección se describen las tareas y contextos que servirán para la prueba del modelo propuesto.

6. Tareas y contextos experimentales

El modelo de contexto y tarea que se propone en este proyecto puede ser implementado bajo distintas condiciones experimentales. Las tareas y contextos que se han eligido para realizar la implementación del modelo se muestran en la Fig. 12. El experimento consta

de tres fases: dos fases de aprendizaje y una fase de prueba. Durante la primera fase se presenta el objeto 1 en el contexto 1, donde el objeto 1 refiere a una pelota y el contexto refiere a la forma geométrica del espacio donde ésta debe colocarse. Para mover la pelota desde la posición inicial a la posición final, la tarea que se ejecuta consiste en empujar el objeto. En la segunda fase del experimento, se presenta el objeto 2 en un segundo contexto reconfigurado, que incluye una plataforma sobre la cual debe colocarse el objeto. En esta fase el objeto 2 refiere a un cubo.

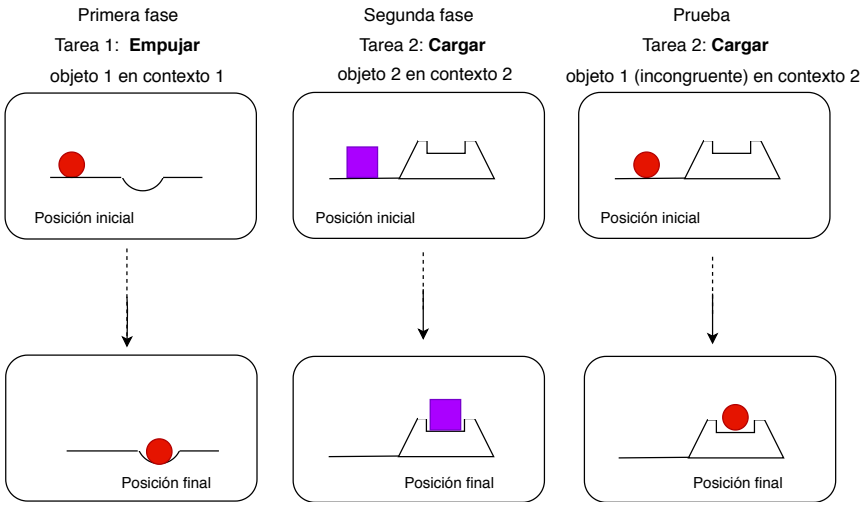

Figura 12: Fases experimentales de contextos y tareas.

De manera similar a la primera fase, en este contexto reconfigurado se debe realizar una tarea para mover el cubo de posición. La tarea consiste en cargar el objeto, ya que la posición final se ubica sobre la plataforma. Es así como cada tarea será asociada con un objeto en un contexto específico. Finalmente, la fase de prueba se basa en presentar el objeto 1 en el contexto 2, generando una situación incongruente. En esta fase el objeto 1 se denomina incongruente dentro de este contexto, ya que previamente el objeto sólo ha sido asociado con el contexto 1. La idea fundamental del experimento es asociar las acciones que se realizan con un objeto al contexto en el que éste se presenta. Por lo tanto, lo que se espera en la fase de prueba es que el objeto 1 sea asociado con la tarea que consiste en cargar, a pesar de que previamente sólo haya sido empujado.

Las tareas que han sido presentadas están inspiradas en experimentos realizados en el

campo de la neurociencia conductual. Fueron motivados, principalmente, por el estudio de Norman and Eacott (2005) para investigar la memoria del contexto en el que algunos objetos aparecen. El experimento que reportan para medir el efecto del cambio de contexto en la exploración de un objeto se muestra en la Fig. 13. De manera similar a las condiciones presentadas para la implementación del modelo propuesto, el experimento consta de tres fases: dos fases de exposición y una de prueba.

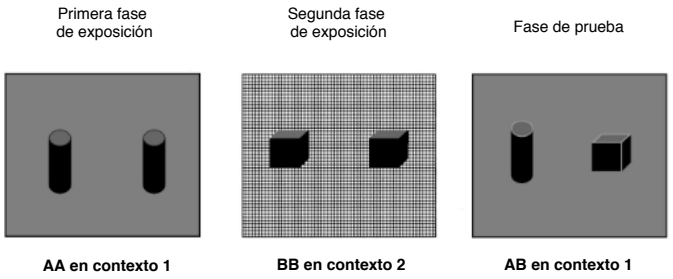

Figura 13: Esquema de las fases de un experimento para estudiar contexto. Figura adaptada de Norman and Eacott (2005).

No obstante, en el experimento propuesto por Norman and Eacott (2005) no se realizan acciones con los objetos presentados en cada contexto. Lo que los autores reportan son los tiempos de exploración que ratas Agouti oscuras pasan en cada objeto. Norman and Eacott (2005) señalan que el experimento examina la discriminación entre objetos de acuerdo a si un objeto familiar es encontrado en el mismo contexto o en un contexto distinto en relación a la experiencia previa con el objeto. Los autores reportan que las ratas sin lesiones en el hipocampo exploraron un mayor tiempo el objeto que había sido encontrado previamente en un contexto distinto con relación al objeto que fue encontrado en el mismo contexto y señalan que esto revela memoria de asociación contexto-objeto, dando sustento a hallazgos previos. Asimismo, su estudio reporta que la corteza postrinal tiene un rol crucial en la asociación objeto-contexto, mientras que otras áreas del hipocampo, como la corteza peririnal está involucrada en la memoria para objetos y sus partes constituyentes.

Por otro lado, en el caso del presente proyecto la tarea que debe realizarse en un momento dado está asociada con un objeto bajo un contexto específico. Para logarlo, la información contextual involucra espacios con formas geométricas distintas en donde deben ser colocados los objetos. Este diseño experimental asume que además de la memoria objeto-contexto, existe una asociación entre las acciones y el contexto en el que se ejecutan. En la Fig. 14

se representan las condiciones de la primera fase del experimento dentro del modelo de procesamiento en línea que fue presentado en la sección anterior.

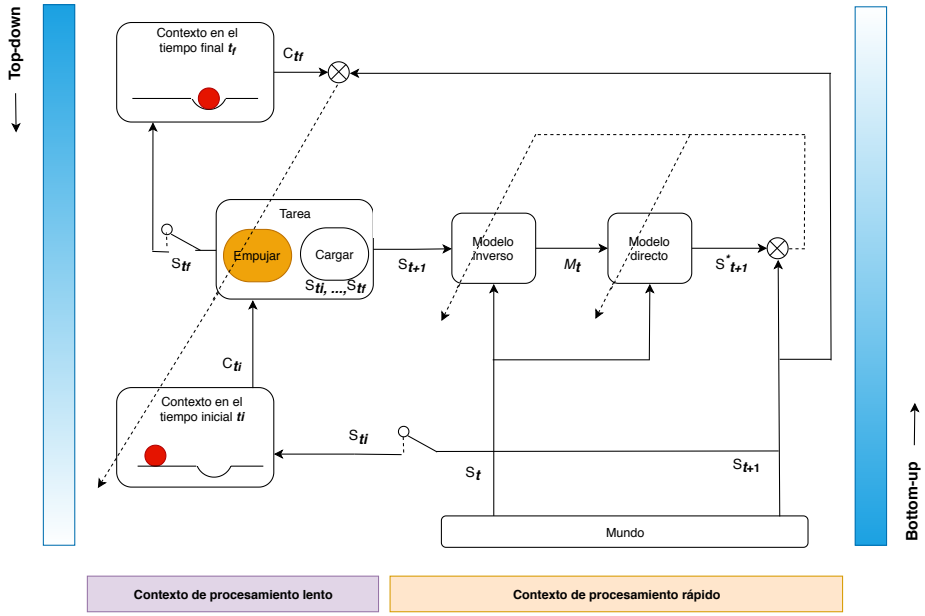

Figura 14: Adaptación del modelo de procesamiento en línea de contexto y tarea con las condiciones de la primera fase de aprendizaje: objeto 1 en contexto 1 con la tarea *Empujar*.

Como puede observarse en la Fig. 14, el contexto inicial está indicado por la posición inicial de la pelota, que en este caso se ubica lejos del espacio donde ésta debe ser colocada. Esta información viene del mundo y entra a la capa de contexto, donde será integrada como el estado actual que indica las condiciones iniciales de la tarea. La información obtenida en la capa de contexto se dirige a la capa de tarea y activa la acción empujar. El módulo Tarea tiene representada las trayectorias o conjunto de estados necesarios para mover el objeto hacia el lugar deseado de cada tarea. Dicho módulo contiene las trayectorias de ambas tareas; no obstante, en el caso de la primera fase la tarea que se activa por el contexto 1 y el objeto 1 corresponde a la tarea *empujar*.

Una vez que la tarea ha sido activada por el contexto, el estado final de la tarea sale del módulo Tareas y activa la capa superior de contexto final, indicando el estado final deseado. El estado final deseado, que sale de la capa de Contexto en el tiempo t_f, servirá como comparador de las predicciones realizadas a partir de los modelos internos acoplados,

indicando si se ha llegado al estado final deseado o no. A su vez, el modelo inverso recibe de la capa de Tareas el segundo estado deseado asociado con la tarea activada. El modelo inverso, que tiene como entradas el estado inicial y el siguiente estado deseado de la tarea, realizará una predicción del movimiento M_t que producirá el cambio entre dichos estados.

Por su parte, el modelo directo recibe como entrada la predicción M_t del modelo inverso y, el estado inicial de la tarea proveniente del mundo, para hacer una predicción del cambio del estado inicial cuando se aplica el movimiento M_t. Una vez que se ejecuta en el mundo el movimiento M_t predicho, el siguiente estado generado S_{t+1} se compara con la predicción $S*_{t+1}$ realizada.

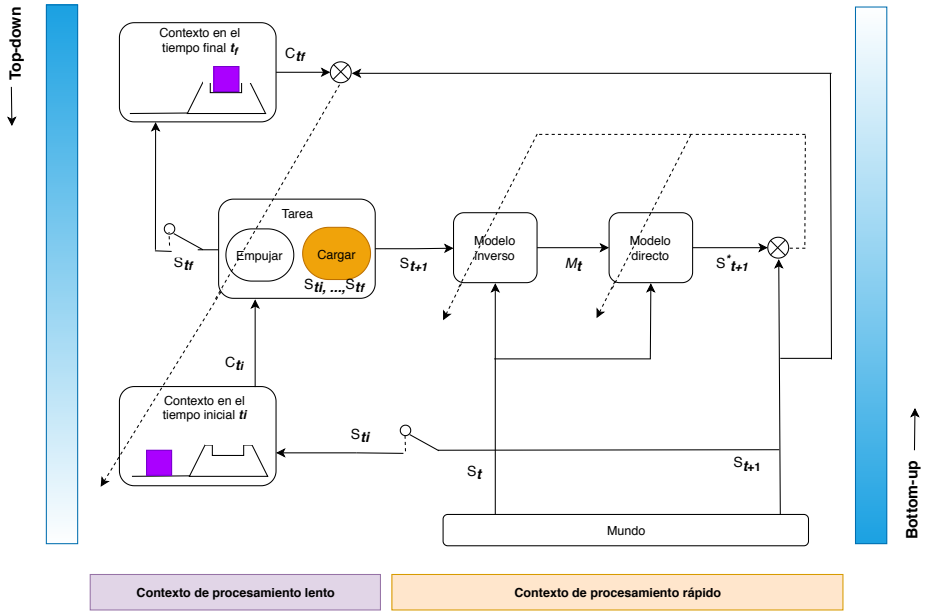

Figura 15: Adaptación del modelo de procesamiento en línea de contexto y tarea con las condiciones de la segunda fase de aprendizaje: objeto 2 en contexto 2 con la tarea *Cargar*.

Si el error es grande, los modelos internos se modifican para corregirlo. Si la predicción y el estado real son similares, entonces el estado del mundo se compara con el contexto final para verificar si el estado final deseado se acerca al siguiente estado producido mediante las predicciones o si ya se ha alcanzado dicho estado. En caso de que no se haya alcanzado el estado final deseado mediante las predicciones, se procede a evaluar el contexto inicial

nuevamente para reiniciar el proceso. La Fig. 15 muestra el mismo proceso con el objeto 2 (cubo) bajo el contexto 2 reconfigurado. La tarea que se activa para este contexto consta en *cargar* el objeto y colocarlo sobre la plataforma. Por su parte, la Fig. 16 muestra la fase de prueba. El objeto incongruente dentro de este contexto activará la tarea que ha sido previamente asociada con dicho contexto, por lo que ahora la pelota será cargada y dirigida hacia la plataforma, a pesar de que esta tarea no haya sido ejecutada con este objeto.

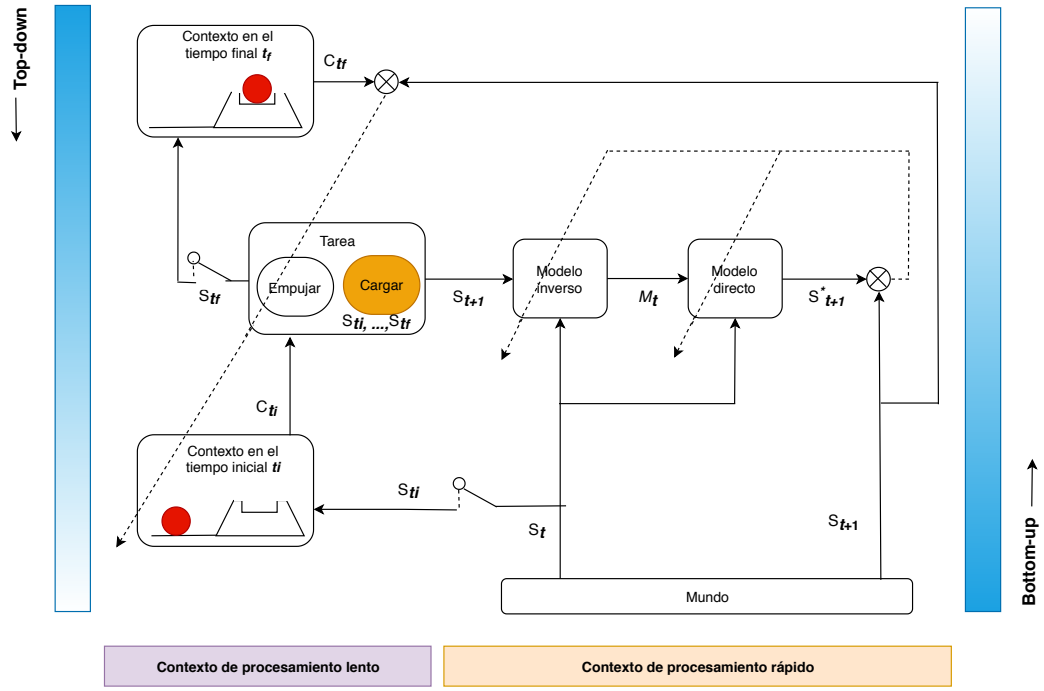

Figura 16: Adaptación del modelo de procesamiento en línea de contexto y tarea con las condiciones de la fase de prueba con situación incongruente: objeto 1 en contexto 2 con la tarea *Cargar*.

Las Figs. 17, 18 y 19 muestran el procesamiento de la información durante las tres fases del experimento en el caso donde el modelo implementado realiza predicciones a largo plazo. La activación de las tareas en cada contexto permanece igual, la única diferencia es que ahora las predicciones del sistema son las que serán comparadas con el estado final deseado en lugar de compararse con la información que proviene del mundo, como se detalló en la sección previa.

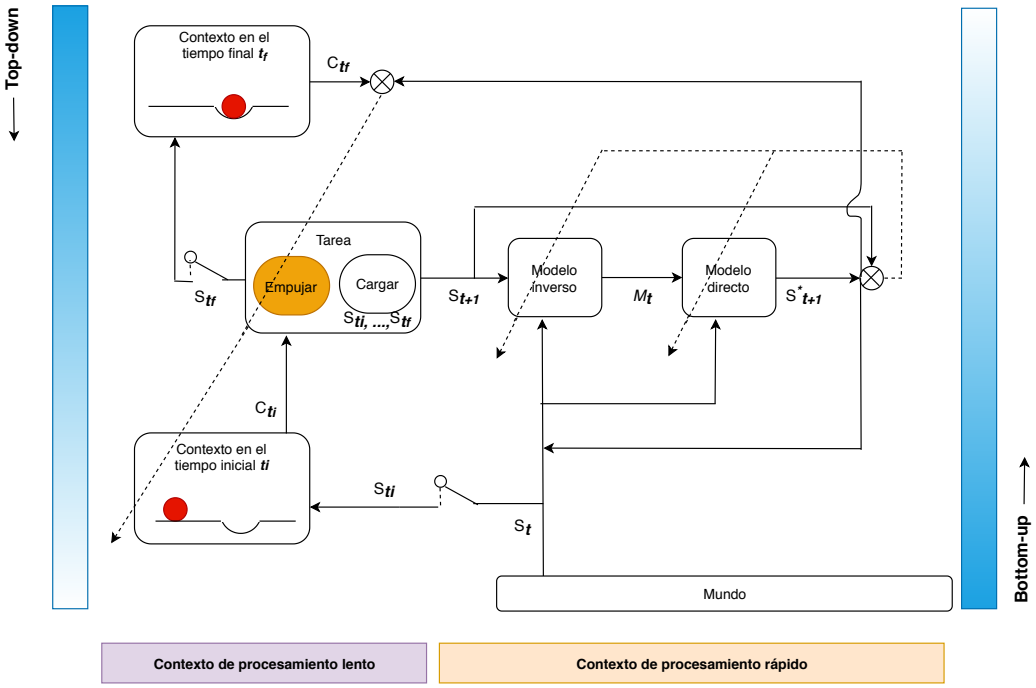

Figura 17: Adaptación del modelo de procesamiento con predicciones a largo plazo de contexto y tarea con las condiciones de la primera fase de aprendizaje: objeto 1 en contexto 1 con la tarea *Empujar*.

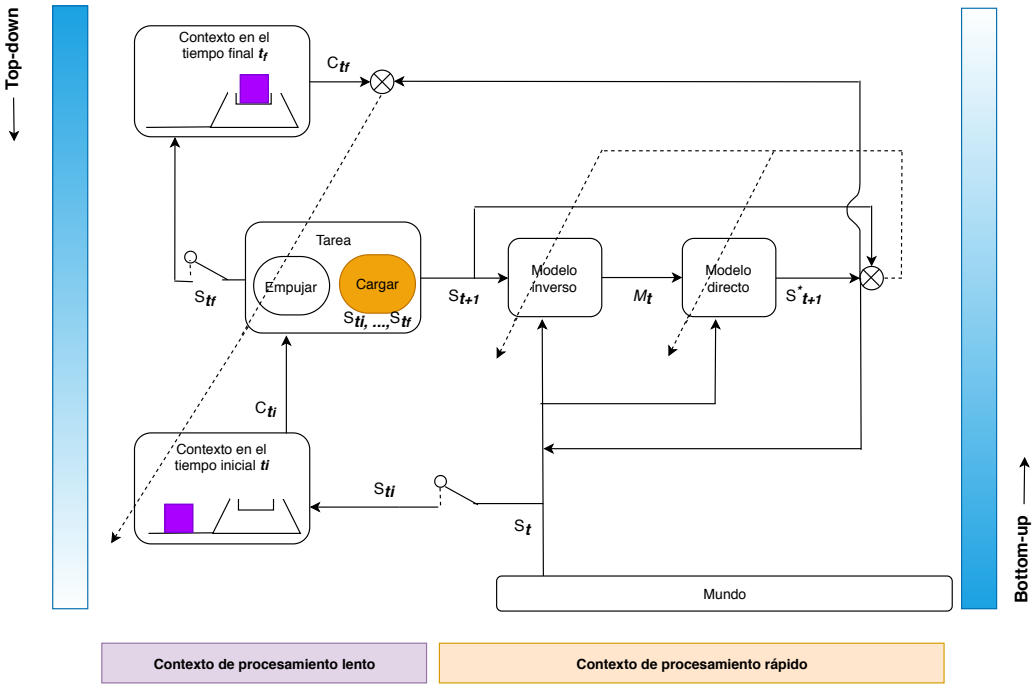

Figura 18: Adaptación del modelo de procesamiento con predicciones a largo plazo de contexto y tarea con las condiciones de la primera fase de aprendizaje: objeto 2 en contexto 2 con la tarea *Cargar*.

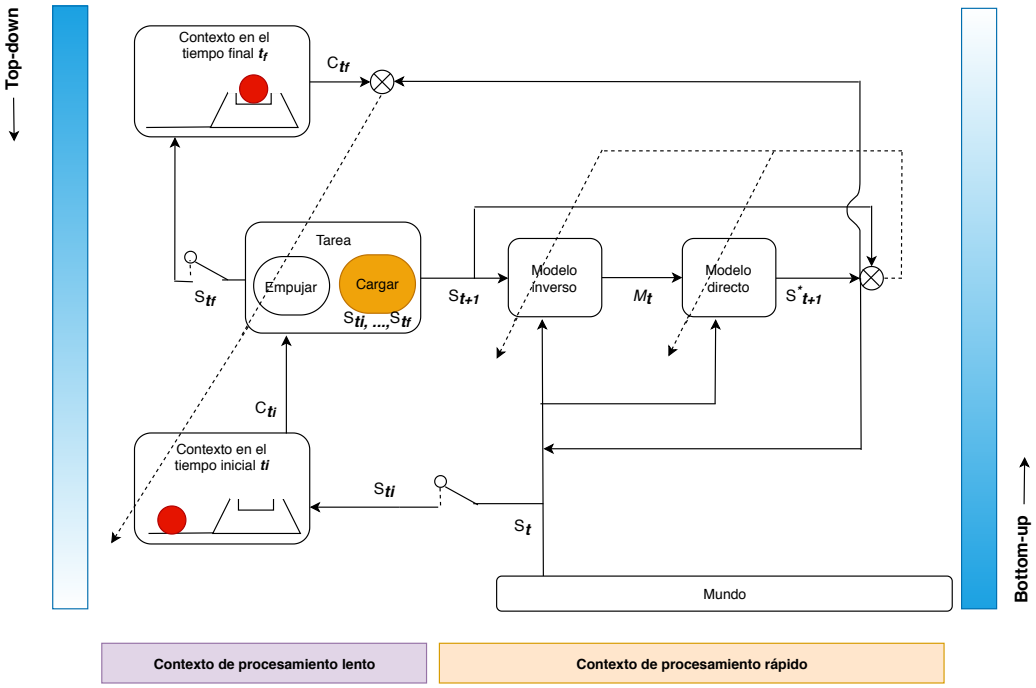

Figura 19: Adaptación del modelo de procesamiento con predicciones a largo plazo de contexto y tarea con las condición incongruente: objeto 1 en contexto 2 con la tarea *Cargar*.

7. Herramientas computacionales

En esta investigación se tomará como base el aprendizaje de máquina para la implementación del modelo propuesto debido a que se considera una fuente de herramientas poderosas para la representación y el procesamiento de información (Figura 20). El aprendizaje de máquina se centra en el estudio de algoritmos capaces de mejorar su desempeño en una tarea mediante el aprendizaje de la experiencia previa (Mjolsness and DeCoste, 2001). A continuación se describirán algunas herramientas computacionales que se consideran de utilidad para el diseño de la arquitectura que se desea proponer, tomando como base las implementaciones previamente revisadas.

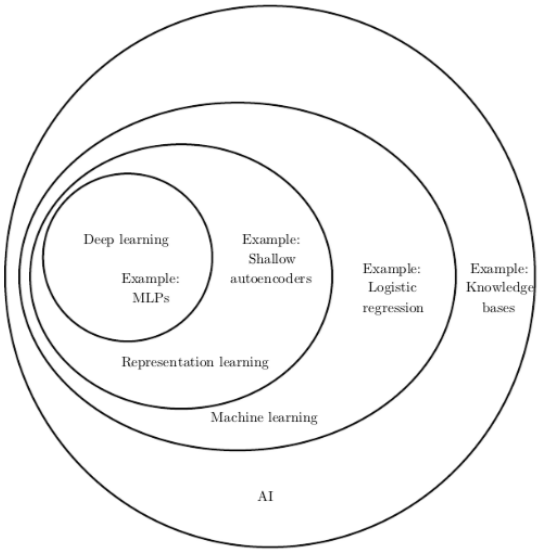

Figura 20: Diagrama de Venn que ilustra que el aprendizaje de máquina forma parte de uno de los enfoques de la Inteligencia artificial que implica herramientas basadas en la representación del conocimiento. El aprendizaje profundo forma parte de este campo (Goodfellow et al., 2016)

7.1. Aprendizaje hebbiano

El postulado de Hebb señala lo siguiente:

çuando el axón de una célula A se encuentra lo suficientemente cerca de la célula B

para excitarla y participa persistentemente en hacerla disparar, un proceso de crecimiento o cambio metabólico en una o ambas células tiene lugar, de tal forma que la eficiencia de A como disparadora de B se incrementa" (Hebb, 1949).

La regla de Hebb se expresa en términos matemáticos como lo indica la siguiente ecuación:

$$\Delta w_{ij} = \alpha u_i u_j$$

donde w_{ij} representa la fuerza de conexión entre el nodo i y el nodo j, α es la tasa de aprendizaje, mientras que u_i y u_j corresponden a la activación de los nodos i y j, respectivamente. Existen modificaciones de ésta expresión que consideran también una tasa de olvido.

7.2. Red neuronal autoencoder

Un *autoencoder* es una red neuronal que está entrenada para intentar copiar su entrada en su salida (Goodfellow et al., 2016). Este tipo de red se compone de tres partes (Figura 21):

- Capa oculta h que describe un código utilizado para representar la entrada
- Función de codificador $h = f(x)$
- Decodificador que produce una reconstrucción $r = g(h)$

Si la red tiene éxito aprendiendo simplemente a configurar $g(f(x)) = x$, entonces no es especialmente útil. En lugar de ello, los *autoencoders* están restringidos de tal manera que se les permite copiar sólo de forma aproximada los datos de entrada que se asemejan a los datos de entrenamiento. Dada una entrada x, la tarea de la red consiste en asignar x a un punto dimensional bajo y, tal que x pueda ser recuperado de y Romero et al. (2017). Por lo cual, la estructura resultante h se puede elegir para representar los datos de entrada en una menor dimensionalidad.

7.3. Red neuronal convolucional

Una red neuronal convolucional (CNN, por sus siglas en inglés) es un tipo de red neuronal especializada en el procesamiento de datos que presentan una topología similar a una

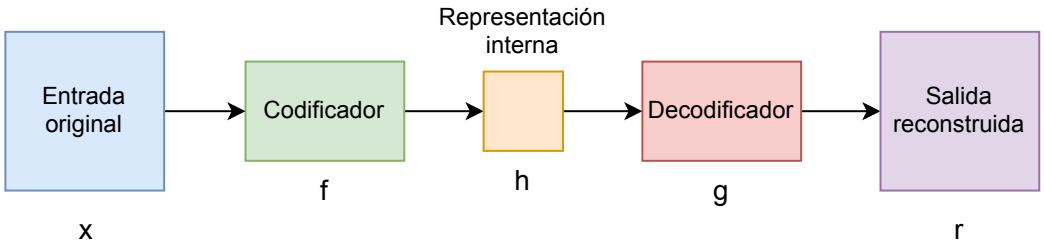

Figura 21: Arquitectura de un autoencoder. Un autoencoder consta de dos componentes: el codificador f (mapeo de x a h) y el decodificador g (mapeo de h a r), donde h es la representación interna o código.

cuadrícula. Las CNNs son simplemente redes neuronales que utilizan la convolución en lugar de la multiplicación general de matrices en al menos una de sus capas. En su forma más general, la convolución es una operación entre dos funciones de un argumento de valor real que produce una tercera función (Goodfellow et al., 2016; LeCun et al., 1989; Zhi et al., 2016). La operación de convolución es típicamente denotada con un asterisco:

$$s(t) = (x * w)(t)$$

donde x a menudo se conoce como la entrada y w como el *kernel* o filtro, mientras que la salida s se conoce como el mapa de características. Un procesamiento típico en cada capa de una CNN implica tres etapas Goodfellow et al. (2016). Durante la primera etapa, se realizan varias operaciones convolucionales entre la entrada correspondiente y el *kernel* para generar el mapa de características. En la segunda etapa, se aplica una función de activación no lineal, como la función de activación lineal rectificada (ReLU) con la finalidad de eliminar los valores negativos de dicho mapa (Figura 23). Finalmente, en la tercera etapa se aplica una función de agrupamiento o *pooling* para modificar aún más la salida de la capa. Un ejemplo de este operador es el *max pooling*, el cual toma el valor máximo del mapa de características (Zhou and Chellappa, 1988).

Después de aplicar estas etapas en cada capa que conforma a la CNN, las salidas obtenidas se utilizan como entrada de una red neuronal *fully-connected* para clasificar la entrada original de la CNN.

7.4. Red neuronal recurrente: LSTM

Una red neuronal recurrente (RNN) forma parte de una familia de redes neuronales que presentan la propiedad de procesar secuencias de datos $x_1, ..., x_\tau$, permitiendo que esta

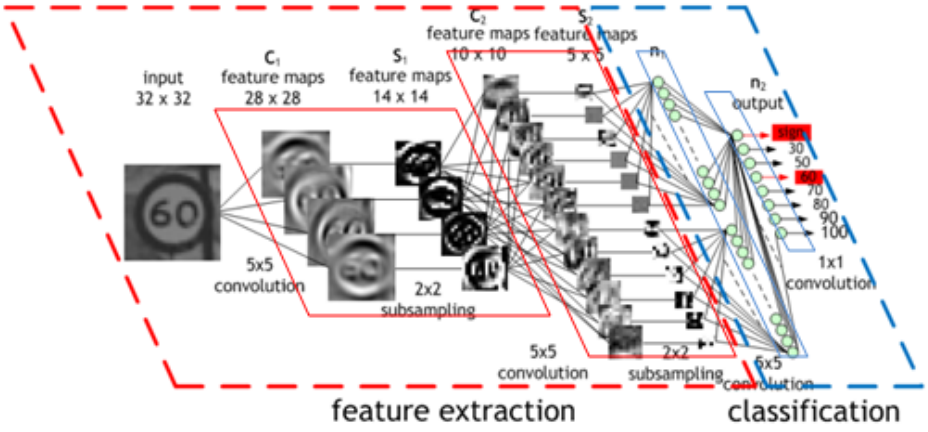

Figura 22: Ilustración del procesamiento en una red neuronal convolucional (Peemen et al., 2016).

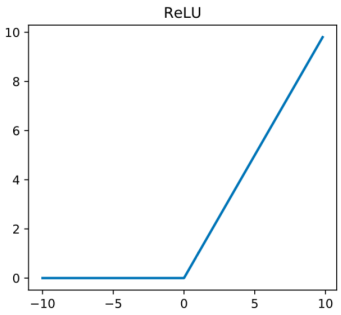

Figura 23: Función de activación ReLU

información persista por largos periodos de tiempo Hochreiter and Schmidhuber (1997). Los modelos que resultan más efectivos en aplicaciones prácticas se denominan RNN con compuertas.

Dentro de las RNNs, se encuentran aquellas conformadas por unidades denominadas celdas de memoria de corto-largo plazo (LSTM, por sus siglas en inglés). Estas celdas de memoria pueden mantener su estado a lo largo del tiempo mediante funciones de activación no lineales que regulan el flujo de información dentro y fuera de la celda. Greff et al. (2017). En su interior, la celda está conformada por las siguientes compuertas (Figura 24):

- Compuerta de olvido: Decide qué información será eliminada de la celda de memoria.
- Compuerta de de entrada: Decide qué valores de la entrada actualizarán el estado de la celda de memoria.
- Compuerta de salida: Decide la salida en función de la entrada y el estado de la celda de memoria actual.

Las ecuaciones para cada una de las compuertas de una celda LSTM son:

$$
\begin{aligned}
i_t &= \sigma\big(w_i[h_{t-1}, x_t] + b_i\big) \\
f_t &= \sigma\big(w_f[h_{t-1}, x_t] + b_f\big) \\
o_t &= \sigma\big(w_o[h_{t-1}, x_t] + b_o\big)
\end{aligned}
$$

donde, i, f, o son las compuertas de entrada, olvido y salida, respectivamente, σ corresponde a una función sigmoide que indicará si una característia se mantendrá o no, w_x corresponde a los pesos de conexión establecido en cada compuerta, h_{t-1} es la salida de la celda LSTM previa, x_t la entrada en el momento actual y b_x el bias de la respectiva compuerta. Por lo tanto, una celda LSTM calcula un estado oculto h_t mediante las siguientes ecuaciones:

$$
\begin{aligned}
\tilde{C}_t &= \tanh\big(w_c[h_{t-1}, x_t] + b_c\big) \\
C_t &= f_t(c_{t-1}) + i_t(\tilde{C}_t) \\
h_t &= \tanh(C_t)(o_t)
\end{aligned}
$$

donde C_t es el estado de la celda (memoria) en el tiempo t y $\tilde{C}_t$ representa el candidato para el estado de la celda (memoria) en el tiempo t.

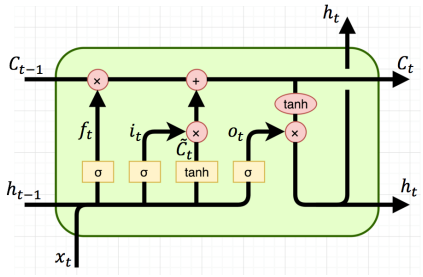

Figura 24: Representación de una celda LSTM. Dentro de la celda se muestran las compuertas de entrada, olvido y salida.

7.5. ConvLSTM

Una convLSTM Xingjian et al. (2015) es una variante de LSTM que realiza operaciones convolucionales dentro de la celda. De igual manera que la LSTM clásica, la convLSTM es un tipo especial de RNN capaz de aprender dependencias a largo plazo. A diferencia de la LSTM, la convLSTM reemplaza la multiplicación de matrices por la operación de convolución en cada compuerta de la celda (Figura 25). Al hacerlo, captura las características espaciales subyacentes mediante operaciones de convolución en datos multidimensionales.

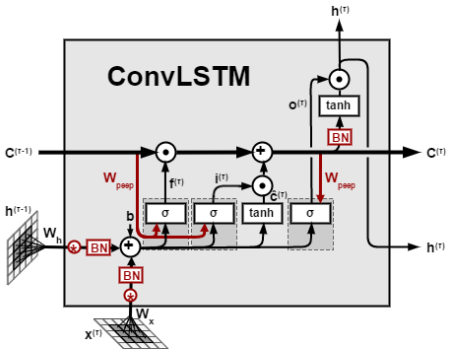

Figura 25: Representación de una celda ConvLSTM. A diferencia de las LSTM tradicionales, una celda convLSTM utiliza la operación de convolución en lugar de la multiplicación de matrices.

7.6. Mapa autoorganizado

El mapa autoorganizado (SOM, por sus siglas en inglés *Self-Organizing Map*) está constituído por una arquitectura y un algoritmo desarrollados por Teuvo Kohonen durante los años 80 para implementarse como una red neuronal artificial (RNA). A diferencia de otras

RNAs, el SOM tiene la propiedad particular de poder proporcionar una representación "interna" organizada espacialmente de la información de entrada. El SOM tiene una estructura tipo lámina u hoja, donde los nodos o neuronas que conforman la red tienen asociadas ciertas coordenadas. Una vez que el mapa ha sido entrenado, únicamente una neurona o conjunto de neuronas responderá de manera activa al patrón de entrada proporcionado. De esta forma, la ubicación espacial o las coordenadas de las neuronas dentro de la red corresponderán a diferentes dominios de patrones de entrada (Kohonen, 1990).

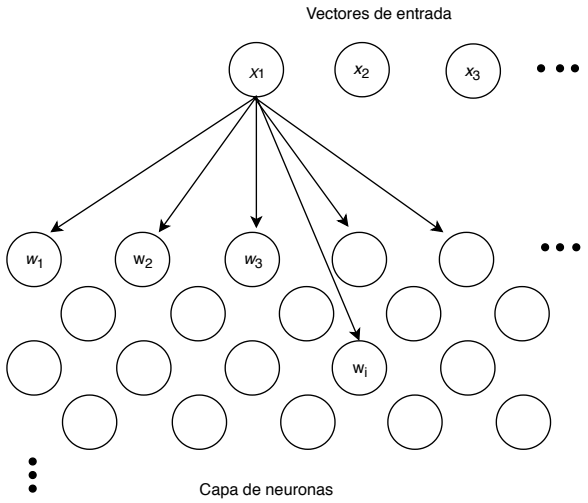

Figura 26: Diagrama de mapa autoorganizado:SOM. Figura adaptada de Kohonen (1990).

El mecanismo de aprendizaje del SOM es denominado no supervisado y por competencia. Las neuronas de la red compiten entre sí y se desarrollan adaptativamente como detectoras de distintos patrones de entrada durante el entrenamiento. Por otro lado, el aprendizaje es no supervisado ya que no se tiene el conocimiento previo de la salida correcta de la red. El aprendizaje en este tipo de redes surge sin la necesidad de etiquetas externas que proporcionen una salida deseada. Esto se logra mediante la construcción de un modelo de densidad del conjunto de los datos de entrada a partir de un proceso denominado cuantización vectorial. Dicho método genera una aproximación a una distribución de probabilidad contínua de un vector variable dado un conjunto finito de entradas.

En un SOM, cada neurona está asociada a un vector de pesos denominado vector de referencia. Supongamos que los pesos de las neuronas de un SOM está representado por

el conjunto de vectores variables $\{w_i(t) : m_i \in R^n, i = 1, 2, ..., k\}$ y, que existe una muestra estadística de vectores observables $x = x(t) \in R^n$, donde t es la coordenada del tiempo y además, los vectores $w_i(0)$ fueron inicializados de alguna forma, por ejemplo, por selección aleatoria. El aprendizaje por competencia implica que $x(t)$ sea comparado simultáneamente con cada vector de pesos $w_i(t)$ en cada instante de tiempo $t = 1, 2, 3...$, entonces la neurona cuyo vector se aproxime mejor a $x(t)$ será denominada neurona ganadora o *best matching unit*. Una vez que la neurona ganadora ha sido identificada, su vector de pesos asociado $w_i(t)$ será modificado para que coincida aún más con el vector de entrada $x(t)$. De manera común, la comparación entre los vectores de referencia y los vectores de entrada de la red se realiza mediante alguna medición de distancia $d(x, m_i)$. Si $i = c$ corresponde al índice de la neurona ganadora, entonces $d(x, w_c)$ será reducida, quedando intactos los vectores de referencia w_i donde $i \neq c$ (Kohonen, 1990).

Por otro lado, la cuantización vectorial es un método clásico para proporcionar una función de densidad de probabilidad contínua $p(x)$ de un conjunto finito de vectores x a partir de un *codebook* o un libro de código de vectores w_i. Una vez que el *codebook* se ha definido, la aproximación de x implica encontrar el vector de referencia w_c más cercano a x. La menor distancia entre ambos vectores minimiza el error:

$$E = \int \|(x - w_c)\|^r \ p(x)dx$$

Usando el criterio del error cuadrático, considerando $r = 2$, se puede demostrar que la optimización de pasos de descenso de gradiente óptima del E en el espacio w_c, produce la secuencia:

$$w_c(t + 1) = w_c(t) + \alpha \left[x(t) - w_c(t)\right]$$

$$w_i(t + 1) = w_i(t), i \neq c$$

Siendo $\alpha(t)$ una secuencia adecuada de coeficientes de ganancia que decrece monotónicamente $0 < \alpha(t) < 1$.

Los dos aspectos fundamentales que producen mapas autoorganizados son: (1) concentración espacial de actividad de la red neuronal; de tal forma que la actividad se concentra en la neurona o set de vecinos que mejor se ajustan a la entrada y (2) mayor sensibilización

o ajuste de la neurona de mejor coincidencia y sus vecinos topológicos a la entrada actual. Por lo tanto, el algoritmo del SOM puede describirse en dos fases:

- Selección de la neurona ganadora
- Adaptación o ajuste de vectores de pesos

Para poder formar los mapas ordenados el aprendizaje de las neuronas que aprenden no se afectan independientemente, sino que se afectan como *subsets* topológicamente relacionados. En modelos de redes neuronales biológicamente inspiradas se puede implementar aprendizaje correlacionado entre neuronas vecinas mediante retroalimentación lateral u otras interacciones laterales. En el SOM, una forma de interacción lateral entre neuronas puede realizarse mediante la formación de estructuras dentro de la red definiendo un *set* de vecindad N_c alrededor de la neurona c. Estableciendo una vecindad, a cada paso de entrenamiento el vector de pesos de la neurona ganadora, junto con los pesos de las neuronas que se encuentren dentro de su vecindad serán ajustados para parecerse más al vector de entrada. En cada paso del entrenamiento, la vecindad siempre estará centrada en la neurona que haya resultado ganadora. El radio de la vecindad puede variar con el tiempo. De hecho, ha sido determinado experimentalmente que para obtener un buen ordenamiento global resulta conveniente fijar N_c amplia al inicio y que se reduzca monotónicamente en el tiempo. Esto ocurre debido a que fijando N_c grande, correspondiente a una resolución espacial gruesa induce un orden global aproximado en los vectores de pesos w_i. Posteriormente, la reducción monotónica de la vecindad mejora la resolución espacial del mapa sin destruir el orden global obtenido.

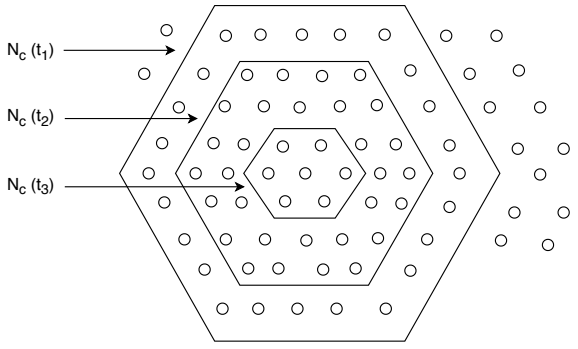

Figura 27: Esquema de la vecindad topológica $N_c(t)$, donde $t_1 < t_2 < t_3$. Figura adaptada de Kohonen (1990).

La activación Aj que corresponde a la neurona j del mapa, estará en función de la distancia entre el vector de entrada x y el vector de pesos asociado a la neurona j. De manera tradicional, la distancia Euclidiana ha sido utilizada para realizar dicho cálculo. No obstante, en el presente proyecto se desea calcular la distancia y la orientación de los vectores. Por lo tanto, la activación estará dada por:

$$A_j = \frac{1}{2}\left(\|x - w_j\| + 1 - \frac{x \cdot w_j}{\|x\|\|w_j\|}\right) \tag{1}$$

donde el primer término corresponde a la distancia euclidiana, mientras que el segundo se refiere a la similitud coseno entre ambos vectores. Una vez que se ha calculado la activación de cada neurona y se ha determinado la neurona ganadora, los vectores de pesos son ajustados de acuerdo a la siguiente ecuación:

$$\Delta w_j = a(t)h_j(x - w_j)$$

donde w es el vector de pesos entre el nodo j y el vector de entrada x, $a(t)$ es la tasa de aprendizaje, que varía en función del tiempo y h_j es la función de vecindad, calculada a partir de la distancia entre cada nodo y la neurona ganadora. La función de vecindad está dada por:

$$h_j = e^{\left(\frac{-B_j}{2\sqrt{n}}\right)}$$

donde B_j es la distancia entre el nodo j y la neurona ganadora y n es el número de nodos del mapa. En el caso de que B_j se encuentre fuera del tamaño de la vecindad v, entonces $h_j = 0$. La vecindad v es monotónicamente decreciente desde v_i hasta $v_f = 1$. La tasa de aprendizaje $a(t)$ está dada por:

$$\alpha(t) = \alpha^i\left(\frac{t(\alpha^f - \alpha^i)}{v_i - v_f}\right)$$

donde α^i y α^f son parámetros que se proporcionan al inicio del entrenamiento. Los SOMs han sido implementados en tareas como reconocimiento de patrones, procesos de control, procesamiento de información semántica. En el área de robótica cognitiva, los SOMs han servido como unidad de arquitecturas computacionales para control visuomotor.

8. Plataforma de experimentación: robot humanoide NAO

La prueba del modelo de contexto y tarea se realizará en la plataforma NAO. El robot humanoide NAO mide 58 cm y es completamente programable e interactivo. Para los fines del proyecto se usará la versión *V5* del robot, con tipo de cuerpo H25 (Fig. 28). Este modelo tiene 25 grados de libertad y es capaz de percibir el entorno mediante sus sensores, los cuales incluyen dos cámaras en su cabeza, cuatro micrófonos, nueve sensores táctiles, dos sensores de ultrasonidos, ocho sensores de presión, un acelerómetro y un giróscopo. Además, como dispositivos de salida presenta 53 LEDs RGB, sintetizador de voz y dos altavoces.

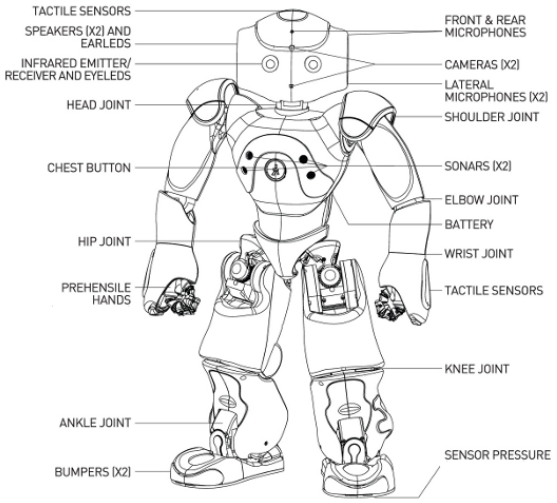

Figura 28: Diseño general del NAO *V5*. Se muestra la ubicación de las articulaciones del agente, así como los sensores que le permiten interactuar con el entorno.

El CPU de la plataforma se ubica en la cabeza del robot. En el caso de la versión 5 del NAO consiste en un procesador tipo ATOM Z530 1.6 GHz con 1GB de RAM, 2GB de Flash Memory y 8 GB Micro SDHC. El sistema operativo de la plataforma NAO se denomina NAOqi OS y es una distribución GNU/Linux basada en Gentoo. El entorno de programación usado para programar al robot es el *NAOqi Framework*. Este entorno permite una comunicación homogénea entre distintos módulos (movimiento, audio, video) y programación homogénea. Además, es *crossplatform* (puede usarse en Windows, Linux y MacOS) y *crosslanguague* (puede programarse en python y C++).

El entorno de programación NAOqi puede ser instalado en cualquier PC mediante el

SDK (por sus siglas en inglés, *Software Developmental Kit*). Esto permite correr NAOqi en una plataforma distinta a la del robot, así como ejecutarse en un simulador como WEBOTS (Fig. 29). Este simulador permite simular entornos y más de cuarenta agentes artificiales distintos. Dentro de este simulador existe un módulo disponible para la licencia del NAO denominado nao_qi, que incluye simulaciones predefinidas con los parámetros específicos del robot. WEBOTS proporciona una plataforma ideal para reproducir comportamientos de manera segura antes de ejecutarlos en el robot real.

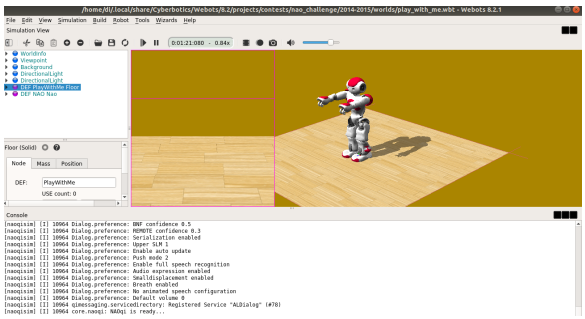

Figura 29: Entorno de simulación de la plataforma WEBOTS. Esta herramienta es capaz de reproducir comportamientos del robot NAO mediante el módulo nao_qi.

En este semestre se realizó la instalación del C++ SDK, así como del simulador WEBOTS.

9. Cronograma de actividades

Finalmente, se presenta el cronograma de actividades para asegurar el término del proyecto en tiempo y forma. La distribución de actividades por semestre se muestra en la Fig. 30.

Referencias

Adams RA, Shipp S and Friston KJ (2013) Predictions not commands: active inference in the motor system. *Brain Structure and Function* 218(3): 611–643.

Asada M, MacDorman KF, Ishiguro H and Kuniyoshi Y (2001) Cognitive developmental robotics as a new paradigm for the design of humanoid robots. *Robotics and Autonomous systems* 37(2-3): 185–193.

Figura 30: Cronograma de actividades del doctorado.

Barsalou LW (2008) Grounded cognition. *Annu. Rev. Psychol.* 59: 617–645.

Bazire M and Brézillon P (2005) Understanding context before using it. In: *International and Interdisciplinary Conference on Modeling and Using Context*. Springer, pp. 29–40.

Blakemore SJ, Wolpert D and Frith C (2000) Why can't you tickle yourself? *Neuroreport* 11(11): R11–R16.

Choi M, Matsumoto T, Jung M and Tani J (2018) Generating goal-directed visuomotor plans based on learning using a predictive coding type deep visuomotor recurrent neural network model. *arXiv preprint arXiv:1803.02578* .

Clark A (2013) The many faces of precision (replies to commentaries on "whatever next? neural prediction, situated agents, and the future of cognitive science"). *Frontiers in psychology* 4: 270.

Clark A (2015) *Embodied prediction*. Open MIND. Frankfurt am Main: MIND Group.

Corbetta D, Thurman SL, Wiener RF, Guan Y and Williams JL (2014) Mapping the feel of the arm with the sight of the object: on the embodied origins of infant reaching. *Frontiers in psychology* 5: 576.

Escobar-Juárez E, Schillaci G, Hermosillo-Valadez J and Lara-Guzmán B (2016) a self-organized internal models architecture for coding sensory–motor schemes. *Frontiers in Robotics and AI* 3: 22.

Gaona W, Escobar E, Hermosillo J and Lara B (2015) Anticipation by multi-modal association through an artificial mental imagery process. *Connection Science* 27(1): 68–88.

Goodfellow I, Bengio Y, Courville A and Bengio Y (2016) *Deep learning*, volume 1. MIT press Cambridge.

Greff K, Srivastava RK, Koutník J, Steunebrink BR and Schmidhuber J (2017) Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems* 28(10): 2222–2232.

Hebb DO (1949) The organization of behavior. a neuropsychological theory .

Hochreiter S and Schmidhuber J (1997) Long short-term memory. *Neural computation* 9(8): 1735–1780.

Jeffery KJ, Anderson MI, Hayman R and Chakraborty S (2004) A proposed architecture for the neural representation of spatial context. *Neuroscience & Biobehavioral Reviews* 28(2): 201–218.

Jordan MI and Rumelhart DE (1992) Forward models: Supervised learning with a distal teacher. *Cognitive science* 16(3): 307–354.

Kawato M (1999) Internal models for motor control and trajectory planning. *Current opinion in neurobiology* 9(6): 718–727.

Kohonen T (1990) The self-organizing map. *Proceedings of the IEEE* 78(9): 1464–1480.

LeCun Y et al. (1989) Generalization and network design strategies. *Connectionism in perspective* : 143–155.

Lee S and Kim I (2018) A robotic context query-processing framework based on spatio-temporal context ontology. *Sensors* 18(10): 3336.

Luo D, Hu F, Deng Y, Liu W and Wu X (2016) An infant-inspired model for robot developing its reaching ability. In: *Development and Learning and Epigenetic Robotics (ICDL-EpiRob), 2016 Joint IEEE International Conference on*. IEEE, pp. 310–317.

Mar T, Tikhanoff V and Natale L (2018) What can i do with this tool? self-supervised learning of tool affordances from their 3-d geometry. *IEEE Transactions on Cognitive and Developmental Systems* 10(3): 595–610.

Marr D (1982) Vision: A computational approach.

Mjolsness E and DeCoste D (2001) Machine learning for science: state of the art and future prospects. *science* 293(5537): 2051–2055.

Möller R and Schenck W (2008) Bootstrapping cognition from behavior—a computerized thought experiment. *Cognitive Science* 32(3): 504–542.

Muller RU and Kubie JL (1987) The effects of changes in the environment on the spatial firing of hippocampal complex-spike cells. *Journal of Neuroscience* 7(7): 1951–1968.

Nadel L and Willner J (1980) Context and conditioning: A place for space. *Physiological Psychology* 8(2): 218–228.

Nguyen-Tuong D and Peters J (2011) Model learning for robot control: a survey. *Cognitive processing* 12(4): 319–340.

Noda K, Arie H, Suga Y and Ogata T (2013) Multimodal integration learning of object manipulation behaviors using deep neural networks. In: *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, pp. 1728–1733.

Norman G and Eacott M (2005) Dissociable effects of lesions to the perirhinal cortex and the postrhinal cortex on memory for context and objects in rats. *Behavioral neuroscience* 119(2): 557.

O'Keefe J and Dostrovsky J (1971) The hippocampus as a spatial map: preliminary evidence from unit activity in the freely-moving rat. *Brain research* .

Peemen M, Shi R, Lal S, Juurlink B, Mesman B and Corporaal H (2016) The neuro vector engine: Flexibility to improve convolutional net efficiency for wearable vision. In: *Proceedings of the 2016 Conference on Design, Automation & Test in Europe*. EDA Consortium, pp. 1604–1609.

Pfeifer R and Scheier C (2001) *Understanding intelligence*. MIT press.

Rao RP and Ballard DH (1999) Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience* 2(1): 79.

Riemann BL and Lephart SM (2002) The sensorimotor system, part ii: the role of proprioception in motor control and functional joint stability. *Journal of athletic training* 37(1): 80.

Romero J, Olson JP and Aspuru-Guzik A (2017) Quantum autoencoders for efficient compression of quantum data. *Quantum Science and Technology* 2(4): 045001.

Schillaci G, Lara B and Hafner VV (2012) Internal simulations for behaviour selection and recognition. In: *International Workshop on Human Behavior Understanding*. Springer, pp. 148–160.

Sherrington CS (1907) On the proprio-ceptive system, especially in its reflex aspect. *Brain* 29(4): 467–482.

Spratling MW (2017) A review of predictive coding algorithms. *Brain and cognition* 112: 92–97.

Turner RM (1998) Context-mediated behavior for intelligent agents. *International Journal of Human-Computers Studies* 48(3): 307–330.

Wijesinghe LP, Triesch J and Shi BE (2018) Robot end effector tracking using predictive multisensory integration. *Frontiers in neurorobotics* 12.

Wilson M (2002) Six views of embodied cognition. *Psychonomic bulletin & review* 9(4): 625–636.

Wolpert DM, Ghahramani Z and Jordan MI (1995) An internal model for sensorimotor integration. *Science* 269(5232): 1880–1882.

Wolpert DM and Kawato M (1998) Multiple paired forward and inverse models for motor control. *Neural networks* 11(7-8): 1317–1329.

Xingjian S, Chen Z, Wang H, Yeung DY, Wong WK and Woo Wc (2015) Convolutional lstm network: A machine learning approach for precipitation nowcasting. In: *Advances in neural information processing systems*. pp. 802–810.

Ye P, Wang T and Wang FY (2018) A survey of cognitive architectures in the past 20 years. *IEEE transactions on cybernetics* (99): 1–11.

Zhang T, Hu F, Deng Y, Nie M, Liu T, Wu X and Luo D (2018) Self-developing proprioception-based robot internal models. In: *International Conference on Intelligence Science*. Springer, pp. 321–332.

Zhi X, Wei D and Zhang W (2016) A generalized convolution theorem for the special affine fourier transform and its application to filtering. *Optik-International Journal for Light and Electron Optics* 127(5): 2613–2616.

Zhong J, Cangelosi A, Ogata T and Zhang X (2018) Encoding longer-term contextual information with predictive coding and ego-motion. *Complexity* 2018.

Zhou YT and Chellappa R (1988) Computation of optical flow using a neural network. In: *IEEE International Conference on Neural Networks*, volume 1998. pp. 71–78.