UNIVERSIDAD DE GUANAJUATO

DIVISIÓN DE CIENCIAS E INGENIERÍAS
CAMPUS LEÓN

"ESTUDIO DE APRENDIZAJE POR REFUERZO PARA PROCESAMIENTO DE IMÁGENES MÉDICAS"

TESIS

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN INGENIERÍA FÍSICA

PRESENTA:

ÁNGEL ISAAC GÓMEZ CANALES

ASESORES:

DIRECCIÓN: DR. LUIS CARLOS PADIERNA GARCÍA
CODIRECCIÓN: DR. ARTURO GONZÁLEZ VEGA

LEÓN, GUANAJUATO 2022

Índice

Índice de Figuras

Introducción

La salud y el bienestar de las personas siempre ha sido una de las áreas de estudio más activas. Debido a esto, en la actualidad el área de la medicina se ha vuelto multidisciplinaria, donde participantes de distintas disciplinas trabajan en conjunto para entregar soluciones a las necesidades de los pacientes. Un ejemplo donde se puede apreciar el beneficio del trabajo multidisciplinario es en las tecnologías e instrumentos usados por los médicos para monitoreo o diagnóstico de los pacientes, los cuales son desarrollados y fabricados por profesionales cuya especialización no es la medicina (Beyer, y otros, 2021). En particular, la física juega un papel importante en el diagnóstico y tratamiento de enfermedades. El término física médica se refiere al trabajo de físicos empleados en hospitales, quienes se enfocan principalmente en aplicaciones médicas de radiación, imagenología para diagnóstico y mediciones clínicas (Keevil, 2012).

En la actualidad existe una discusión continua en la física médica sobre la inteligencia artificial (IA), pues se espera que esta pueda tener un impacto significativo en los procesos del cuidado de la salud. Las disciplinas de imagenología y radioterapia se consideran como las primeras en implementar ampliamente el uso de IA (Andersson, y otros, 2021). La IA puede servir a los médicos como una segunda opinión para ayudar en la toma de decisiones, disminuyendo así la tasa de errores en los diagnósticos, un ejemplo de estos es la clasificación de tumores en benignos o malignos.

Dentro del amplio campo de la IA, existe una aplicación o subcampo conocido como Machine Learning (ML), este subcampo permite a las máquinas poder aprender a partir de datos preexistentes sin tener que ser programadas explícitamente para esto. A su vez, ML cuenta con tres líneas principales de aprendizaje, la primera es el aprendizaje supervisado, donde se le deben pasar al algoritmo de aprendizaje tanto los datos como las salidas esperadas para cada dato. Por otra parte, en el área de aprendizaje no supervisado solo es necesario pasar al algoritmo de aprendizaje los datos para entrenamiento. Finalmente, la tercera área, conocida como aprendizaje por refuerzo aprende de los datos en un estilo de prueba y error. Adicionalmente, se puede considerar una cuarta línea que es transversal, pues resuelve tareas de las tres áreas antes mencionadas, esta se conoce como Deep Learning.

Aunque usualmente se suele asociar el aprendizaje por refuerzo con agentes capaces de jugar videojuegos, esta no es la única aplicación de este tipo de aprendizaje en Machine Learning. El aprendizaje por refuerzo también puede ayudar en el área médica particularmente en imagenología. Un ejemplo de esto es el modelo de aprendizaje por refuerzo estudiado en este trabajo, y propuesto por (Furuta, Inoue, & Yamasaki, 2019). Este modelo ayuda a eliminar el ruido en imágenes médicas (el ruido en una imagen es una alteración en los pixeles de esta que no se origina por el contenido de la escena original en la imagen), considerando cada píxel en la imagen como un agente que puede realizar un conjunto de acciones con el propósito de eliminar este ruido. El presente estudio se centra en el área de aprendizaje por refuerzo y aprendizaje profundo para verificar sus ventajas y limitantes como supresor de ruido en imágenes.

Con base en lo expuesto anteriormente, se plantea lo siguiente:

Hipótesis

Arquitecturas de redes neuronales profundas que emplean aprendizaje por refuerzo ayudarán a eliminar ruido de imágenes médicas.

Para la verificación de esta hipótesis se plantean los siguientes objetivos.

Objetivo General

Analizar el funcionamiento, ventajas y alcance del aprendizaje por refuerzo como estrategia de entrenamiento de redes neuronales profundas para eliminación de ruido en imágenes médicas dentales.

Objetivos Específicos

1. Comprender los fundamentos y componentes de arquitecturas de redes neuronales clásicas.
2. Analizar arquitecturas de redes neuronales profundas, comúnmente empleadas para procesamiento de imágenes (convolucionales, generativas y recurrentes).
3. Formar un marco teórico conciso sobre los conceptos fundamentales del aprendizaje por refuerzo.
4. Implementar redes neuronales profundas que resuelvan problemas benchmark de procesamiento de imágenes y aprendizaje por refuerzo.
5. Recopilar y estudiar un conjunto de investigaciones actuales de aprendizaje por refuerzo para procesamiento de imágenes.
6. Replicar la experimentación de un artículo científico donde utilicen aprendizaje por refuerzo para eliminación de ruido en imágenes.
7. Plantear metodologías con diferentes aprendizajes para corregir ruido.
8. Probar las arquitecturas replicadas para eliminar ruido de imágenes médicas.

Marco Teórico

Machine Learning

Machine Learning es un área de estudio dentro del campo de la inteligencia artificial, la cual busca crear algoritmos capaces de aprender a partir de datos que se le pasan a estos algoritmos (Goodfellow, Bengio, & Courville, 2016). El término "aprender" puede resultar confuso en este contexto, por esto, se define a continuación de acuerdo con (Mitchell, 1997):

"" Se dice que un programa de computadora aprende de la experiencia *E* con respecto a una tarea *T* y métrica de desempeño *P*, si su desempeño en la tarea *T*, medido por la métrica *P*, mejora con la experiencia E. ""

En complemento a esto, Machine Learning también puede ser definido como un conjunto de métodos capaces de detectar patrones en datos de manera automática, para luego usar estos patrones para predecir datos futuros, o llevar a cabo otro tipo de toma de decisiones (Murphy, 2012).

Debido a la gran cantidad de algoritmos de Machine Learning que existen, resulta de gran utilidad clasificar estos algoritmos con base en algún criterio, como puede ser si estos algoritmos son entrenados o no bajo la supervisión de un humano. Con base en este criterio, se pueden clasificar los algoritmos de Machine Learning en tres categorías: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje por refuerzo (Géron, 2020). En la **Figura 1** se muestra una taxonomía de Machine Learning, de igual manera se muestran las tres categorías en las que se clasifican los algoritmos de Machine Learning, adicional a estas tres categorías, se observa una categoría transversal con el nombre de Deep Learning.

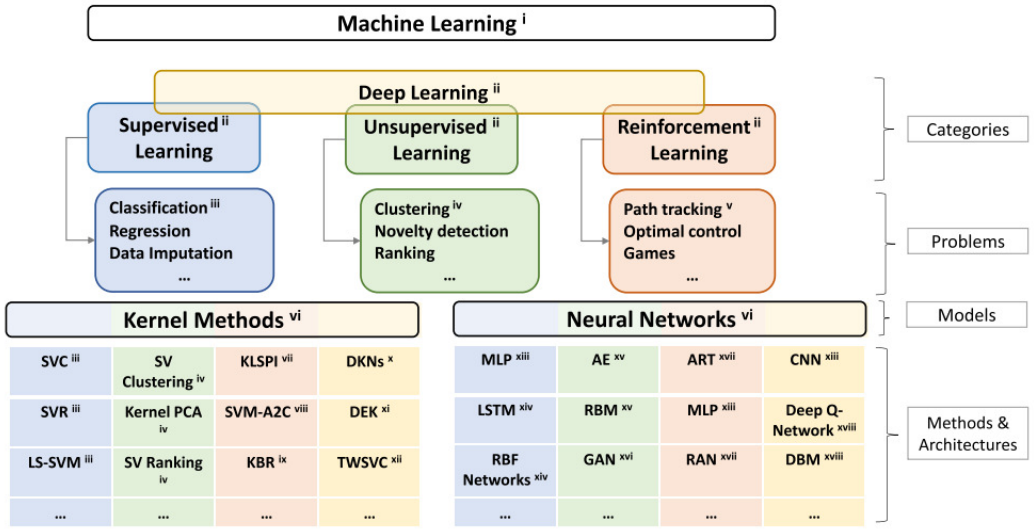

Figura 1. Taxonomía de Machine Learning (Bedolla, Padierna, & Castañeda, 2021).

Deep Learning es un tipo especial de Machine Learning, pues está diseñado para tratar con datos de altas dimensiones. Deep Learning se considera como una categoría transversal ya que puede ser usado para resolver las limitaciones de los algoritmos del aprendizaje supervisado, no supervisado y aprendizaje por refuerzo (Bedolla, Padierna, & Castañeda, 2021).

Ahora se explican un poco más detalladamente las categorías de aprendizaje supervisado y no supervisado. Posteriormente, se explicará de manera más detallada la categoría de aprendizaje por refuerzo, ya que es el enfoque de Machine Learning en el que se centra este trabajo. La categoría transversal de Deep Learning se explicará más adelante.

Aprendizaje supervisado

Una gran cantidad de la actividad de investigación en Machine Learning se centra en el área del aprendizaje supervisado. La característica que define al aprendizaje supervisado es la disponibilidad de datos de entrenamiento etiquetados, es decir cada dato de entrenamiento incluye su solución deseada. Como el nombre sugiere, para este aprendizaje debe haber un "supervisor" que instruya al sistema con las etiquetas para asociar a los datos de entrenamiento (Cunningham, Cord, & Delany, 2008). Después de entrenar estos modelos con los datos de entrenamiento etiquetados, pueden ser usados para predecir la etiqueta de datos que no han sido visto previamente por el modelo.

De manera un poco más formal, el aprendizaje supervisado implica aprender un mapeo entre un conjunto de variables de entrada X y una variable de salida y, y luego aplicar este mapeo para predecir la salida de datos no vistos previamente por el modelo (Cunningham, Cord, & Delany, 2008).

Una de las tareas más comunes dentro del aprendizaje por refuerzo es la de clasificación. En esta tarea, se le pide al programa predecir a cuál de las k categorías disponibles pertenece una entrada. Para resolver esta tarea, usualmente al algoritmo se le pide producir una función $f: \mathbb{R}^n \rightarrow \{1, \dots, k\}$, donde k es el número de clases. Cuando $y = f(\boldsymbol{x})$, el modelo asigna una entrada descrita por el vector $\boldsymbol{x}$, a una categoría y identificada por código numérico. Existen también otras variantes de clasificación donde, por ejemplo, la salida de f es una distribución de probabilidad sobre las clases disponibles (Goodfellow, Bengio, & Courville, 2016). Un ejemplo de clasificación puede ser un filtro de spam, pues debe clasificar correos entrantes en dos clases, spam o no spam.

Otra de las tareas comunes en aprendizaje supervisado es la de regresión. En esta tarea se le pide al programa predecir un valor numérico, dada alguna entrada. Para resolver esta tarea, se le pide al algoritmo que produzca una función $f: \mathbb{R}^n \rightarrow \mathbb{R}$. Esta tarea es similar a la clasificación, pero el formato de salida es distinto en este caso (Goodfellow, Bengio, & Courville, 2016). Un ejemplo de regresión es la predicción del precio de una casa usando características de esta como el número de habitaciones, tamaño del terreno, etc.

Aunque las dos tareas anteriores son las más comunes en aprendizaje supervisado, existen otras menos comunes, pero que igualmente son de gran utilidad. Un ejemplo es la segmentación de imágenes, donde dada una imagen, se le pide al programa dibujar una máscara de un objeto específico al nivel de pixeles. Finalmente, otra tarea es la de detección de objetos, donde dada una imagen, se le pide al programa dibujar una caja que encierre un cierto objeto que se desea detectar dentro de la imagen (Chollet, 2018).

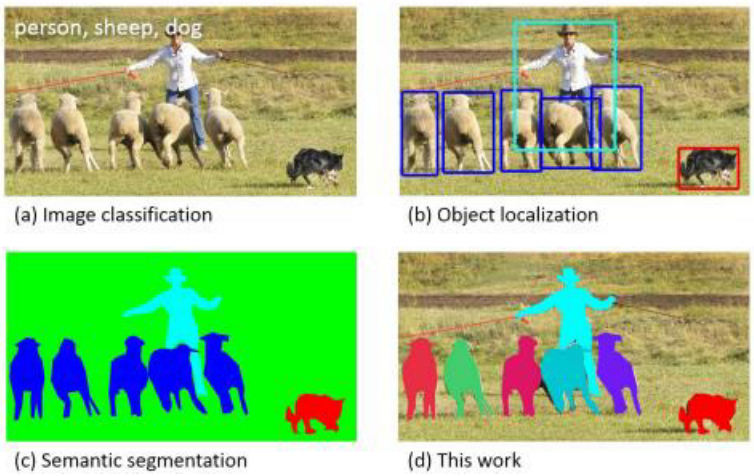

Figura 2. Tareas de aprendizaje supervisado (Lin, y otros, 2015)

En la **Figura 2** se muestran algunas de las tareas de aprendizaje supervisado aplicadas a una imagen. En el inciso a, se muestra la clasificación de los objetos en la imagen. En el inciso b se muestra una detección de objetos en la imagen. En el inciso c se muestra una segmentación, donde a cada píxel de la imagen se le asigna una máscara cuyo color depende de la clase a la que pertenece el píxel, siendo en este caso las clases: persona, oveja, perro y fondo. El tipo de segmentación presente en el inciso c se conoce como segmentación semántica. Finalmente, en el inciso d se muestra una segmentación por instancias, donde se asigna una máscara a cada objeto de interés en la imagen, el color de la máscara cambia, aunque una instancia pertenezca a la misma clase.

Ahora se discute el área de aprendizaje no supervisado.

Aprendizaje no supervisado

Esta área de Machine Learning consiste en encontrar transformaciones interesantes de los datos de entrada sin la ayuda de etiquetas, principalmente para el propósito de visualización de datos (Chollet, 2018). Contrario al aprendizaje supervisado, en esta área de Machine Learning no se conocen las salidas deseadas para cada entrada, es decir, el sistema trata de aprender sin un “supervisor”.

Una de las tareas principales del aprendizaje no supervisado es la de clustering. Así como otras tareas en esta área, el clustering trata de encontrar una estructura en una colección de

datos no etiquetados. El clustering es el proceso de agrupar objetos similares en varios grupos, dicho de otra forma, dividir el datasets en subsets, de tal forma que los datos en un mismo subsets sean similares de acuerdo con una métrica de distancia, pueden usarse distintas métricas de distancia para medir la similitud entre datos, como pueden ser la distancia Euclidiana o la distancia de Manhattan. Estos subsets se conocen como clústeres, un clúster es una colección de objetos que son "similares" entre ellos y que son "distintos" a objetos pertenecientes a clústeres distintos (Madhulatha, 2012). En la **Figura 3** se muestra un ejemplo de clustering mediante el algoritmo de K-means, uno de los algoritmos más populares para clustering.

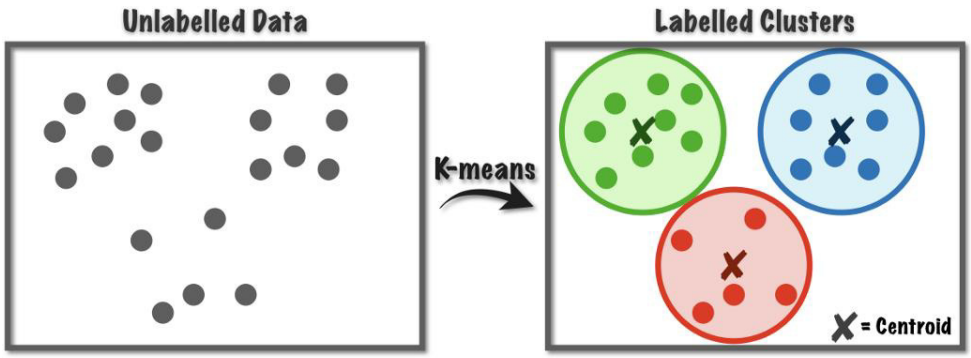

Figura 3. Clustering mediante K-means (Jeffares, 2019)

Otra de las tareas importantes del aprendizaje no supervisado es la de reducción de dimensionalidad. La reducción de dimensionalidad es la transformación de datos con una dimensión alta, en una representación significativa de menores dimensiones (Van Der Maaten, Postma, & Van Den Herik, 2009). Dicho de manera sencilla, el objetivo de la reducción de dimensionalidad es simplificar los datos sin perder mucha información. Una manera de realizar esto sería combinar varias variables que estén correlacionadas en una sola variable (Géron, 2020). Esta tarea resulta muy útil para visualización de datos, pues permite graficar datos en altas dimensiones, las cuales resultaría imposible graficar para visualizar los datos.

Por último, otra de las tareas importantes del aprendizaje no supervisado es la detección de anomalías. La detección de anomalías se refiere al problema de encontrar patrones en los datos que no se ajustan al comportamiento esperado. Estos patrones que no se ajustan al comportamiento esperado se conocen como anomalías, outliers, observaciones discordantes, excepciones, aberraciones, sorpresas, peculiaridades o contaminantes (Chandola, Banerjee, & Kumar, 2009). Una de las aplicaciones principales de la detección de anomalías es la prevención de fraudes de tarjeta de crédito. En la **Figura 4** se muestra un ejemplo de detección de anomalías.

Figura 4. Detección de anomalías (Géron, 2020).

Por último, se discute el área de aprendizaje por refuerzo, el aprendizaje por refuerzo se discute con más detalle puesto que el presente trabajo se enfoca en la aplicación de estas técnicas e imágenes médicas.

Aprendizaje por refuerzo

El aprendizaje por refuerzo es una de las áreas de Machine Learning. A diferencia del aprendizaje supervisado y no supervisado, el aprendizaje por refuerzo es el problema al que se enfrenta un agente el cual debe aprender cierto comportamiento interactuando con un entorno mediante prueba y error (Kaelbling, Littman, & Moore, 1996) como se muestra en la **Figura 5**.

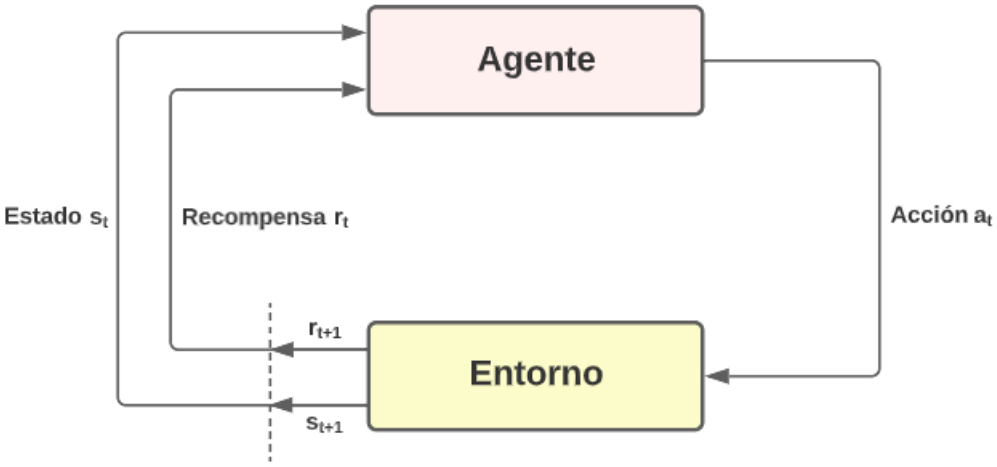

Figura 5. Idea básica del aprendizaje por refuerzo

En este contexto, el agente es el sistema de aprendizaje, este puede observar e interactuar con el entorno mediante la selección y realización de acciones, por las cuales recibirá una recompensa (estas pueden ser negativas o positivas dependiendo de la acción), las recompensas suelen ser denotadas por *r*. El objetivo del agente es aprender la mejor estrategia, llamada política, para obtener la mayor recompensa acumulada a lo largo del tiempo. La política define qué acción debe tomar el agente cuando este está en una situación dada (Géron, 2020). Estas situaciones en las que un agente se puede encontrar son conocidas como estados y se denotan por *s*, los estados son las posiciones o momentos dentro del entorno en las que el agente puede estar. El agente interactúa con el entorno para pasar de un estado a otro, esto llevando a cabo una acción, las acciones suelen ser denotadas por *a.*

De la **Figura 5** se observa que al tiempo t, el agente se encuentra en el estado s_t, posteriormente el agente interactúa con el entorno realizando la acción a_t, después de la interacción el agente pasa al estado s_{t+1} y recibe la recompensa r_{t+1} por esta interacción.

La mayoría de los problemas de aprendizaje por refuerzo pueden ser modelados como procesos de decisión de Markov.

Proceso de Decisión de Markov

El proceso de decisión de Markov es un modelo matemático estocástico para la toma de decisiones. De manera similar a lo discutido para el aprendizaje por refuerzo, el agente interactúa con el entorno tomando decisiones, las cuales cambian el estado del entorno, devolviendo al agente una recompensa. Los procesos de decisión de Markov constan de cinco elementos (S, A, P, R, γ), donde:

- S es el conjunto de estados
- A es el conjunto de acciones, llamado espacio de acción.
- $P(s'|s,a)$ es la probabilidad de pasar del estado actual s al siguiente estado s' realizando la acción a
- $R(s,a,\ s')$ es la función de recompensa, la cual asigna la recompensa que el agente obtiene al pasar del estado s al estado s' realizando la acción a.
- γ es el factor de descuento, representa la importancia que se le da a las recompensas inmediatas y a las recompensas futuras. El factor de descuento toma valores entre 0 y 1. Un valor cercano a 0 indica que se les da mayor importancia a recompensas inmediatas, mientras que un valor cercano a 1 indica que se les da mayor importancia a recompensas futuras.

(Ravichandiran, 2020)

Conceptos básicos de aprendizaje por refuerzo

Episodio

Al iniciar la interacción con el entorno, el agente se encontrará en el estado inicial e interactuará con el entrono realizando acciones hasta llegar a un estado final. A este proceso de interacción entre el entorno y el agente desde el estado inicial hasta el estado final se le conoce como episodio o trayectoria, esto se denota por τ.

Retorno

Previamente se mencionó que el objetivo del agente es maximizar la recompensa acumulada que recibe a lo largo del tiempo. Formalmente, esto se define de la siguiente manera, el agente busca maximizar el *retorno esperado*, donde el retorno, denotado por R_t o G_t, se define como la suma de las recompensas obtenidas después del tiempo t, esto es:

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_T$$

Donde r_{t+1} es la recompensa inmediata que recibe el agente al tiempo *t+1* y T es el paso final del episodio.

Esta ecuación es usada para tareas donde un episodio termina al llegar a un estado particular, llamado *estado terminal*, este tipo de tareas se conoce como *tareas episódicas*.

Existe otro tipo de tareas en las que la interacción entre el agente y el entorno no termina en un estado particular, sino que continua sin límite, este tipo de tareas se conocen como *tareas continuas*. En $T = \infty$ para la ecuación de retorno, esto puede causar problemas, ya que el retorno puede divergir. Para evitar esto, la recompensa inmediata al tiempo t, r_t, se multiplica por el factor de descuento γ. Con esto, la ecuación de retorno se define como:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

En este caso el agente trata de maximizar la recompensa descontada (multiplicada por el factor de descuento) que recibe a lo largo del tiempo.

(Sutton & Barto, 1998)

Como se mencionó previamente, el factor de descuento γ representa la importancia que se le da a las recompensas inmediatas y a las recompensas futuras. El factor de descuento toma valores entre 0 y 1. Un valor cercano a 0 indica que se les da mayor importancia a

recompensas inmediatas, mientras que un valor cercano a 1 indica que se les da mayor importancia a recompensas futuras.

Para unificar la notación del retorno para las tareas episódicas y las tareas continuas, se considera que para las tareas episódicas al terminar un episodio se entra a un estado especial que realiza transiciones solo hacia él mismo, recibiendo recompensa 0, esto se muestra en la **Figura 6** (Sutton & Barto, 1998).

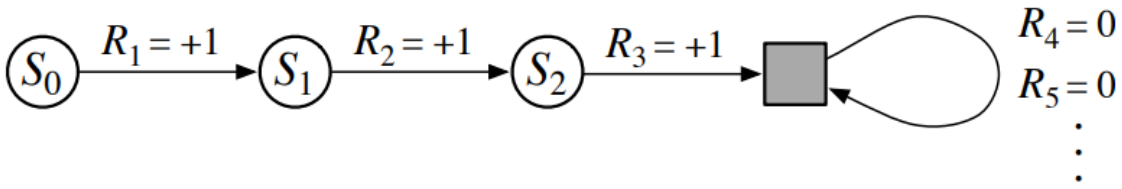

Figura 6. Estado especial para las tareas episódicas (Sutton & Barto, 1998)

En el caso especial de la **Figura 6**, R_t indica la recompensa inmediata al tiempo t, en vez del retorno.

Política

Como se mencionó previamente, la política define el comportamiento del agente en el entorno, la política indica al agente la acción que debe tomar en cada estado. La política suele denotarse por π. Al iniciar la interacción entre el agente y el entorno, la política es inicializada de manera aleatoria, por lo que el agente realizará acciones de manera aleatoria en cada estado e intentará aprender si esa acción es buena o mala, con base en las recompensas obtenidas. Luego de una serie de iteraciones, el agente aprenderá a realizar las acciones que le devuelvan la mayor recompensa, el agente habrá aprendido una política con la que puede maximizar la recompensa obtenida. Esta política se conoce como política óptima y se denota por π^*

Las políticas pueden ser clasificadas en dos categorías:

- Políticas deterministas
- Políticas estocásticas

Una política determinista es aquella que indica al agente que realice una acción particular en un estado. La política determinista mapea un estado a una acción, las políticas deterministas se denotan por μ, en vez de π. La política determinista se puede expresar como:

$$a_t = \mu(s_t)$$

Este tipo de política es una función que dado un estado $s_t \in S$ al tiempo t, mapea ese estado a la acción $a_t \in A$. Donde S es el conjunto de todos los estados del entorno y A es el conjunto de todas las acciones del entorno

Contrario a lo anterior, una política estocástica no mapea un estado a una acción particular, una política estocástica mapea un estado a una distribución de probabilidad sobre el espacio de acción, es decir, el conjunto de acciones que el agente puede tomar. Debido a esto, en un estado particular, el agente realizará diversas acciones cada vez que visite este estado, con base en la distribución de probabilidad de la política. Las políticas estocásticas se expresan como:

$$\pi(a_t|s_t)$$

(Ravichandiran, 2020)

Esto devuelve la probabilidad de tomar la acción $a_t \in A$ en el estado $s_t \in S$ al tiempo t.

Función de Valor

La función de valor o función de valor de estado $V^{\pi}(s)$ denota qué tan bueno es un estado, esto es, la recompensa que se puede esperar a largo plazo si se sigue la política π desde ese estado s, esta función, como el nombre sugiere, asigna un valor a cada estado. La función de valor del estado *s* está definida como el retorno esperado que obtiene un agente iniciando desde el estado s y siguiendo la política π. La función de valor se define como:

$$V^{\pi}(s) = \mathbb{E}_{\pi}[R_t|s_t = s] = \mathbb{E}_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} |s_0 = s]$$

Donde $\mathbb{E}_{\pi}[\cdot]$ denota el valor esperado de una variable aleatoria dado que se sigue la política π.

Dado que la función de valor depende de la política que se sigue, el valor de un estado varía dependiendo de la política seguida. La función de valor óptima $V^{*}(s)$ es aquella que entrega el máximo valor de estado respecto a todas las otras funciones de valor. Esto se expresa como:

$$V^{*}(s) = \max_{\pi} V^{\pi}(s)$$

Función de Valor estado-acción

La función de valor estado-acción, también conocida como función Q, asigna un valor a un par estado-acción, indica que tan bueno es realizar una acción en un estado dado. La función de valor estado-acción se define como el retorno esperado que obtiene un agente iniciando

desde el estado s, y realizando la acción a siguiendo la política π. El valor de un par estado acción es denotado por $Q(s,a)$ y se define como:

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[R_t|s_t = s, a_t = a] = \mathbb{E}_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} |s_t = s, a_t = a]$$

Así como para la función de valor, la función de valor estado acción depende de la política, por lo que esta función varía dependiendo de la política seguida por el agente. La función Q óptima es aquella que devuelve el máximo valor de estado-acción, o valor Q, sobre todas las funciones Q, esto se expresa como:

$$Q^{*}(s,a) = \max_{\pi} Q^{\pi}(s,a)$$

La política óptima π^* puede ser obtenida de tomar las acciones que devuelven el mayor valor Q, esto es:

$$\pi^{*}(s) = \max_{a} Q^{\pi}(s,a)$$

El valor de las funciones $V^{\pi}(s)$ y $Q^{\pi}(s,a)$ puede obtenerse de la experiencia. Cuando un agente sigue la política π y obtiene un promedio, para cada estado por el que pase el agente, de los retornos obtenidos después de pasar por ese estado, el promedio convergerá a $V^{\pi}(s)$. Similarmente, si además de para cada estado, se obtiene un promedio para cada acción tomada en cada estado, estos promedios convergerán a $Q^{\pi}(s,a)$ para cada acción.

Ecuación de Bellman

La ecuación de Bellman permite encontrar la política óptima, esto mediante la función de valor y función Q óptimas, las cuales son calculadas por la ecuación de Bellman de manera recursiva.

Ecuación de Bellman para la función de valor.

Las funciones de valor tienen la propiedad de satisfacer relaciones recursivas particulares, esto es aprovechado en el aprendizaje por refuerzo. La ecuación de valor muestra una relación entre el valor de un estado y los valores de los estados sucesivos a ese estado. Para cualquier política π y cualquier estado s (Sutton & Barto, 1998). La ecuación de Bellman permite expresar la función de valor en términos del valor del siguiente estado, esto se muestra en la siguiente ecuación:

$$V^{\pi}(s) = \mathbb{E}_{\pi}[R_t|s_t = s] = \sum_{a} \pi(a|s) \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V^{\pi}(s')]$$

(Ravichandiran, 2020)

Donde s' es el estado siguiente a s, $P(s'|s,a)$ es la probabilidad de llegar al estado s' desde el estado s realizando la acción a y $R(s,a,s')$ es la recompensa inmediata obtenida al llegar al estado s' desde el estado s realizando la acción a.

Se multiplica por la probabilidad de acción para tomar en cuenta la naturaleza estocástica de la política π.

El término $R(s,a,s') + \gamma V^{\pi}(s')$ es conocido como *Bellman backup*.

Ecuación de Bellman de la función Q

De manera similar al caso de la función de valor de estado, el valor Q de un par estado-acción puede obtenerse como la suma de la recompensa inmediata y el valor descontado (multiplicado por el factor de descuento) Q del siguiente par estado acción, esto es:

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[R_t|s_t = s, a_t = a] = \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma \sum_{a'} \pi(a'|s')\, Q^{\pi}(s',a')]$$

En este caso se elimina el término $\sum_a \pi(a|s)$ debido a que en el caso de la función Q $Q(s,a)$ ya se está indicando la acción a que se realizará en el estado s. En cambio, se añade el término $\sum_{a'} \pi(a'|s')$ ya que en el siguiente estado s', se debe elegir una acción a' siguiendo la política π para calcular el valor Q del siguiente par estado acción $Q(s',a')$.

(Ravichandiran, 2020)

Redes neuronales artificiales

Las redes neuronales artificiales (ANN, por sus siglas en inglés) están inspiradas en los primeros modelos de procesamiento sensorial del cerebro. En 1943, Warren McCulloch y Walter Pitts modelaron una neurona como un switch que recibe información de otras neuronas y dependiendo del aporte total ponderado de estas, la neurona es activada o permanece inactiva. En 1960 se demostró que las redes de las neuronas modeladas por McCulloch y Pitts tienen propiedades similares a las del cerebro: pueden llevar a cabo un sofisticado reconocimiento de patrones, e incluso pueden funcionar si algunas neuronas son destruidas (Krogh, 2008).

El siguiente paso importante en el desarrollo de las ANN fue hecho por Frank Rosenblatt en 1957 con una de las arquitecturas más simples de ANN, el *perceptrón*. El perceptrón está basado en una neurona llamada *Linear Threshold Unit* (LTU), ahora las entradas y salidas son números, en vez de neuronas activadas o desactivadas como en el caso del modelo propuesto por McCulloch y Pitts. En el perceptrón, a cada conexión de entrada se le asigna un peso. La LTU calcula la suma ponderada de las entradas y luego aplica la función escalón a esa suma y devuelve el resultado: $h_w(\boldsymbol{x}) = \text{step}(z)$ (Géron, 2020), donde $\boldsymbol{x}$ es el vector con las entradas del perceptrón, z es la suma ponderada de las entradas $z = \boldsymbol{x}^T\boldsymbol{w} = w_1x_1 + w_2x_2 + \cdots + w_nx_n$, siendo $\boldsymbol{w}$ el vector que contiene las entradas de los pesos y $\text{step}(\cdot)$ la función escalón. En la **Figura 7** se puede observar el diagrama del perceptrón.

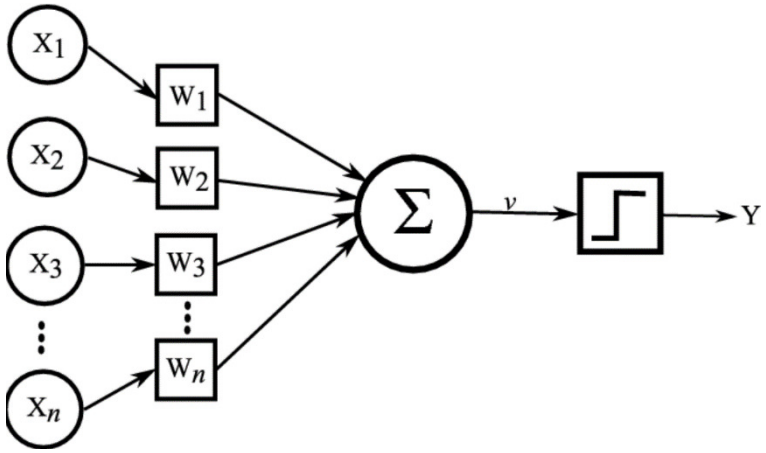

Figura 7. Diagrama del perceptrón (Kuipers & Prasad, 2021)

En una monografía escrita en 1969, Marvin Minsky y Seymour Papert destacan varias debilidades serias de los perceptrones, en particular, el hecho de que son incapaces de resolver problemas triviales como el problema de clasificación XOR, u OR exclusivo (Géron, 2020). Esto debido a que los perceptrones son modelos lineales, por lo que en un problema de clasificación tratarán de dividir las clases usando una línea recta, con esta línea recta no

es posible separa las clases en el problema XOR. En la **Figura 8** se muestra que no es posible separar las dos clases del problema XOR con una línea recta, por lo que no es posible resolver este problema usando el perceptrón, mientras que el problema OR si es posible solucionarlo con un modelo lineal como el perceptrón.

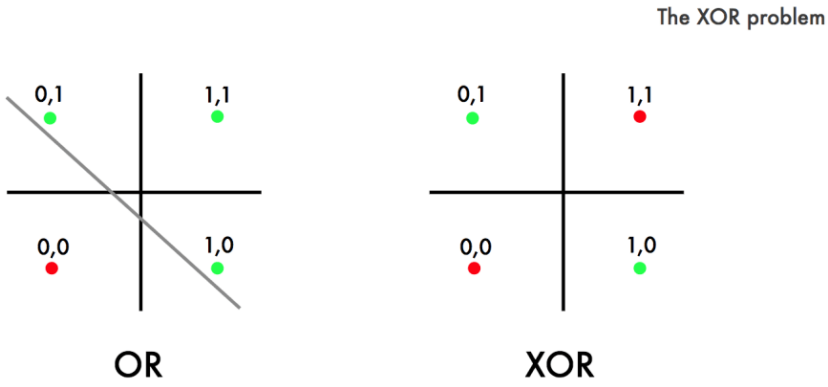

Figura 8. Problema XOR (Ahire, 2020)

Las limitaciones del perceptrón pueden ser eliminadas apilando múltiples perceptrones. La ANN que resulta de esto se conoce como *perceptrón multicapa* (MLP, por sus siglas en inglés). Un MLP (**Figura 9**) está compuesto por una capa de entrada, una o más capas ocultas, las capas ocultas son las capas que se encuentran entre la capa de entrada y la de salida, y una capa de salida (Géron, 2020).

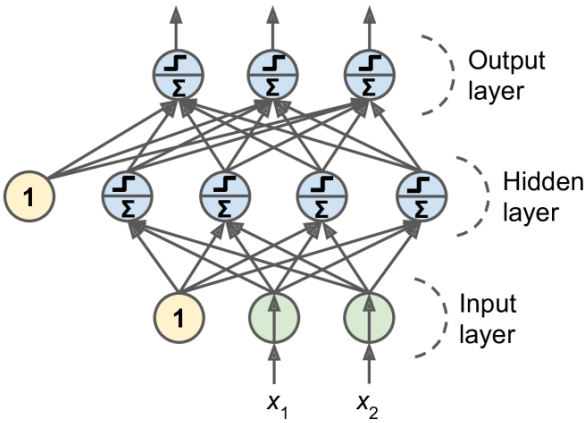

Figura 9. Perceptrón multicapa (Géron, 2020)

En la **Figura 9** se puede observar una neurona extra (neurona amarilla con el número 1), esta neurona se suele añadir a las capas de la ANN, excepto a la capa de salida, y se conoce como *neurona de sesgo* o *bias*, la salida de esta neurona siempre es un uno.

Cuando todas las neuronas en una capa están conectadas a cada neurona en la capa anterior, la capa se conoce como *capa totalmente conectada* o capa *densa* (Géron, 2020), este es el

caso de las capas en la **Figura 9**. A partir de ahora, el término red neuronal se usará para referirse a una red neuronal artificial.

Cuando una ANN contiene más de dos capas ocultas, esta se conoce como *red neuronal profunda* (DNN, por sus siglas en inglés). El campo de Deep Learning estudia las DNN, y de manera más general, modelos que contienen apilamientos profundos de cálculos (Géron, 2020). La **Figura 10** muestra la diferencia entre una ANN simple y una DNN.

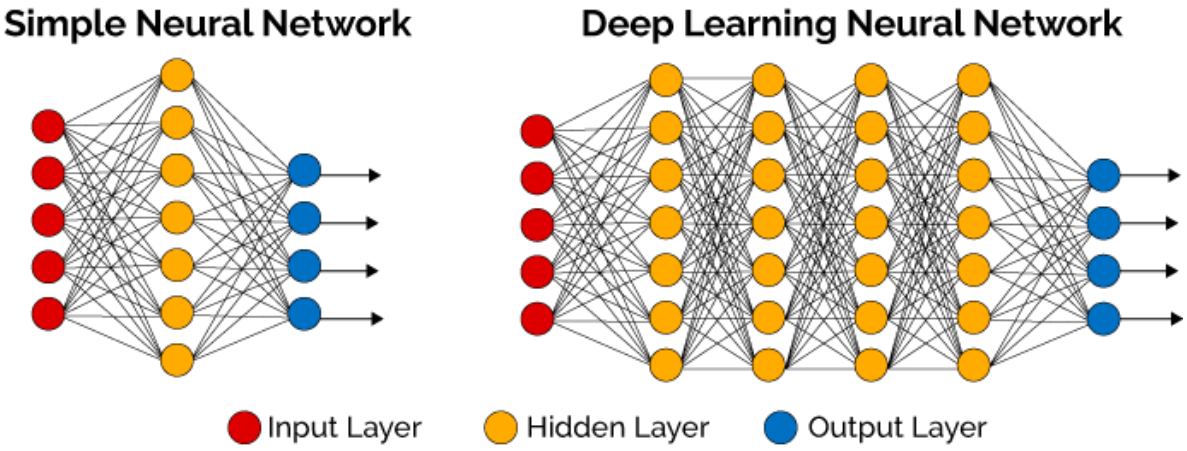

Figura 10. ANN vs DNN (Vázquez, 2017)

Deep Learning

Como se mencionó antes, el Deep Learning es un área especial de Machine Learning, diseñada para tratar con problemas de altas dimensiones. Deep Learning es una categoría transversal de Machine Learning, ya que ha sido usado para resolver tareas de aprendizaje no supervisado, supervisado y aprendizaje por refuerzo (Bedolla, Padierna, & Castañeda, 2021). En particular, este trabajo se centra en el uso de Deep Learning para resolver problemas de aprendizaje por refuerzo.

Las técnicas convencionales de Machine Learning son limitadas en su habilidad para procesar datos de forma cruda, es decir, sin que estos hayan sido previamente preprocesados. Debido a esto, por mucho tiempo, construir un sistema de Machine Learning requería una ingeniería cuidadosa y un dominio del tema considerable para diseñar un extractor de características que transformara los datos crudos, sin preprocesamiento, en una representación viable de la cual el sistema pudiera aprender. Contrario a esto, el aprendizaje de características, o representation learning es un conjunto de métodos que permite que se le pasen datos crudos a una máquina para que esta automáticamente descubra las representaciones necesarias para resolver problemas como clasificación o detección (LeCun, Bengio, & Hinton, 2015). Los métodos de Deep Learning son un ejemplo de estas técnicas. En la **Figura 11** se muestra lo anterior para el caso de un clasificador de imágenes de coches, mientras que con técnicas tradicionales de Machine Learning se debe realizar una extracción de características manual para poder pasar estas características al clasificador, en Deep Learning el modelo realiza automáticamente la extracción de características, por lo que se le pueden pasar datos crudos al modelo.

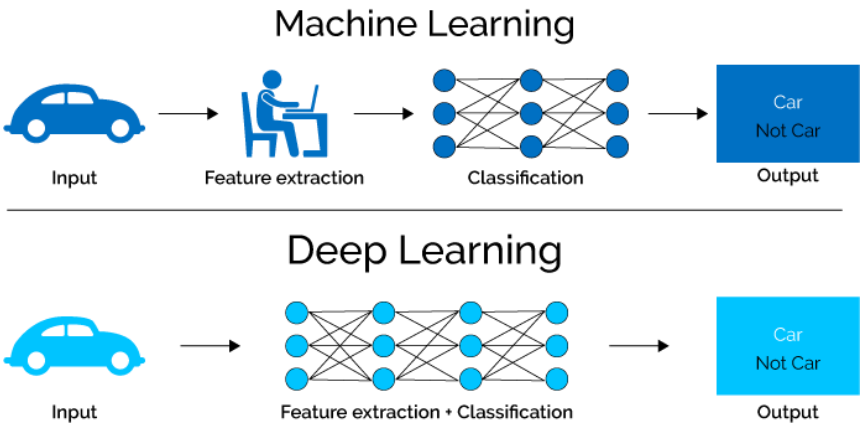

Figura 11. Machine Learning vs Deep Learning (Mahapatra, 2018)

Los métodos de Deep Learning cuentan con varios niveles de aprendizaje de representaciones, obtenidos al hacer una composición de modelos simples, pero no lineales, cada uno de estos transforma la representación desde un nivel (empezando desde la entrada de datos crudos) hasta otra representación de nivel más alto y un poco más abstracto. Con la composición de suficientes de estas transformaciones, se pueden aprender funciones bastante complejas. Por ejemplo, en el caso de imágenes, las representaciones aprendidas en las primeras capas

normalmente representan la presencia o ausencia de bordes en orientaciones y ubicaciones particulares dentro de la imagen. La segunda capa normalmente detecta patrones encontrando arreglos particulares de bordes. La tercera capa puede ensamblar estos patrones en

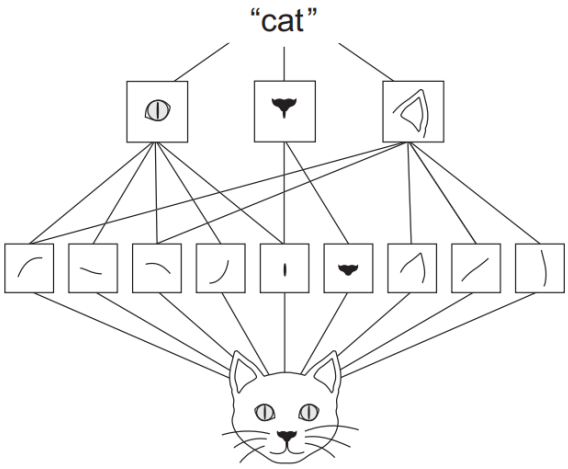

Figura 12. Representaciones aprendidas por cada capa de un modelo de Deep Learning (Chollet, 2018)

combinaciones más grandes que corresponden a partes de objetos familiares, capas subsecuentes detectarían objetos como combinaciones de estos objetos (LeCun, Bengio, & Hinton, 2015). En la **Figura 12** se muestran estas representaciones que aprende un modelo de Deep Learning, para el caso de la imagen de un gato.

El aspecto más importante de Deep Learning es lo que se muestra en la **Figura 11**, las características o representaciones no son extraídas por humanos, sino que son aprendidas a partir de los datos.

Dentro de los métodos de Deep Learning, existe uno en particular que ha sido ampliamente estudiado, estas son las redes neuronales convolucionales (CNN, por sus siglas en inglés).

Redes neuronales convolucionales

Las redes neuronales convolucionales (CNN) son una arquitectura conocida de Deep Learning inspiradas en el mecanismo de percepción visual de los seres vivos. En 1980 Kunihiko Fukushima propuso el neocognitrón, el cual se considera el predecesor de las CNN, el neocognitrón se inspiraba en el trabajo de David Hubel y Torsten Wiesel, quienes en 1959 descubrieron que, en la corteza visual de los animales, las células son las responsables de detectar la luz en los campos receptivos. Posteriormente, en un artículo de 1990, Yann LeCun, y otros, desarrollaron una ANN multicapa llamada *LeNet-5* la cual era capaz de clasificar dígitos escritos a mano, marcando el inicio de las CNN. LeNet-5 podía obtener representaciones de las imágenes originales, lo que hacía posible reconocer patrones visuales directamente de las imágenes con poco o ningún preprocesamiento. Sin embargo, estas redes no pudieron escalar a problemas más complejos debido a la falta de datos de entrenamiento

y a la falta de poder computacional (Gu , y otros, 2018). Sin embargo, en los últimos años esto ha dejado de ser una limitación por lo que las CNN se han convertido en la técnica de Deep Learning más estudiada.

Las CNN están diseñadas para procesar datos que vienen en forma de múltiples arreglos, por ejemplo, una imagen de color compuesta por tres arreglos de dos dimensiones que contienen las intensidades de los pixeles para los tres canales de color. Muchos datos se encuentran en la forma de arreglos múltiples: arreglos 1D para las señales y secuencias, 2D para las imágenes o espectrogramas de audio y 3D para videos o imágenes volumétricas (LeCun, Bengio, & Hinton, 2015).

Como el nombre sugiere, las CNN son redes que llevan a cabo la operación de convolución, la cual es un tipo especial de operación lineal. Entonces, las CNN son redes neuronales que emplean, en al menos una de sus capas, la operación de convolución en vez de una multiplicación de matrices (Goodfellow, Bengio, & Courville, 2016), como lo hacen las redes neuronales tradicionales. Por lo anterior, a continuación, se describe la operación de convolución.

Convolución

En su forma más general, la convolución es una operación sobre dos funciones de argumento real (Goodfellow, Bengio, & Courville, 2016). La convolución está definida por la siguiente expresión:

$$s(t) = \int_{-\infty}^{\infty} x(\tau)\omega(t-\tau)d\tau$$

La operación de convolución suele denotarse por un asterisco, esto es:

$$s(t) = (x * \omega)(t)$$

En la terminología de redes convolucionales, al primer argumento de la convolución (en este caso x) se le conoce como *entrada*, y al segundo argumento (en este caso ω) se le conoce como *kernel*. A la salida normalmente se le conoce como *mapa de características.*

Lo anterior es para el caso de funciones de argumento real, sin embargo, cuando se trabaja en una computadora, los datos estarán discretizados, por lo que no será posible usar esta ecuación. Para solucionar lo anterior, se introduce la convolución discreta:

$$s(t) = (x * \omega)(t) = \sum_{\tau=-\infty}^{\infty} x(\tau)\omega(t-\tau)$$

Modelo de aprendizaje por refuerzo para eliminación de ruido en imágenes

PixelRL es un modelo de aprendizaje por refuerzo para procesamiento de imágenes con recompensas por píxel propuesto por (Furuta, Inoue, & Yamasaki, 2019). Aunque pixelRL puede resolver tareas de procesamiento de imágenes como eliminación de ruido, restauración de imágenes y mejoramiento de color en imágenes, este trabajo se centra en la tarea de eliminación de ruido. En pixelRL, cada píxel cuenta con un agente, y el agente cambia el valor de su píxel realizando una acción de las acciones en el espacio de acción. El valor de cada píxel es considerado como el estado actual y este se actualiza iterativamente por cada acción del agente.

A3C (Asynchronous Advantage Actor Critic)

PixelRL se basa en el modelo A3C (Asynchronous Advantage Actor Critic), el cual cuenta con dos redes: la red de política y la red de valor. Los parámetros de cada una de estas redes se denotan por θ_p y θ_v, respectivamente. Ambas redes reciben como entrada el estado actual $s^{(t)}$, donde $s^{(t)}$ denota el estado al tiempo t. La salida de la red de valor es $V(s^{(t)})$, la recompensa esperada si se inicia el episodio desde el estado $s^{(t)}$, esto indica qué tan bueno es el estado actual.

Por otra parte, la red de política devuelve la política $\pi(a^{(t)}|s^{(t)})$, la cual es la probabilidad de realizar la acción $a^{(t)}$, donde $a^{(t)}$ es una de las acciones en el espacio de acción. El tamaño de la salida de la red de política es igual al número de acciones en el espacio de acción.

Para poder encontrar los parámetros óptimos de estas dos redes es necesario encontrar el gradiente de sus parámetros θ_p y θ_v para poder actualizar los parámetros en la dirección de mayor disminución del error de la red, esta es la dirección de menos el gradiente.

En el caso de θ_v, se usa el retorno para poder calcular el gradiente, el retorno al tiempo t se denota por:

$$R^{(t)} = r^{(t)} + \gamma r^{(t+1)} + \gamma^2 r^{(t+2)} + \cdots + \gamma^{n-1} r^{(t+n-1)} + \gamma^n V\left(s^{(t+n)}\right)$$

Donde γ es el factor de descuento y $V\left(s^{(t+n)}\right)$ es el valor del estado $s^{(t+n)}$, se agrega el valor del estado $s^{(t+n)}$, ya que esto indica la recompensa esperada al iniciar desde el estado $s^{(t+n)}$, y el retorno se define como la suma de las recompensas obtenidas después del tiempo t. Finalmente, n es el número de pasos de aprendizaje.

Con esto, el gradiente de θ_v se calcula como:

$$d\theta_v = \nabla_{\theta_v} \left(R^{(t)} - V\left(s^{(t)}\right)\right)^2$$

Para el caso de θ_p, se introduce la función *advantage*, $A(a^{(t)}, s^{(t)})$, la cual se define como:

$$A\left(a^{(t)}, s^{(t)}\right) = R^{(t)} - V(s^{(t)})$$

Esta función indica qué tan buena es una acción $a^{(t)}$ desde el estado $s^{(t)}$ respecto al promedio de las acciones.

Con esto, se define el gradiente de θ_p como:

$$d\theta_p = -\nabla_{\theta_p} \log \pi(a^{(t)}|s^{(t)})\, A\left(a^{(t)}, s^{(t)}\right)$$

(Furuta, Inoue, & Yamasaki, 2019)

PixelRL

Sea $\boldsymbol{I_i}$ el i-ésimo píxel de una imagen $\boldsymbol{I}$ que contiene N pixeles. Como se mencionó previamente, cada píxel cuenta con un agente, y la política de cada agente se denotará por $\pi_i(a_i^{(t)}|s_i^{(t)})$ con $i = 1, 2, 3, \dots, N$, aquí $a_i^{(t)}$ es la acción tomada por el agente i al tiempo t, y $\boldsymbol{s_i^{(t)}}$ es el estado del agente i al tiempo t. El espacio de acción se denota por $\mathcal{A}$, y $s_i^{(0)} = I_i$, es decir, el píxel i en la imagen original $\boldsymbol{I}$ con ruido se denota por $\boldsymbol{s_i^{(0)}}$. El espacio de acción es un conjunto de acciones predefinidas, estas acciones se muestran en la **Tabla 1**, estas son las acciones que cada pixel puede tomar para eliminar el ruido de la imagen.

Acción	Tamaño del filtro	Parámetros
Filtro de caja	5x5	-
Filtro bilateral	5x5	$\sigma_c = 1.0, \sigma_S = 5.0$
Filtro bilateral	5x5	$\sigma_c = 0.1, \sigma_S = 5.0$
Filtro de mediana	5x5	-
Filtro Gaussiano	5x5	$\sigma = 1.5$
Filtro Gaussiano	5x5	$\sigma = 0.5$
Valor del píxel +1	-	-
Valor del píxel -1	-	-
No hacer nada	-	-

Tabla 1. Espacio de acción PixelRL

Para aplicar los filtros de la **Tabla 1** a una imagen se realiza una convolución entre la imagen que se desea filtrar y el filtro a usar.

Cada agente recibe una recompensa $\boldsymbol{r^{(t)}} = (r_1^{(t)}, \dots, r_N^{(t)})$ del entorno al realizar una acción $\boldsymbol{a^{(t)}} = (a_1^{(t)}, \dots, a_N^{(t)})$ al tiempo t, además de pasar al siguiente estado $\boldsymbol{s^{(t+1)}} = (s_1^{(t+1)}, \dots, s_N^{(t+1)})$. Al ser un problema de aprendizaje por refuerzo, el objetivo del modelo, en este caso pixelRL, es encontrar las políticas óptimas $\boldsymbol{\pi} = (\pi_1, \dots, \pi_N)$ que maximicen la media de las recompensas esperadas sobre todos los pixeles, esto es:

$$\boldsymbol{\pi}^* = \underset{\boldsymbol{\pi}}{\operatorname{argmax}}\, \mathbb{E}_{\boldsymbol{\pi}} \sum_{t=0}^{\infty} \gamma^t \bar{r}^{(t)}$$

Donde $\bar{r}^{(t)}$ es la media de las recompensas sobre todos los pixeles al tiempo t, esto es:

$$\bar{r}^{(t)} = \frac{1}{N} \sum_{i=1}^{N} r_i^{(t)}$$

Para esto, (Furuta, Inoue, & Yamasaki, 2019) proponen usar una *Fully Convolutional Network* (FCN), la cual es una red neuronal que solo realiza operaciones de convolución. Esto permite que todos los agentes compartan parámetros y también hace posible la paralelización de los cálculos para los N agentes usando una GPU, haciendo el entrenamiento más eficiente. En la **Figura 13** se muestra la arquitectura descrita, la cual es una extensión de A3C a la forma de FCN, de aquí se observa que la imagen primero se pasa por la FCN con parámetros compartidos para todos los agentes, posteriormente los mapas de características obtenidos por la FCN se pasan por las redes de valor y política, para obtener el valor del estado y la política, respectivamente.

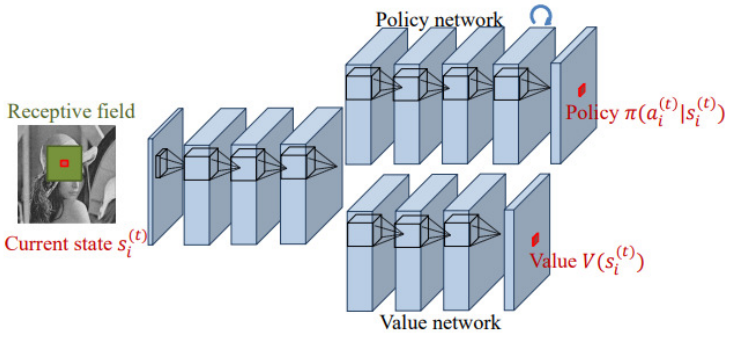

Figura 13. Arquitectura del Fully Convolutional A3C (Furuta, Inoue, & Yamasaki, 2019)

Para calcular la recompensa se utiliza la imagen original sin ruido, la cual se denota por $\boldsymbol{I}^{target}$, la recompensa al tiempo t para el píxel i se calcula usando la siguiente expresión:

$$r_i^{(t)} = \left(I_i^{target} - s_i^{(t)}\right)^2 - \left(I_i^{target} - s_i^{(t+1)}\right)^2$$

Donde I_i^{target} es el píxel i de la imagen original sin ruido. Esta expresión denota qué tanto se disminuye el error cuadrático en el píxel i al aplicar la acción $a_i^{(t)}$ en el estado $s_i^{(t)}$. Esta expresión es la que se busca maximizar.

Debido a la estructura de las imágenes (arreglo de dos dimensiones), (Furuta, Inoue, & Yamasaki, 2019) proponen un método llamado *Reward Map Convolution* (RMC) para mejorar el desempeño de pixelRL aprovechando esta propiedad de la estructura de las imágenes.

Reward Map Convolution (RMC)

En la **Figura 13** se observa un cuadrado verde en la imagen de entrada al modelo, esto se conoce como *campo receptivo* e indica los valores que tomará en cuenta cada agente, cuando este campo receptivo es de tamaño 1x1, el agente solo tomará en cuenta el valor de su píxel, por lo que los valores de los N pixeles son independientes, en este caso los gradientes son similares a los mostrados anteriormente para las dos redes del A3C, en este caso se toma el promedio sobre los gradientes de los N agentes, estos se muestran a continuación, por simplicidad, se toma el caso donde $n = 1$:

$$R_i^{(t)} = r_i^{(t)} + \gamma V\left(s_i^{(t+1)}\right)$$

$$d\theta_v = \nabla_{\theta_v} \frac{1}{N} \sum_{i=1}^{N} \left(R_i^{(t)} - V\left(s_i^{(t)}\right)\right)^2$$

$$A\left(a_i^{(t)}, s_i^{(t)}\right) = R_i^{(t)} - V\left(s_i^{(t)}\right)$$

$$d\theta_p = -\nabla_p \frac{1}{N} \sum_{i=1}^{N} \log \pi(a_i^{(t)} | s_i^{(t)}) A\left(a_i^{(t)}, s_i^{(t)}\right)$$

El subíndice i indica el agente sobre el que se está trabajando.

Lo anterior es para un campo receptivo de 1x1, donde el agente toma en cuenta solo el valor de su píxel. Para el caso de campos receptivos más grandes, como el de la **Figura 13** las redes de política y valor toman en cuenta no solo el valor del píxel del agente, sino los valores de los píxeles vecinos para calcular la política y el valor del estado para el agente. Esto quiere decir que la acción $a_i^{(t)}$ no solo afecta al estado $s_i^{(t+1)}$, sino que también afecta a las políticas y valores de $\mathcal{N}(i)$ al tiempo $(t+1)$, donde $\mathcal{N}(i)$ es una ventana centrada en el píxel i, es decir, la acción $a_i^{(t)}$ afecta al estado $s_i^{(t+1)}$ y también a los vecinos del píxel i al tiempo $(t+1)$. Para tomar en cuenta esta vecindad que es afectada por la acción $a_i^{(t)}$ se reescribe el retorno como:

$$R_i^{(t)} = r_i^{(t)} + \gamma \sum_{j \in \mathcal{N}(i)} \omega_{i-j} V\left(s_j^{(t+1)}\right)$$

Donde ω_{i-j} es el peso que indica qué tanto se consideran los valores V de los pixeles vecinos al tiempo $(t+1)$. Se puede considerar a $\boldsymbol{\omega}$ como un kernel de convolución cuyos pesos son aprendidos simultáneamente a los parámetros θ_v y θ_p. Así como para θ_p y θ_v, el gradiente para $\boldsymbol{\omega}$ se puede calcular como:

$$d\boldsymbol{\omega} = -\nabla_{\boldsymbol{\omega}} \frac{1}{N} \sum_{i=1}^{N} \log\, \pi\left(a_i^{(t)} \middle| s_i^{(t)}\right) \left(R_i^{(t)} - V\left(s_i^{(t)}\right)\right) + \nabla_{\boldsymbol{\omega}} \frac{1}{N} \sum_{i=1}^{N} \left(R_i^{(t)} - V\left(s_i^{(t)}\right)\right)^2$$

El primer término de esta expresión favorece una recompensa total esperada mayor, mientras que el segundo término funciona como un regularizador de tal forma que el retorno R_i no sea desviado de la predicción $V\left(s_i^{(t)}\right)$ por la convolución.

Con lo anterior, y considerando que cada píxel i tiene una coordenada 2D (i_x, i_y), el segundo término puede verse como una convolución entre $\boldsymbol{\omega}$ y $\boldsymbol{V}\left(\boldsymbol{s}^{(t+1)}\right)$. Con esto, es posible reescribir el retorno como:

$$\boldsymbol{R}^{(t)} = \boldsymbol{r}^{(t)} + \gamma \boldsymbol{\omega} * \boldsymbol{V}\left(\boldsymbol{s}^{(t+1)}\right)$$

Aquí $\boldsymbol{R}^{(t)}$, $\boldsymbol{r}^{(t)}$ y $\boldsymbol{V}\left(\boldsymbol{s}^{(t+1)}\right)$ son las matrices cuyos elementos c son $R_i^{(t)}$, $r_i^{(t)}$ y $V\left(s_i^{(t+1)}\right)$, respectivamente.

Nuevamente, esto es para el caso donde $n = 1$. De manera más general, para el caso de un aprendizaje de n pasos y usando la forma matricial podemos definir el retorno como:

$$\boldsymbol{R}^{(t)} = \boldsymbol{r}^{(t)} + \gamma \boldsymbol{\omega} * \boldsymbol{r}^{(t+1)} + \gamma^2 \boldsymbol{\omega}^2 * \boldsymbol{r}^{(t+2)} + \cdots + \gamma^{n-1} \boldsymbol{\omega}^{n-1} * \boldsymbol{r}^{(t+n-1)} + \gamma^n \boldsymbol{\omega}^n * \boldsymbol{V}\left(\boldsymbol{s}^{(t+n)}\right)$$

De manera similar, se puede reescribir la función advantage en forma matricial como:

$$\boldsymbol{A}\left(\boldsymbol{a}^{(t)}, \boldsymbol{s}^{(t)}\right) = \left(\boldsymbol{R}^{(t)} - \boldsymbol{V}\left(\boldsymbol{s}^{(t)}\right)\right)$$

Con lo anterior, es posible reescribir los gradientes de las redes de valor y política en forma matricial como:

$$d\theta_v = \nabla_{\theta_v} \frac{1}{N} \mathbf{1}^\top \left\{\left(\boldsymbol{R}^{(t)} - \boldsymbol{V}\left(\boldsymbol{s}^{(t)}\right)\right) \odot \left(\boldsymbol{R}^{(t)} - \boldsymbol{V}\left(\boldsymbol{s}^{(t)}\right)\right)\right\} \mathbf{1}$$

$$d\theta_p = -\nabla_{\theta_p} \frac{1}{N} \mathbf{1}^\top \left\{\log \boldsymbol{\pi}\left(\boldsymbol{a}^{(t)} \middle| \boldsymbol{s}^{(t)}\right) \odot \boldsymbol{A}\left(\boldsymbol{a}^{(t)}, \boldsymbol{s}^{(t)}\right)\right\} \mathbf{1}$$

Donde $\boldsymbol{\pi}\left(\boldsymbol{a}^{(t)} \middle| \boldsymbol{s}^{(t)}\right)$ es una matriz cuyo elemento (i_x, i_y) es $\pi(a_i^{(t)} | s_i^{(t)})$, $\mathbf{1}$ es un vector con todos sus elementos igual a uno, y $\odot$ es la multiplicación elemento a elemento.

De igual forma, es posible reescribir el gradiente de $\boldsymbol{\omega}$ en forma matricial como:

$$d\boldsymbol{\omega} = -\nabla_{\boldsymbol{\omega}} \frac{1}{N} \mathbf{1}^\top \left\{\log \boldsymbol{\pi}\left(\boldsymbol{a}^{(t)} \middle| \boldsymbol{s}^{(t)}\right) \odot \boldsymbol{A}\left(\boldsymbol{a}^{(t)}, \boldsymbol{s}^{(t)}\right)\right\} \mathbf{1}$$

$$+\nabla_{\boldsymbol{\omega}} \frac{1}{N} \mathbf{1}^\top \left\{\left(\boldsymbol{R}^{(t)} - \boldsymbol{V}\left(\boldsymbol{s}^{(t)}\right)\right) \odot \left(\boldsymbol{R}^{(t)} - \boldsymbol{V}\left(\boldsymbol{s}^{(t)}\right)\right)\right\} \mathbf{1}$$

Métricas de desempeño

Una de las partes fundamentales en Machine Learning es la evaluación del modelo, para poder determinar si el modelo desarrollado es adecuado para su aplicación o si es necesario realizar cambios en este para que pueda tener un mejor desempeño. Dependiendo de la tarea que se busca resolver con un modelo de Machine Learning existen distintas métricas, por ejemplo, en el caso de clasificación una métrica que puede usarse es el *accuracy*, para regresión, se puede usar el *error cuadrático medio* para evaluar al modelo, finalmente en el caso de eliminación de ruido una métrica usada es la *proporción máxima de señal a ruido.* A continuación, se describen las métricas de accuracy y proporción máxima de señal a ruido, debido a que en el presente trabajo se resuelve un problema de clasificación, además del problema central del trabajo, la eliminación de ruido de imágenes.

Accuracy

El accuracy es una métrica usada para evaluar modelos de clasificación, se define como la razón entre el número de predicciones que el modelo realizó correctamente y el número total de predicciones realizadas por el modelo. El accuracy se define por la siguiente expresión:

$$Accuracy = \frac{Número\ de\ predicciones\ correctas}{Número\ total\ de\ predicciones}$$

En el caso de clasificación binaria, es decir, cuando solo hay dos clases en las que se pueden clasificar los datos, es posible calcular el accuracy en términos de positivos y negativos, esto es:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Donde TP indica un verdadero positivo, esto es cuando una instancia es clasificada como positiva y sí pertenece a la clase negativa. TN indica un verdadero negativo, es decir, una instancia es clasificada como negativa y sí pertenece a la clase negativa. FP denota un falso positivo, es decir, una instancia perteneciente a la clase negativa es clasificada como positiva. De manera similar para los falsos negativos FN, estos denotan que una instancia de la clase positiva fue clasificada como positiva.

El accuracy toma valores entre 0 y 1, donde 0 indica que todas las instancias fueron clasificadas de manera incorrecta, mientras que un accuracy de 1 indica que todas las instancias fueron clasificadas de manera correcta.

Proporción máxima de señal a ruido

La proporción máxima de señal a ruido (PSNR, por sus siglas en inglés) es una métrica comúnmente usada en mejoramiento de imágenes y para medir la calidad de una imagen después de su compresión, se define como la razón entre la energía máxima de una señal y el ruido que afecta la fidelidad de la representación de la señal, para este trabajo, una señal es una imagen. La PSNR se suele expresar en términos de la escala de decibeles (dB). Para el caso de arreglos de dos dimensiones o matrices, como es el caso de las imágenes, PSNR se define por la siguiente expresión:

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right)$$

Donde MAX_I es el valor máximo que puede tomar un píxel en la imagen prístina, en el caso de imágenes almacenadas con tipo de dato entero de 8 bits, este valor es 255, y MSE es el error cuadrático medio definido como:

$$MSE = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\|I(i,j) - K(i,j)\|$$

Donde I es la imagen prístina, K es la imagen degradada, M es el número de filas, y N el número de columnas en las imágenes, las imágenes deben tener el mismo tamaño.

No existe un rango especifico de valores que tome PSNR, sin embargo, un PSNR mayor se asocia con una mejor calidad de reconstrucción de una imagen.

Metodología

Materiales

Los experimentos que se describen a continuación fueron llevados a cabo usando el lenguaje de programación *Python*. Además, se hizo uso de varias librerías disponibles para el lenguaje Python, como *numpy* que permite trabajar con arreglos multidimensionales de datos, *OpenCV* para la manipulación de imágenes y *Scikit-learn* la cual provee diversas funcionalidades útiles para Machine Learning. Para trabajar con redes neuronales se utilizaron dos frameworks, el primero es *Tensorflow*, y el segundo es el framework de *Chainer*, el cual permite una mayor flexibilidad al trabajar con modelos de aprendizaje por refuerzo.

Debido a la gran cantidad de datos y cálculos requeridos para trabajar con Deep Learning, las unidades de procesamiento gráfico (GPU) son de suma importancia para poder entrenar estas arquitecturas, ya que las GPU son considerablemente más rápido que las CPU debido a que las primeras cuentan con una mayor cantidad de núcleos, lo que permite llevar a cabo cálculos en paralelo. Para poder hacer uso de este hardware, se utilizó la plataforma *Google Colaboratory*, esta plataforma permite ejecutar código de Python desde la nube, en formato de *Jupyter Notebooks.* Una de las ventajas de *Google Colaboratory* es que permite ejecutar el código usando una GPU para acelerar el entrenamiento en el caso de modelos de Deep Learning.

Google Colaboratory permite ejecutar código usando una GPU hasta por 12 horas continuas, esto de manera gratuita, existe la posibilidad de aumentar este tiempo obteniendo planes de pago, sin embargo, para este trabajo se utilizó el plan gratuito. El hardware disponible en *Google Colaboratory* puede variar según la disponibilidad, pero generalmente existe la posibilidad de usar una tarjeta gráfica NVIDIA TESLA K80 con 12 GB de RAM, igualmente a 12 GB de RAM en el sistema, y 60 GB de almacenamiento.

Aunque la GPU supera en velocidad a la CPU en la mayoría de los cálculos, existen algunos cálculos que son llevado a cabo de manera más rápida y eficiente en la CPU, por lo que es necesario poder realizar algunos procesos en la GPU y otros en la CPU, para esto se usa la librería *CuPy*, la cual permite tener control sobre el hardware en el que se ejecutarán los procesos.

Experimentos desarrollados

En este trabajo se realizaron tres experimentos, cada uno de estos se describe de manera breve a continuación, posteriormente en la siguiente sección se explicada la metodología seguida en cada uno de estos experimentos.

Primer experimento: Perceptrón aplicado a datasets artificiales

Como primer paso, se implementó desde cero el perceptrón, para poder tener un mejor entendimiento sobre el funcionamiento de las redes neuronales. El objetivo del perceptrón fue resolver un problema de clasificación binaria, esto es, se clasificarán las instancias en el dataset en dos clases, para tres datasets sintéticos, uno de estos linealmente separable y los dos restantes no linealmente separables, es decir, no se pueden separar las dos clases usando una línea recta. Estos datasets se pueden apreciar en la **Figura 14**.

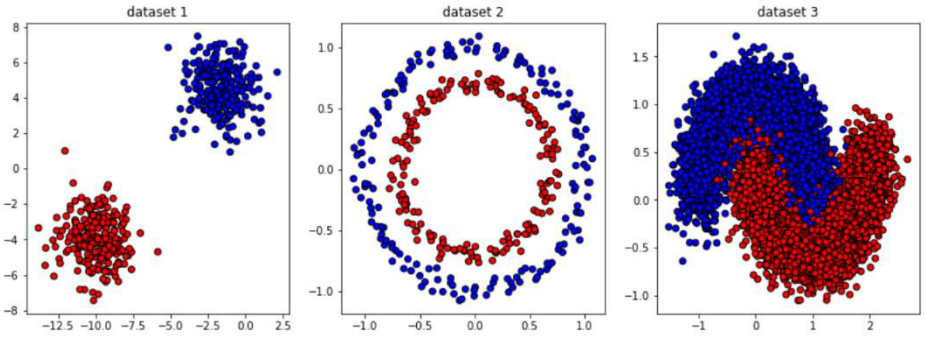

Figura 14. Datasets sintéticos para clasificación binaria

Adicional a la implementación propia del perceptrón, se resuelven los datasets anteriores usando la implementación del perceptrón proporcionada por la librería *Scikit-learn*, esto para comparar los resultados obtenidos con ambas implementaciones.

Segundo experimento: CNN y GAN para generación de imágenes del dataset CIFAR-10

El segundo experimento consiste en implementar una arquitectura de red convolucional para la generación de imágenes, esto con el propósito de comprender de mejor manera los conceptos de Deep Learning. El objetivo de este modelo implementado es la generación de imágenes sintéticas con el aspecto de las imágenes del dataset CIFAR-10 (**Figura 17**).

Este tipo de modelos generativos se conocen como *redes generativas antagónicas* (GAN, por sus siglas en inglés), las GAN constan de dos redes neuronales, una conocida como discriminador y la otra conocida como generador. El trabajo del generador es, como su nombre lo sugiere, generar imágenes parecidas a las del dataset de entrenamiento, a partir de un vector de números aleatorios. El trabajo del discriminador es decidir si una imagen es real (proveniente del dataset de entrenamiento) o falsa (creada por el generador), el discriminador

toma como entrada una imagen. En la **Figura 15** se muestra un diagrama simplificado de una GAN.

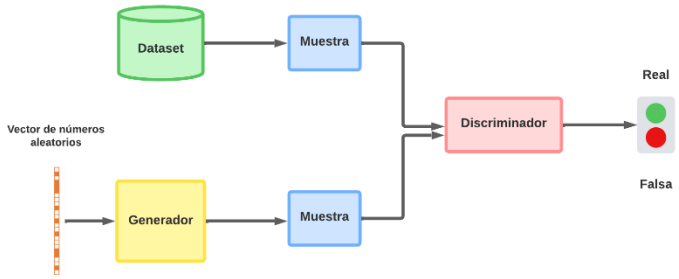

Figura 15. Diagrama de GAN

Tercer experimento: Replica de artículo científico para la eliminación de ruido de imágenes usando Redes Neuronales Convolucionales A3C

En el tercer experimento se busca implementar la arquitectura propuesta por (Furuta, Inoue, & Yamasaki, 2019) para la eliminación de ruido en imágenes mediante el uso de aprendizaje por refuerzo, una vez se haya entrenado el modelo y sea capaz de eliminar ruido de imágenes comunes, este modelo se probará con imágenes médicas.

Metodología general

A continuación, se presenta la metodología general que se llevara a cabo para el desarrollo de los experimentos contenidos en el presente trabajo, la metodología seguida es la propuesta por (Contreras & Vehi, 2018), esta metodología se ilustra en el diagrama de la **Figura 16**.

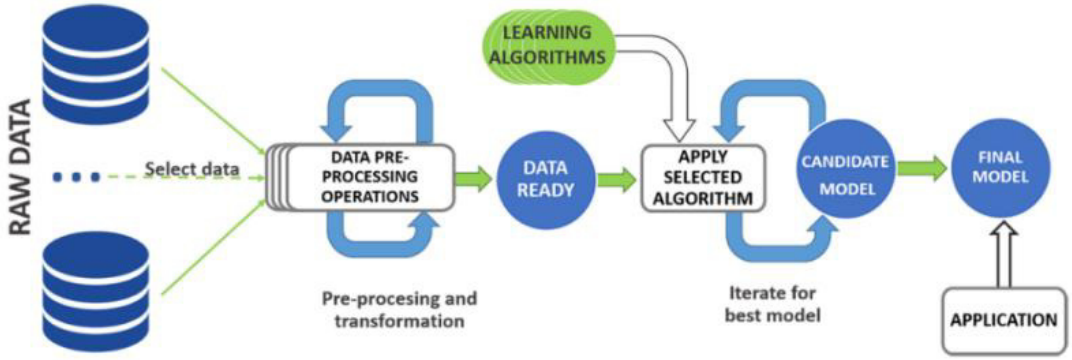

Figura 16. Diagrama general del proceso de un algoritmo de aprendizaje (Contreras & Vehi, 2018)

Ahora se define cada una de las etapas en el diagrama.

Primera etapa: adquisición y preprocesamiento de los datos

La primera etapa consiste en, primeramente, obtener los datos. En todos los experimentos que se presentan a continuación los datos usados están disponibles de manera pública.

Para el primer experimento, los datos que se usan datos artificiales, estos son generados usando la librería *Scikit-learn*, la cual provee el módulo datasets, este módulo incluye métodos para poder generar datasets artificiales con distintas formas y número de instancias en el dataset. En la **Figura 14** se muestran los datasets artificiales generados para desarrollar el primer experimento.

De los datasets empleados, uno de estos es linealmente separable y los dos restantes son no linealmente separables, es decir, no se pueden separar las dos clases usando una línea recta. De la **Figura 14** se observa que el dataset número 1 es el dataset linealmente separable, mientras que los datasets dos y tres no son linealmente separables. Cada uno de los datasets cuenta con una cantidad distinta de instancias, en la **Tabla 2** se muestra el número de instancias para cada dataset. A cada una de las instancias en el datasets se le asigna un color dependiendo de la clase a la que esa instancia pertenezca.

Dataset	Número de instancias
Dataset 1	500
Dataset 2	350
Dataset 3	10,000

Tabla 2. Número de instancias por dataset para el primer experimento

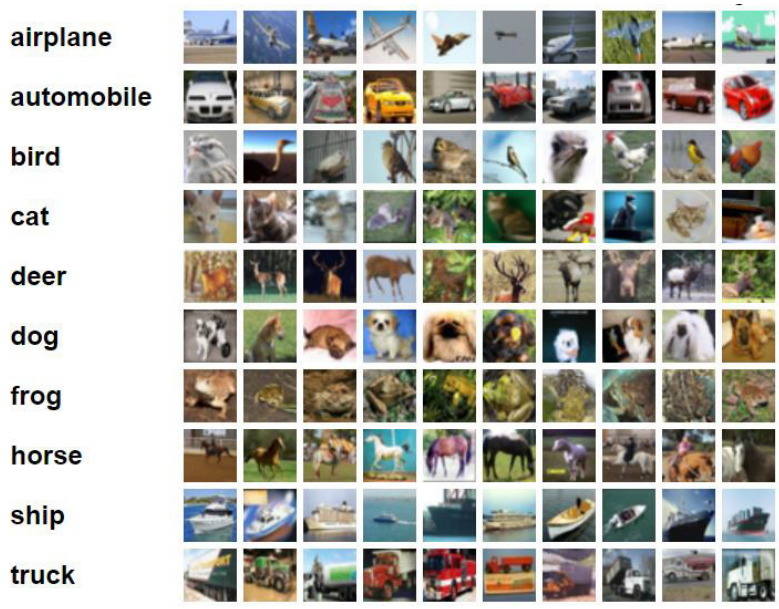

Figura 17. Muestra de imágenes del dataset CIFAR-10 (Krizhevsky & Hinton, 2009)

Para el segundo experimento los datos se obtuvieron del dataset CIFAR-10, el cual es un dataset que consiste en 60,000 imágenes a color de tamaño 32x32, categorizadas en 10 clases, las imágenes fueron recopiladas por (Krizhevsky & Hinton, 2009). En la **Figura 17** se muestran algunas imágenes del dataset CIFAR-10 junto con sus etiquetas, aunque para la tarea de generación de imágenes las etiquetas de las imágenes no son relevantes.

Finalmente, para el tercer experimento se usaron imágenes de dos datasets, el primero es BSD68 (Martin, Fowlkes, Tal , & Malik, 2001), el dataset contiene 432 imágenes de entrenamiento y 68 imágenes de prueba, se agregaron imágenes un total de 4,744 imágenes del dataset Waterloo exploration database (Ma, y otros, 2017) a las imágenes de prueba del dataset BSD68, obteniendo un total de 5,176 imágenes de entrenamiento. En la **Figura 18** se muestran algunas imágenes del dataset usado para el tercer experimento.

Figura 18. Muestra de imágenes usadas para el tercer experimento

Aunque las imágenes de la **Figura 18** se muestran sin ruido, es importante mencionar que para realizar el tercer experimento se les añadió ruido a estas imágenes para poder entrenar el modelo para eliminación de ruido.

En la mayoría de los casos es necesario procesar los datos antes de poder pasarlos al algoritmo de aprendizaje, esto se conoce como preprocesamiento. El preprocesamiento incluye distintas transformaciones a los datos, como normalización de los datos, eliminar valores atípicos, extracción de características, entre otros. El preprocesamiento que se debe aplicar a los datos depende del tipo de datos con el que se trabajará.

En el caso del primer experimento no hace falta aplicar preprocesamiento a los datos, debido a que estos son generados de manera artificial usando la librería *Scikit-learn*, por lo que estos están en el formato adecuado para poder pasarlos al algoritmo de aprendizaje, además de que no cuentan con valores atípicos.

Para el segundo experimento, el preprocesamiento que se debe realizar es normalizar las intensidades de los pixeles en las imágenes de CIFAR-10. Los valores de las intensidades de los pixeles en las imágenes de CIFAR-10 toman valores entre 0 y 255, sin embargo, estos valores se deben normalizar para que tomen valores en el rango de -1 a 1. Está normalización se logra aplicando la siguiente transformación a la imagen original:

$$I_N = \frac{I - 127.5}{127.5}$$

Donde I_N es la imagen normalizada e I es la imagen original.

Esta normalización se realiza debido a que cuando las intensidades de pixeles toman un rango amplio de valores, como el rango de 0 a 255 que las imágenes comúnmente tienen, los cálculos que deben realizar los modelos de Machine Learning se vuelven más complejos y costosos computacionalmente, por lo que, para evitar esto, se normalizan las intensidades de las imágenes para que estas tomen un rango de valores más reducido, como es el caso del rango de -1 a 1, esto también ayuda a que el modelo pueda converger más rápido en el entrenamiento. Aunque también sería posible normalizar las intensidades de los pixeles para que estén en el rango de 0 a 1, en este caso se elige el rango de -1 a 1 debido a que los valores de salida del modelo generador toman valores en el rango de -1 a 1, y este rango de intensidades debe coincidir con el rango de intensidades de las imágenes de entrenamiento.

Para el tercer experimento, el primer preprocesamiento consiste en transformar las imágenes del dataset de entrenamiento a escala de grises, ya que estas se encuentran a color y el modelo desarrollado trabaja con imágenes con un solo canal de color. Para convertir las imágenes se usó el método *cvtColor* de la librería *OpenCV*. Luego, se aplicó la técnica de data augmentation (técnica para aumentar la cantidad de datos disponibles aplicando transformaciones a los datos disponibles, estas transformaciones pueden ser rotaciones, zoom, oscurecer imágenes, etc. Data augmentation es útil cuando no se pueden obtener más datos.), para poder tener una mayor cantidad de imágenes de entrenamiento, mejorando así el desempeño del modelo. Las transformaciones que se aplicaron a los datos fueron las siguientes, de acuerdo con (Furuta, Inoue, & Yamasaki, 2019):

- Recortes aleatorios de tamaño 70x70
- Rotaciones aleatorias
- Flips izquierda/derecha

Los recortes aleatorios consisten en recortar una sección de tamaño 70x70 de la imagen original. Las rotaciones aleatorias consisten en rotar las imágenes por un ángulo aleatorio entre $0°$ y $\pm 100°$, y los flips izquierda/derecha consisten en invertir los valores de cada fila en la dirección izquierda/derecha, esto es, espejear las filas de manera que estas aparezcan

en el orden inverso. En la **Figura 19** se muestran estas transformaciones aplicadas a una imagen del dataset usado para el tercer experimento.

Figura 19. Transformaciones aplicadas para aumento de datos

Como último paso de preprocesamiento, al igual que para el experimento número dos, se normalizaron las intensidades de los pixeles, pero en este caso la normalización se realizó para que las intensidades de los pixeles estuvieran en el rango de 0 a 1, en vez de en el rango de 0 a 255 que originalmente tenían.

Aunque las imágenes usadas en el tercer experimento tienen tamaños variables, y comúnmente al usar un modelo de Deep Learning se debe ajustar el tamaño de las imágenes para que estas tengan un tamaño fijo debido a que el tamaño de la entrada de las DNN es fijo, en este caso no es necesario realizar esto, esta es una de las ventajas del modelo propuesto por (Furuta, Inoue, & Yamasaki, 2019), las imágenes pueden tener tamaños variables debido a que el número de agentes a entrenar se ajusta al número de pixeles en la imagen.

Segunda etapa: selección del algoritmo de aprendizaje

La segunda etapa es un proceso iterativo, pues se deben proponer varios algoritmos de aprendizaje que puedan resolver la tarea, posteriormente se debe aplicar cada uno de estos algoritmos candidatos a los datos de entrenamiento y evaluar su desempeño, luego de haber evaluado los algoritmos propuestos para la tarea se elige aquel algoritmo que haya tenido el mejor desempeño sobre los datos de entrenamiento.

Para poder elegir un modelo adecuado para la tarea que se va a resolver se debe considerar el tipo de datos con los que se trabajará, la dimensionalidad del espacio de las variables de entrada para el modelo, además del tipo de tarea que se va a resolver: clasificación, regresión, clustering etc.

Para el primer experimento se trabajará con datos numéricos de valor real, que se encuentran en dos dimensiones, y se busca resolver una tarea de clasificación binaria, es decir, se busca clasificar los datos en dos clases. Para esto se prueban dos implementaciones de la

arquitectura del perceptrón, la primera es una implementación propia desde cero, y la segunda es usando la implementación de la librería *Scikit-learn*. Ambas implementaciones se evaluarán y compararán una con otra para poder determinar la que mejor se adecua a la tarea en cuestión.

Para el segundo experimento, se busca resolver una tarea de aprendizaje no supervisado, usando un modelo generativo, para resolver esta tarea se empleó una GAN con capas convolucionales, esta arquitectura de GAN se conoce como GAN convolucional profunda (DCGAN).

Para el tercer experimento, entrenó el modelo propuesto por (Furuta, Inoue, & Yamasaki, 2019) con tres configuraciones de ruido. Para la primera configuración, el modelo fue entrenado con ruido gaussiano (ruido proveniente de una distribución gaussiana) de media 0 y varianza 15. La segunda configuración consiste en entrenar el modelo con ruido gaussiano de media 0 y varianza 25. Finalmente, para la tercera configuración el modelo fue entrenado con ruido de media 0 y varianza 50.

Tercera etapa: evaluación del modelo

La tercera etapa del proceso general de un algoritmo de Machine Learning consiste en, una vez construidos y entrenados los modelos de Machine Learning, se deben evaluar estos modelos usando datos "desconocidos" para el modelo, es decir, datos que no hayan sido usados durante la etapa de entrenamiento del modelo, esto con el propósito de evitar sesgos en la evaluación de los modelos. Para lograr esto, el conjunto de datos disponible se debe dividir en dos subsets, como se muestra en la **Figura 20**, el primero es el conjunto de datos de entrenamiento, usado solamente durante la etapa de entrenamiento del modelo. El segundo subset corresponde al conjunto de datos de prueba, los cuales deben ser separados de los datos de entrenamiento y estos solo se le pasan al modelo para evaluación cuando haya finalizado la etapa de entrenamiento, como se menciona previamente, con el fin de evitar sesgos en la evaluación.

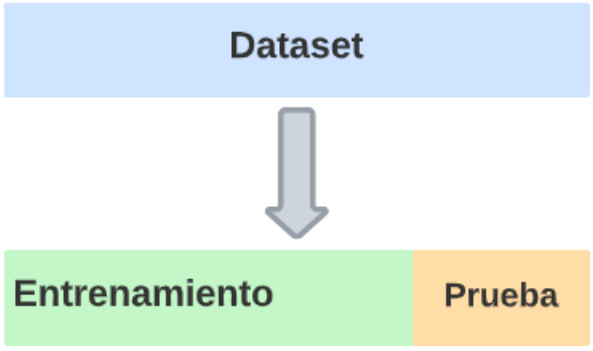

Figura 20. División de los datos en entrenamiento y prueba

Los algoritmos de aprendizaje empleados se evalúan sobre los datos de prueba usando una de las métricas discutidas en la sección de **Marco Teórico**, la métrica que se usará depende de la tarea a resolver en cada experimento.

Para el primer experimento, primero se dividió el datasets en los dos subsets de entrenamiento y prueba. Se designó el 90% del dataset para el subset de entrenamiento y el 10% restante para el subset de prueba. Como esta es una tarea de aprendizaje supervisado, específicamente clasificación, la métrica con la que se evalúan los modelos implementados es la de **Accuracy**.

Para el segundo experimento no es necesario realizar la división del dataset en entrenamiento y prueba, pues las GAN no tienen una fase de prueba como las de otros algoritmos. Esto debido a que las GAN son evaluadas con la calidad de las imágenes que generan. Debido a que las GAN se evalúan con la calidad de las imágenes generadas, no es tan sencillo realizar esta evaluación, por esto, para este experimento el modelo solo se evalúa de manera cualitativa mostrando algunas imágenes generadas por la DCGAN. Adicionalmente, se muestran también las funciones de pérdida (función que calcula la distancia entre la salida obtenida por el modelo y la salida esperada) de la DCGAN.

Para el tercer experimento, se cuenta con un total de 5,244 imágenes de las cuales 5,176 se usan para entrenar el modelo y las 68 imágenes restantes se usan para evaluar el modelo. En este caso la tarea que se busca resolver es la de eliminación de ruido, por lo que la métrica que se usará para evaluar los modelos implementados es la de **Proporción máxima de señal a ruido** (PSNR).

Resultados experimentales

A continuación, se presentan los resultados obtenidos para los tres experimentos descritos en la sección de **Metodología**. Como se discutió antes, para cada uno de los experimentos se usan métricas distintas para su evaluación, esto debido a que cada uno de los experimentos resuelve una tarea distinta.

Primer experimento: perceptrón aplicado a datasets artificiales

Para el primer experimento que consiste en la clasificación binaria de los datasets artificiales mostrados en la **Figura 14** ¡Error! No se encuentra el origen de la referencia.haciendo uso del perceptrón, teniendo en cuenta que el perceptrón es un modelo lineal, por lo que su frontera de decisión será una línea recta, se espera que se obtengan buenos resultados para el dataset 1, debido a que este es linealmente separable. Sin embargo, no se espera que el perceptrón tenga un buen desempeño con los datasets 2 y 3, ya que estos no son linealmente separables, por lo que la frontera de decisión lineal del perceptrón no logrará separar las dos

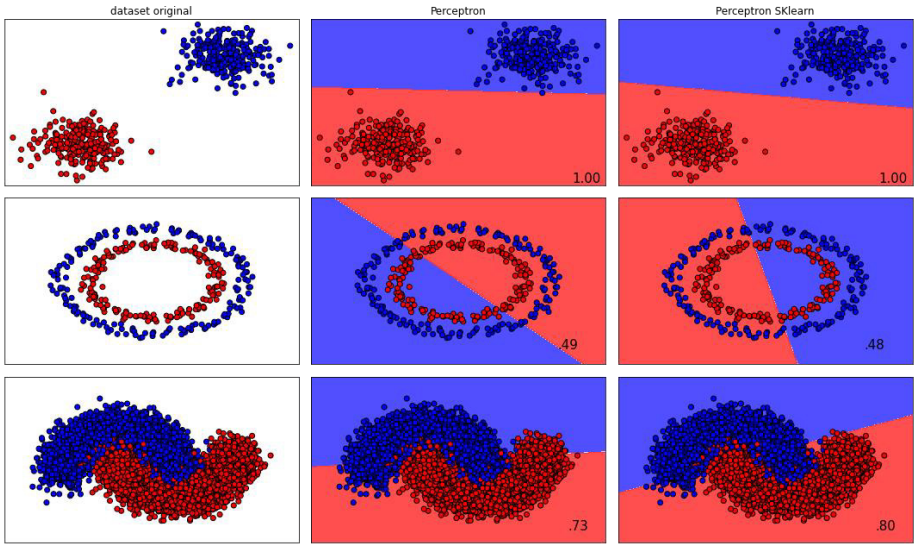

Figura 21. Resultados de la clasificación con perceptrón

clases adecuadamente.

Como se mencionó en la sección de **Metodología**, se compararán dos implementaciones del perceptrón, una implementación propia hecha desde cero y la implementación de la librería *Scikit-learn*. En la **Figura 21** se observan los resultados de la clasificación en una cuadricula. La primera columna de la cuadricula corresponde a los datasets originales, la segunda columna corresponde a la clasificación realizada por la implementación propia del perceptrón, y la última columna corresponde a la clasificación realizada por el perceptrón implementado con la librería de *Scikit-learn*. Para cada una de las imágenes de las columnas dos y tres, se

grafica el dataset asignando el color a cada punto de acuerdo con la clase a la que pertenece el punto, además, se grafica la frontera de decisión del clasificador, la cual indica la clase a la que pertenecería un punto situado en una posición dada, finalmente, en la esquina inferior derecha de cada imagen en las columnas 2 y 3 se muestra el **Accuracy** de cada uno de los clasificadores para cada dataset. En adición a esto, en la **Tabla 3** se muestran los accuracies obtenidos por cada clasificador para cada dataset.

Dataset	Accuracy perceptrón	Accuracy perceptrón Scikit-learn
Dataset 1	1.00	1.00
Dataset 2	0.49	0.48
Dataset 3	0.73	0.80

Tabla 3. Accuracies de los clasificadores para cada dataset

Los resultados cumplen con lo esperado, ya que de la **Figura 21** y la **Tabla 3** se observa que los perceptrones de ambas implementaciones tienen un muy buen desempeño para el dataset 1, alcanzando un accuracy de 1.0, mientras que para los datasets 2 y 3 el desempeño de ambas implementaciones del perceptrón es menor, llegando a accuracies de 0.49 para la implementación propia del perceptrón y 0.48 para la implementación de *Scikit-learn*, pues los datasets no son linealmente separables. El dataset número dos fue con el que se obtuvieron los peores resultados para ambos perceptrones, en el dataset 3 la implementación propia del perceptrón tuvo un accuracy de 0.73, mientras que la implementación de *Scikit*-learn tuvo un accuracy de 0.80. Nuevamente, acorde con lo esperado, las fronteras de decisión que se observan en la **Figura 21** son lineales, es decir, las dos clases son separadas por una línea recta. Con estos resultados se puede concluir que el perceptrón es un modelo lineal, ya que crea una frontera de decisión lineal, por lo que solo tendrá un buen desempeño al clasificar datasets que son linealmente separables.

Segundo experimento: CNN y GAN para generación de imágenes del dataset CIFAR-10

En el segundo experimento, en el cual se busca generar imágenes del dataset CIFAR-10 (**Figura 17**) mediante una GAN con capas convolucionales (DCGAN), se espera que las imágenes generadas por esta arquitectura puedan lograr una similitud considerable con las del dataset CIFAR-10, esto debido a que se incluyen capas convolucionales a la arquitectura de la GAN, estas capas convolucionales ayudan en el procesamiento de las imágenes de entrada. En la **Figura 22** se observan algunas de las imágenes generadas por esta arquitectura implementada de DCGAN, las imágenes se generaron después de entrenar el modelo por 1, 20, 40, 60, 80 y 100 épocas, respectivamente.

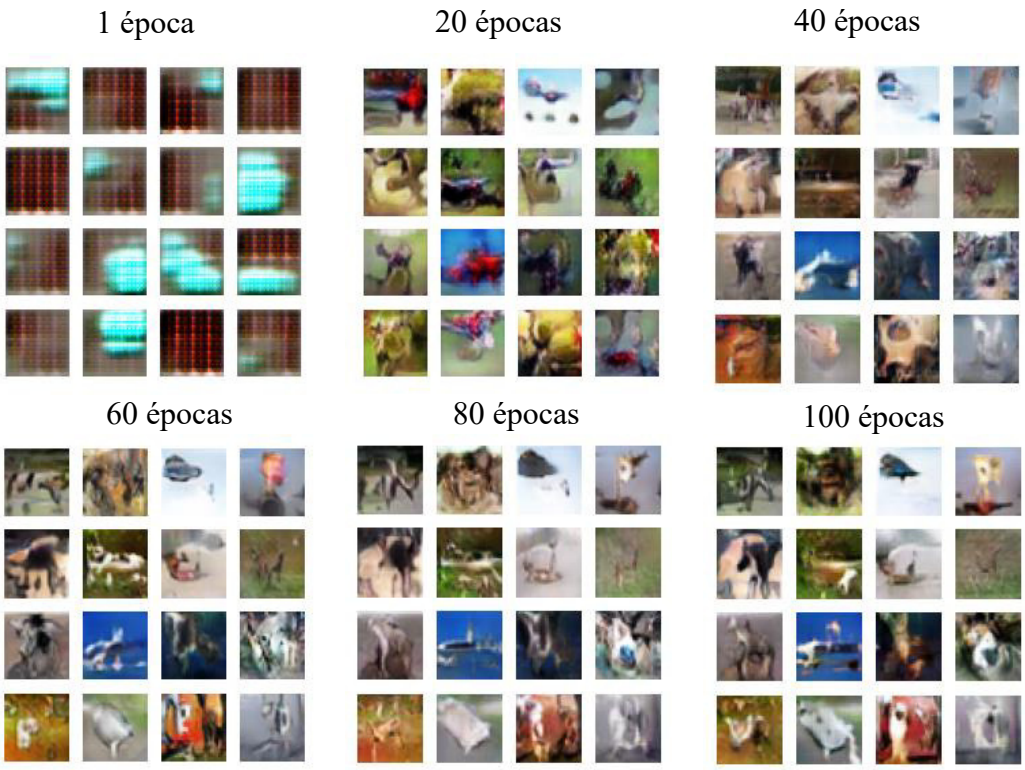

Figura 22. Imágenes generadas usando DCGAN

En la **Figura 22** se observa que las imágenes no tienen similitud con las del dataset CIFAR-10, en estas imágenes solo se muestran algunas manchas azules y una cuadricula. Sin embargo, según van avanzado las épocas, estas manchas azules presentes las imágenes generadas en la época 1 desaparecen y las imágenes generadas parecen tener cierta similitud con las del dataset CIFAR-10. Aunque al llegar a la época 100 en las imágenes generadas no se observa claramente algún objeto de los presentes en el dataset CIFAR-10, como caballos, perros, coches, etc., las imágenes parecen tener presentes objetos de estas clases, pero con una cierta deformación, por lo que estos resultados se podrían considerar como aceptables.

Una razón por la que las imágenes generadas no tienen la forma exacta de los objetos presentes en el dataset CIFAR-10 podría ser que las imágenes de este dataset están divididas en 10 clases, por lo que la GAN no se centra en generar imágenes de un solo tipo, como imágenes de caballos, sino que tiene que crear imágenes de 10 clases.

Otra razón es que las GAN son muy inestables, ya que las GAN están compuestas por dos redes neuronales que compiten entre sí. Esta inestabilidad se puede ver en la **Figura 23** donde se grafica la pérdida del discriminador (derecha) y del generador (izquierda) de la DCGAN implementada.

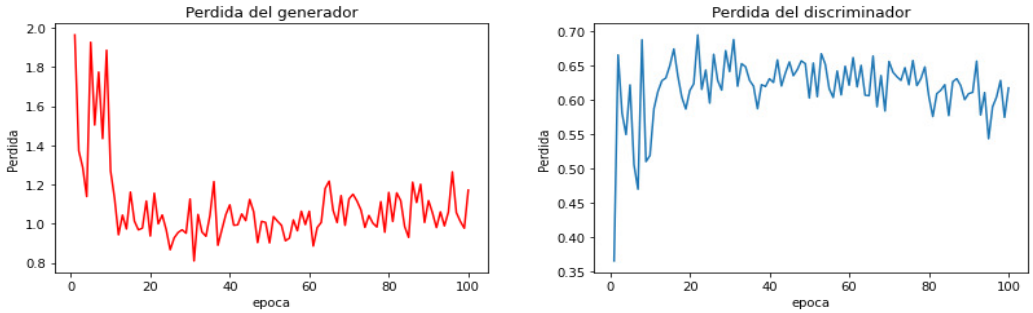

Figura 23. Pérdidas del discriminador (derecha) y del generador (izquierda) de la DCGAN

En la **Figura 23** se observa que las pérdidas de ambos componentes de la GAN presentan una oscilación, lo que exhibe la inestabilidad comentada antes. A pesar de esta oscilación, lo que se observa en las gráficas concuerda con lo esperado, pues la pérdida del generador disminuye, lo que indica que las imágenes generadas por esta red están siendo clasificadas como reales por la red discriminadora, mientras que la pérdida de la red discriminadora aumenta, lo que indica que esta red está clasificando imágenes provenientes del generador como reales.

Tercer experimento: Réplica de artículo científico para para eliminación de ruido de imágenes usando Redes Neuronales Convolucionales A3C

Finalmente, para el tercer experimento se entrenó el modelo para eliminación de ruido en imágenes propuesto por (Furuta, Inoue, & Yamasaki, 2019), se entrenaron tres modelos. Cada uno de estos tres modelos fue entrenado con ruido gaussiano, donde la desviación estándar σ_t (el subíndice t indica que esta desviación estándar es para las imágenes de entrenamiento) de la distribución gaussiana que generaba el ruido para entrenar las imágenes de cada modelo era distinta. El primer modelo se entrenó con imágenes con ruido gaussiano de $\sigma_t = 15$, el segundo con $\sigma_t = 25$ y el tercero con $\sigma_t = 50$, la media de las distribuciones gaussianas para los tres modelos fue de 0. Una vez entrenados los modelos, cada uno fue probado con ruido de desviación estándar $\sigma_p = 15, 25, 50$ (el subíndice p indica que esta desviación estándar es para las imágenes de prueba). A continuación, se muestran los

resultados obtenidos con cada uno de los modelos al ser entrenados con el ruido gaussiano de las desviaciones estándar σ_t mencionadas antes.

Primero se muestran los resultados obtenidos con cada uno de los modelos al pasarles imágenes que contienen ruido gaussiano de $\sigma_p = 15$, esto se observa en la **Figura 24**.

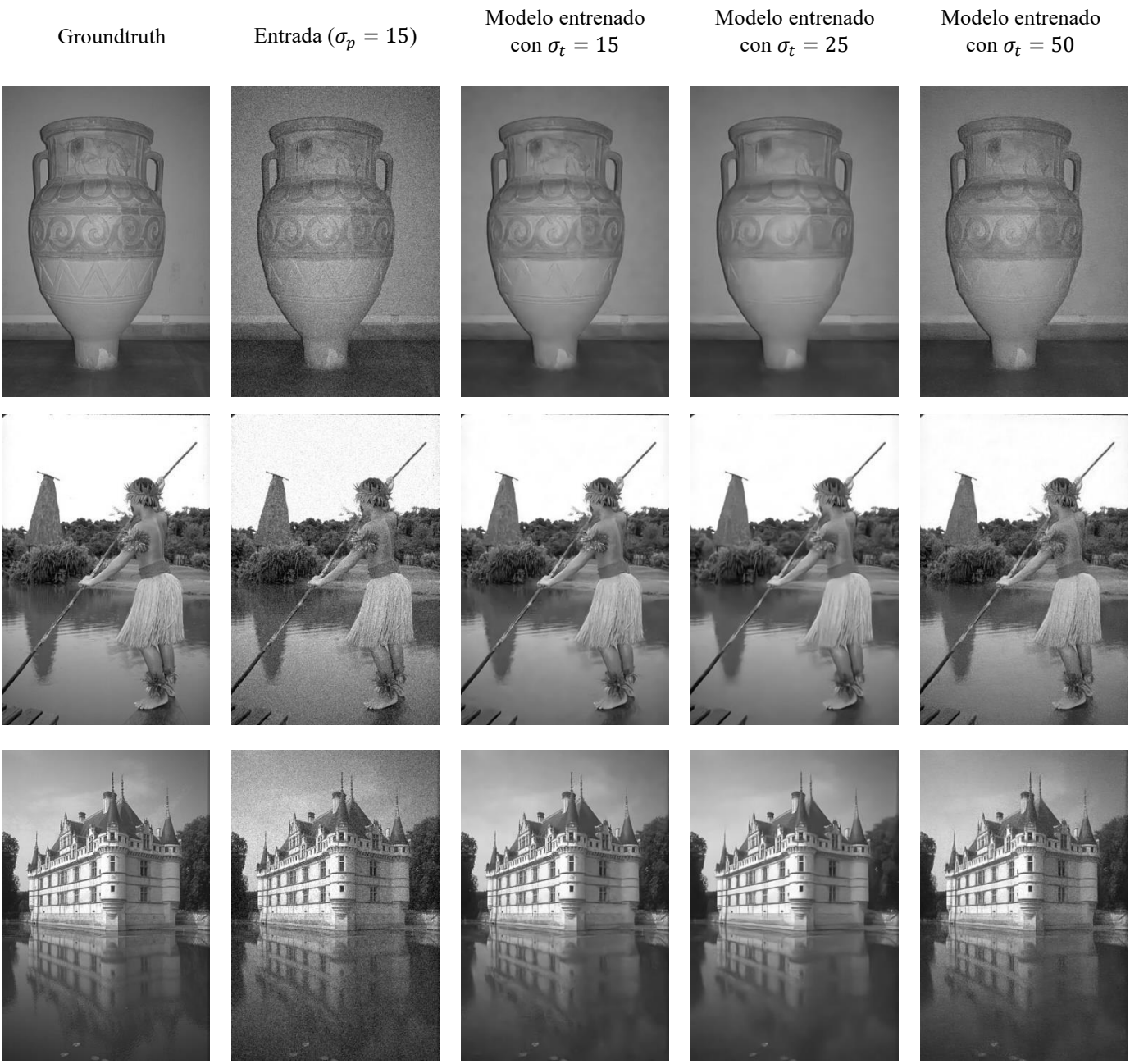

Figura 24. Resultados para imágenes con ruido de $\sigma_p = 15$

En la primera columna de la **Figura 24** se muestra el groundtruth de cada una de las imágenes que se muestra. En la segunda columna se muestra la imagen que los modelos reciben como entrada, estas son imágenes con ruido gaussiano de $\sigma_p = 15$. En las columnas 3, 4 y 5, se

muestran las imágenes después de que los modelos entrenados con imágenes con ruido gaussiano de $\sigma_t = 15, 25$ y 50, respectivamente, le eliminaran el ruido a la imagen de entrada. Para los siguientes casos (imágenes de prueba con ruido gaussiano de $\sigma_p = 25,\ 50$) los resultados se muestran en el mismo formato que en la **Figura 24**.

De la **Figura 24** se observa que los tres modelos fueron capaces de eliminar satisfactoriamente el ruido gaussiano de desviación estándar $\sigma_p = 15$ de las imágenes de prueba. Sin embargo, en el caso del modelo entrenado con ruido gaussiano de $\sigma_t = 50$, las imágenes aún tienen un poco de ruido, en el caso del modelo entrenado con ruido gaussiano de $\sigma_t = 25$, algunas imágenes lucen un poco suavizadas, esto se observa en la imagen del jarrón de la **Figura 24**, donde se pierden algunos detalles de este debido al suavizado, por otra parte, el modelo entrenado con $\sigma_t = 15$ logró eliminar el ruido de las imágenes y además en estas imágenes no se nota el suavizado que se aprecia en las imágenes tratadas con el modelo de $\sigma_t = 25$.

Ahora se muestran los resultados obtenidos con cada uno de los modelos al pasarles imágenes que contienen ruido gaussiano de $\sigma_p = 25$, esto se muestra en la **Figura 25**.

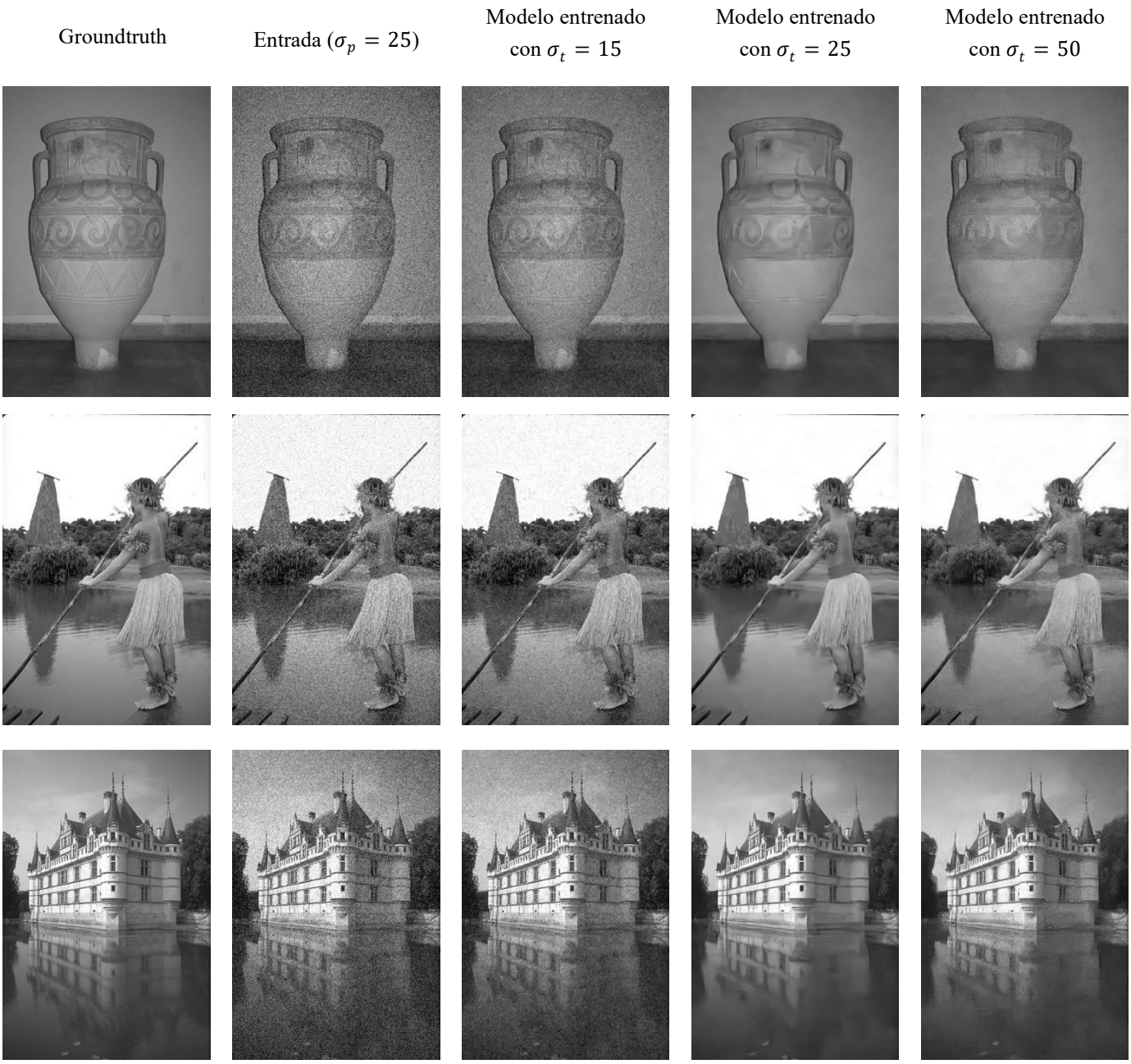

Figura 25. Resultados para imágenes con ruido de $\sigma_p = 25$

En la **Figura 25** se observa que para el caso de las imágenes con ruido gaussiano de $\sigma_p = 25$, no todos los modelos lograron eliminar satisfactoriamente, como sucedía con las imágenes de ruido gaussiano de $\sigma_p = 15$. En este caso, para el modelo entrenado con imágenes con ruido gaussiano de $\sigma_t = 15$, no se obtienen buenos resultados, pues la disminución de ruido no es muy notoria. Para el modelo entrenado con ruido gaussiano de $\sigma_t = 25$, este logra eliminar el ruido presente en las imágenes, sin embargo, aún se puede notar el suavizado que también se aprecia para este modelo en la **Figura 24**. Por último, el modelo entrenado con

ruido gaussiano de $\sigma_t = 50$ logra eliminar la mayor parte del ruido, aunque en algunas imágenes se puede apreciar que pierden detalle, como es el caso de la imagen del jarrón.

Finalmente, en la **Figura 26** se muestran los resultados obtenidos con cada uno de los modelos al pasarles imágenes con ruido gaussiano de $\sigma_p = 50$.

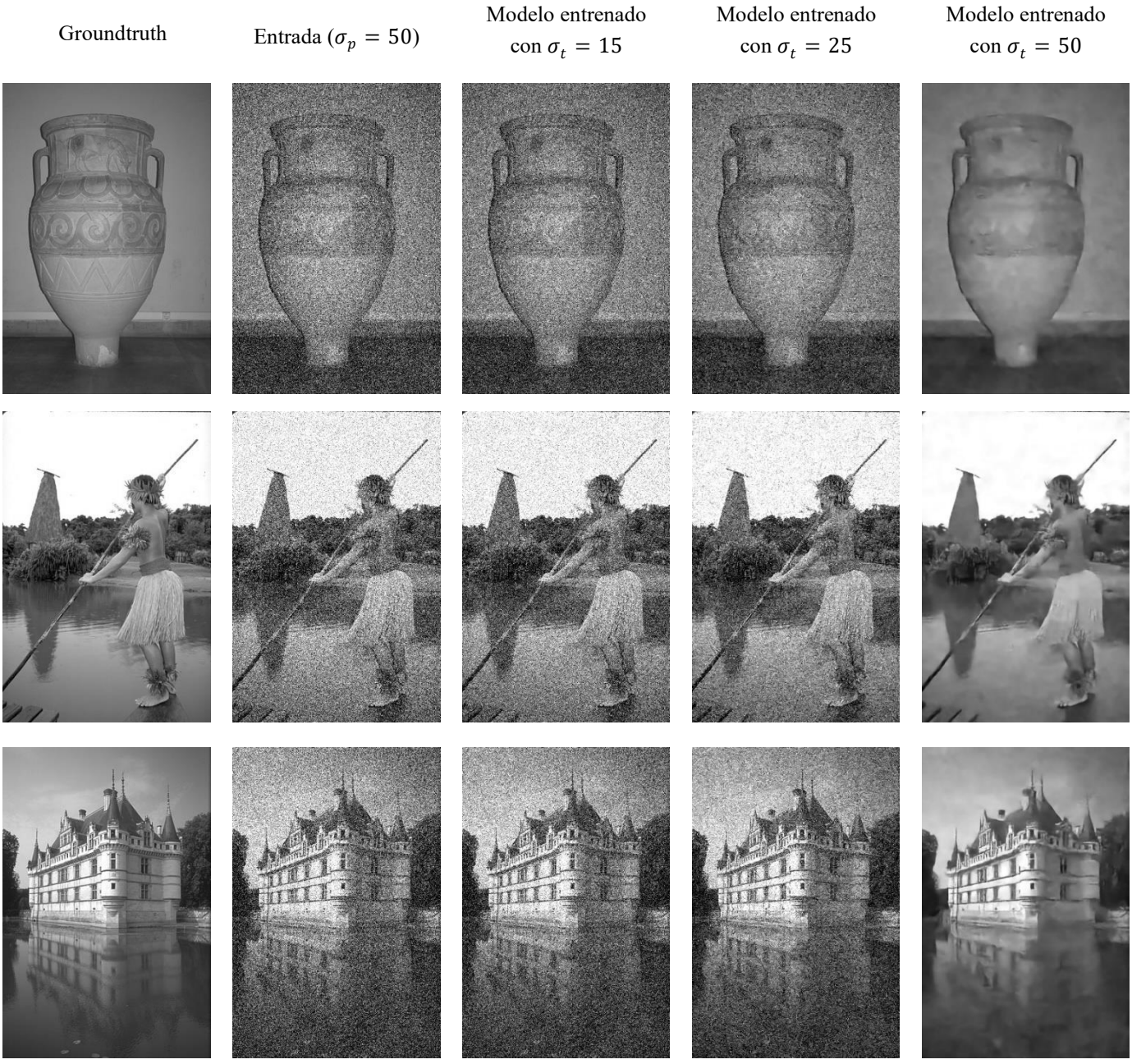

Figura 26. Resultados para imágenes con ruido de $\sigma_p = 50$

La **Figura 26** muestra que para el caso de las imágenes con ruido gaussiano de $\sigma_p = 50$ solo un modelo fue capaz de eliminar el ruido presente en las imágenes. Los modelos entrenados con ruido gaussiano de $\sigma_t = 15$ y 25 lograron una muy poca, o incluso nula, supresión del

ruido gaussiano de $\sigma_p = 50$, por otra parte, el modelo entrenado con ruido gaussiano de $\sigma_t = 50$ fue el único capaz de eliminar este ruido, aunque este modelo fue capaz de eliminar el ruido, las imágenes no fueron restauradas perfectamente, ya que en la **Figura 26** se aprecia que las imágenes tratadas por el modelo entrenado con ruido gaussiano de $\sigma_t = 50$ lucen suavizadas.

Para complementar estos resultados, la **Tabla 4** muestra el PSNR promedio sobre todas las imágenes en el set de prueba obtenido con cada uno de los modelos para las imágenes con ruido gaussiano de $\sigma_p = 15, 25$ y 50.

σ_t del ruido de las imágenes de entrenamiento	σ_p de las imágenes de prueba		
	$\sigma_p = 15$	$\sigma_p = 25$	$\sigma_p = 50$
$\sigma_t = 15$	**31.36**	24.07	15.64
$\sigma_t = 25$	29.85	**28.76**	17.60
$\sigma_t = 50$	28.97	27.17	**25.74**
Original	24.79	20.47	14.90

Tabla 4. PSNR en dB del set de prueba con ruido gaussiano

En la **Tabla 4**, "Original" es el PSNR entre la entrada del modelo y el groundtruth. La primera columna indica la desviación estándar del ruido presente en las imágenes con las que se entrenó cada modelo. Las columnas 2, 3 y 4 indican el PSNR obtenido al probar cada modelo con imágenes que cuentan con ruido gaussiano de $\sigma_p = 15, 25$ y 50, respectivamente. Es importante recordar que un PSNR mayor indica un mejor desempeño del modelo.

En la **Tabla 4** se corrobora lo observado en las figuras anteriores, pues de la columna de las imágenes de prueba con ruido gaussiano de $\sigma_p = 15$ se observa que el modelo que obtuvo el mayor PSNR fue el modelo entrenado con $\sigma_t = 15$ con un PSNR de 31.36, el modelo que obtuvo el menor desempeño fue el entrenado con $\sigma_t = 50$ con un PSNR de 28.97, sin embargo, el desempeño solo fue un poco menor que el modelo entrenado con $\sigma_t = 25$ cuyo PSNR fue de 29.85. Para las imágenes con ruido de $\sigma_p = 25$, el modelo que obtuvo el mejor desempeño fue el modelo entrenado con $\sigma_t = 25$, el cual obtuvo un PSNR de 28.76, así como para las imágenes de $\sigma_p = 15$, el modelo entrenado con $\sigma_t = 50$ obtuvo un desempeño un poco menor, PSNR de 27.17, que el modelo entrenado con $\sigma_t = 25$, el modelo que obtuvo el peor desempeño fue el entrenado con $\sigma_t = 25$ con un PSNR de 24.07. Finalmente, para las imágenes con $\sigma_p = 50$, los modelos con $\sigma_t = 15$ y 25 no obtuvieron un buen desempeño pues solo pudieron mejorar el PSNR original en 0.74 y 2.7 dB respectivamente, para este caso el mejor modelo fue el entrenado con $\sigma_t = 50$, obteniendo un PSNR de 25.74.

Con base en los resultados anteriores se puede concluir que los modelos solo podrán eliminar ruidos con desviaciones estándar menores o iguales a las de las distribuciones gaussianas del ruido con el que fueron entrenados, además para obtener el mejor desempeño en la eliminación del ruido es necesario que el modelo sea entrenado con ruido gaussiano de la misma desviación estándar que el ruido gaussiano de las imágenes de las que eliminará el ruido.

Ahora se muestra el desempeño del modelo entrenado con $\sigma_t = 50$ al eliminar ruido de imágenes recopiladas de internet de las cuales se desconoce el ruido que las contamina. Se elige el modelo entrenado con $\sigma_t = 50$ debido a lo antes mencionado, debido a que se desconoce el ruido que contamina las siguientes imágenes y de los resultados anteriores se observa que en general este modelo tiene un desempeño mejor que los otros modelos, pues si el ruido gaussiano tiene una desviación estándar mayor al ruido con el que fueron entrenados los modelos, estos serán incapaces de eliminar el ruido. Por esta razón es preferible usar el modelo entrenado con $\sigma_t = 50$. En la **Figura 27** se muestran estas imágenes contaminadas con ruido desconocido, así como la imagen obtenida después de que esta es procesada por el modelo.

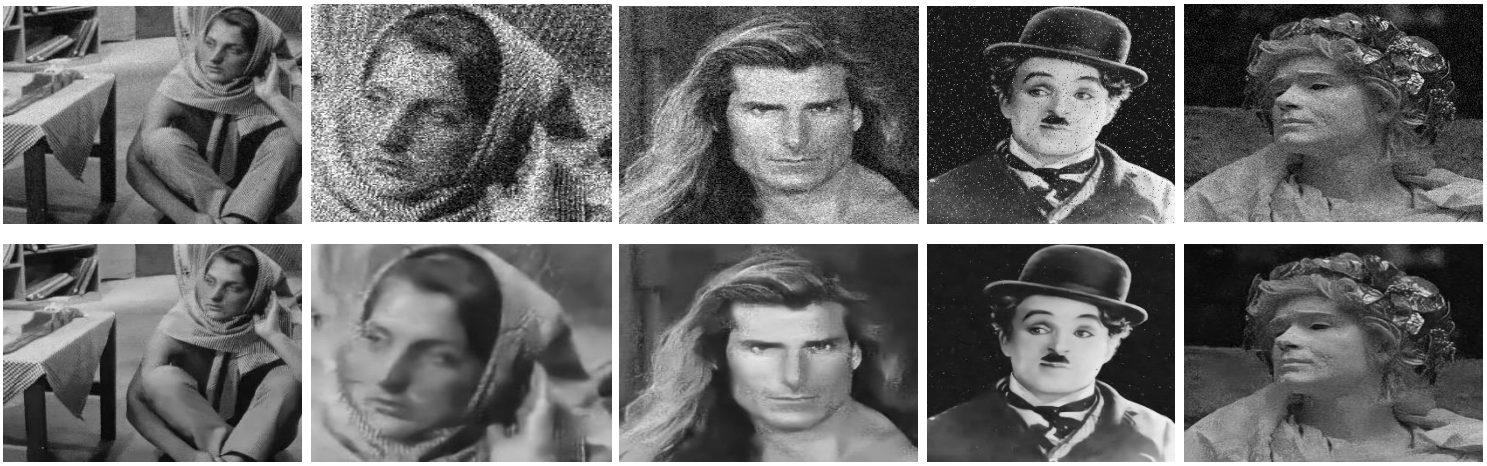

Figura 27. Resultados para imágenes con ruido desconocido

En la fila superior de la **Figura 27** se muestran las imágenes con ruido, mientras que en la segunda fila de la **Figura 27** se muestran las imágenes obtenidas al pasar las imágenes con ruido por el modelo entrenado con $\sigma_t = 50$. De ahora en adelante al hablar del "modelo" se hará referencia al modelo entrenado con $\sigma_t = 50$

Los resultados que se obtienen al pasar imágenes con ruido desconocido por el modelo son aceptables, pues de la **Figura 27** se observa que para todas las imágenes el modelo logró eliminar o reducir considerablemente el ruido, a pesar de esto, algunas imágenes como la del rostro de la persona en la segunda columna de la **Figura 27** lucen suavizadas, esto sucede principalmente con las imágenes que tienen una alta densidad de ruido. Para el caso de la imagen del hombre con sombrero en la cuarta columna de la **Figura 27**, la cual parece estar contaminada con ruido sal y pimienta (ruido que convierte pixeles en negro o blanco aleatoriamente), aunque después de tratar la imagen se pueden percibir algunos pixeles en blanco, el modelo obtiene resultados aceptables incluso ante este tipo de ruido el cual es más complicado de eliminar debido a que al convertir un pixel en negro o blanco aleatoriamente, se pierde la información del pixel original.

Para mostrar las ventajas de este modelo, se compara el modelo con métodos clásicos de eliminación de ruido, como lo son los filtros gaussianos. Para esto se usan las imágenes de la **Figura 27** donde el ruido es más notorio, a la imagen de la mujer en la segunda columna de

la **Figura 27** se le llamará imagen 1, a la imagen del hombre con cabello largo de la columna 3 de la **Figura 27** se le llamará imagen 2, y a la imagen del hombre con sombrero de la columna 4 de la **Figura 27** se le llamará imagen 3. A cada una de estas imágenes se les aplicaron filtros gaussianos con desviación estándar $\sigma = 5$ y tamaño de filtro 5x5, 7x7, 9x9 y 11x11. En la **Tabla 5** se muestran los PSNR obtenidos para cada imagen con cada filtro, además del PSNR obtenido usando el modelo replicado.

	Filtro 5x5	**Filtro 7x7**	**Filtro 9x9**	**Filtro 11x11**	**Modelo replicado**
Imagen 1	15.33	15.18	15.06	14.96	15.20
Imagen 2	19.69	19.42	19.22	19.07	19.33
Imagen 3	17.48	17.29	17.15	17.02	17.23

Tabla 5. PSNR métodos clásicos vs modelo replicado

En la **Figura 28** se muestra cada una de las imágenes 1, 2 y 3 con ruido, además de la imagen filtrada que obtuvo el mejor y el peor PSNR usando los filtros antes descritos, de igual manera, se muestra la imagen obtenida con el modelo replicado.

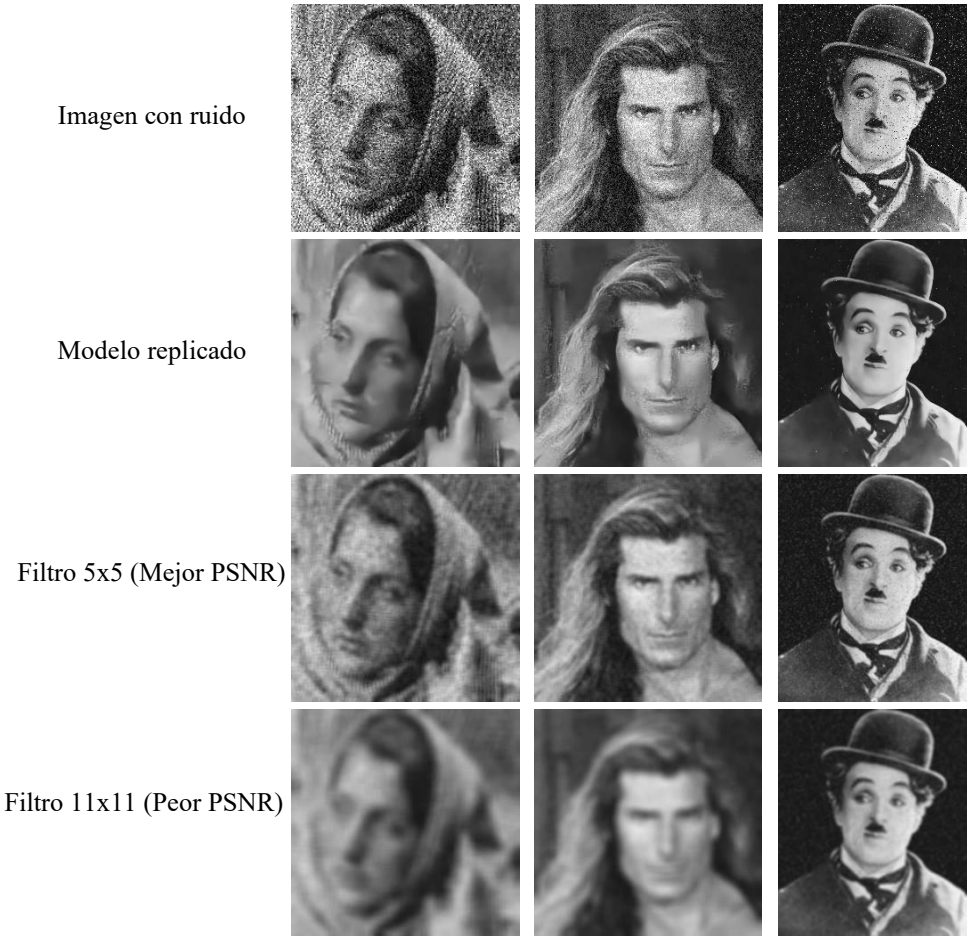

Figura 28. Métodos clásicos vs modelo replicado

De la **Tabla 5** y la **Figura 28** se puede notar que aunque al usar los métodos clásicos se obtienen PSNR similares a los obtenidos con el modelo replicado, e incluso algunas configuraciones de los filtros superan al modelo replicado, las imágenes que se obtiene después del procedimiento de eliminación de ruido son distintas, las imágenes obtenidas mediante métodos clásicos pierden aún más detalle que las obtenidas con el modelo, esto debido al suavizamiento del filtro gaussiano, además de que todavía se pueden observar algunas pequeñas manchas en estas imágenes obtenidas mediante métodos clásicos.

Ahora que se ha probado que el modelo es capaz de eliminar ruido de origen desconocido en imágenes, se probará este modelo con imágenes médicas, en particular, imágenes de piezas dentales. Estas imágenes son parte del *Medical Image Dataset*, disponible de manera pública en el sitio web *kaggle*. En la **Figura 29** se muestran algunas imágenes tratadas por el modelo, además de las imágenes tratadas con el modelo, también se muestran estas imágenes al ser limpiadas por un filtro gaussiano de 5x5 con $\sigma = 5$ para poder comparar los resultados del modelo con los resultados al usar técnicas clásicas de eliminación de ruido, se usa el filtro de 5x5 ya que fue con el que se obtuvieron los mejores resultados de acuerdo con la **Tabla 5**.

Figura 29. Resultados para imágenes médicas

De la **Figura 29** se observa que el modelo fue capaz de eliminar el ruido presente en estas imágenes, sin embargo, una desventaja es que algunas imágenes lucen suavizadas, por lo que es posible que se pierdan algunos detalles de las imágenes importantes que pudieran ayudar en el diagnostico de algún padecimiento. En el caso de las imágenes limpiadas mediante el filtro gaussiano de tamaño 5x5, en estas aún se puede notar ligeramente la presencia de ruido. Con esto, se puede concluir que este modelo replicado puede usarse para tratar imágenes médicas, no obstante, se debe tomar en cuenta la aplicación en la que se usará este método ya que podría ocurrir que al limpiar las imágenes con este método se pierda información crítica para realizar diagnósticos mediante las imágenes.

Conclusiones

Se logró replicar el artículo de técnicas basadas en aprendizaje por refuerzo para la eliminación de ruido en imágenes, los resultados obtenidos al probar estas técnicas con imágenes que no eran médicas fueron buenos y similares a los obtenidos por los autores del artículo, (Furuta, Inoue, & Yamasaki, 2019). De estos resultados se logró mostrar que el modelo entrenado es capaz de eliminar ruido en imágenes, aunque no se conozca el tipo de ruido que está contaminando a la imagen.

Aunque también se logró aplicar este modelo a imágenes médicas, los resultados para este tipo de imágenes no fueron tan buenos como se esperaba. Esto debido a que a pesar de que el modelo fue capaz de eliminar el ruido presente en estas imágenes, se obtenían imágenes suavizadas, por lo que se perdían algunos detalles, estos detalles perdidos podrían resultar de suma importancia para el personal médico, o incluso para otros modelos de Machine Learning, al momento de realizar un diagnóstico. No obstante, esto no quiere decir que el modelo no sea de utilidad, este modelo podría usarse en casos donde pequeños detalles en las imágenes no tomen gran importancia, sino que lo más importante sea evitar el ruido en las imágenes.

Una de las limitaciones de esta técnica es el espacio de acción limitado, el modelo solo puede elegir una de nueve acciones, por lo que puede ocurrir que estas acciones no sean de utilidad para suprimir algún tipo de ruido. Por lo que podría probarse añadiendo más tipos de filtros al espacio de acción para comprobar si esto permite eliminar una cantidad mayor de tipos de ruido. Incluso, se podría probar modificando parámetros de los filtros ya presentes en el espacio de acción, para verificar si esto logra hacer que el modelo pueda generar imágenes con menor pérdida de detalle.

Una de las dificultades principales que se presentaron al momento de entrenar los modelos fue la disponibilidad de hardware, debido al uso de Deep Learning, se requería GPU para poder entrenar estos modelos de manera más rápida por lo que se utilizó *Google Colaboratory* para realizar el entrenamiento, sin embargo, debido a los límites de uso de la versión gratuita de esta herramienta, los modelos solo podían ser entrenados por 12 horas.

Otra dificultad encontrada fue al momento de intentar probar el modelo entrenado con imágenes médicas, ya que hay muy poca disponibilidad de imágenes médicas que contengan ruido en la web. Pese a esto, se logró encontrar un dataset de imágenes médicas con ruido para poder evaluar el modelo, aunque estas imágenes no contenían una gran cantidad de ruido.

Referencias

Ahire, J. B. (3 de Abril de 2020). *Demystifying the XOR problem*. Obtenido de DEV Community: https://dev.to/jbahire/demystifying-the-xor-problem-1blk

Andersson, J., Nyholm, T., Ceberg, C., Almén, A., Bernhardt, P., Fransson, A., & Olsson, L. E. (2021). Artificial intelligence and the medical physics profession-A Swedish perspective. *Physica Medica*, 218-225.

Bedolla, E., Padierna, L. C., & Castañeda, R. (2021). Machine Learning for Condensed Matter Physics. *Journal of Physics: Condensed Matter*.

Beyer, T., Bailey, D. L., Birk, U. J., Buvat, I., Catana, C., Cheng, Z., . . . Moser, E. (2021). Medical Physics and Imaging-A Timely Perspective. *Frontiers in Physics*.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly Detection: A Survey. *ACM Computing Surveys*.

Chollet, F. (2018). *Deep Learning with Python.* Nueva York: Manning.

Contreras, I., & Vehi, J. (2018). Artificial intelligence for diabetes management and decision support: literature review. *Journal of medical Internet research*, e10775.

Cunningham, P., Cord, M., & Delany, S. J. (2008). *Machine Learning Techniques for Multimedia.* Berlin: Springer.

Furuta, R., Inoue, N., & Yamasaki, T. (2019). PixelRL: Fully convolutional network with reinforcement learning for image processing. *IEEE Transactions on Multimedia*, 1704-1719.

Géron, A. (2020). *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow.* O´Reilly.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* MIT Press.

Gu , J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., . . . Chen, T. (2018). Recent Advances in Convolutional Neural Networks. *Pattern Recognition*, 354-377.

Jeffares, A. (19 de Noviembre de 2019). *K-Means: A Complete Introduction*. Obtenido de Towards Data Science: https://towardsdatascience.com/k-means-a-complete-introduction-1702af9cd8c

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 237-285.

Keevil, S. F. (2012). Physics and medicine: a historical perspective. *The Lancet*, 1517-1524.

Krizhevsky, A., & Hinton, G. (2009). Learning Multiple Layers of Features from Tiny Images.

Krogh, A. (2008). What are artificial neural networks? *Nature biotechnology*, 195-197.

Kuipers, M., & Prasad, R. (2021). Journey of Artificial Intelligence. *Wireless Personal Communications*, 3275-3290.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep Learning. *Nature*, 436-444.

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., . . . Dollár, P. (2015). Microsoft COCO: Common Objects in Context. *arxiv.org*.

Ma, K., Duanmu, Z., Wu, Q., Wang, Z., Yong, H., Li, H., & Zhang, L. (2017). Waterloo exploration database: New challenges for image quality assessment models. *IEEE Transactions on Image Processing*, 1004-1016.

Madhulatha, T. S. (2012). An Overview on Clustering Methods. *IOSR Journal of Engineering*, 719-725.

Mahapatra, S. (21 de Marzo de 2018). *Why Deep Learning over Traditional Machine Learning?* Obtenido de Towards Data Science: https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063

Martin, D., Fowlkes, C., Tal , D., & Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. *Proceedings Eighth IEEE International Conference on Computer Vision*, 416-423.

Mitchell, T. M. (1997). *Machine Learning.* Nueva York: McGraw-Hill.

Murphy, K. P. (2012). *Machine Learning: a Probabilistic Perspective.* Cambridge: MIT Press.

Ravichandiran, S. (2020). *Deep Reinforcement Learning with Python.* Birmingham: Packt.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction.* Cambridge: MIT Press.

Van Der Maaten, L., Postma, E., & Van Den Herik, J. (2009). Dimensionality Reduction: A Comparative. *Tilburg University*.

Vázquez, F. (21 de Diciembre de 2017). *Deep Learning made easy with Deep Cognition*. Obtenido de Medium: https://becominghuman.ai/deep-learning-made-easy-with-deep-cognition-403fbe445351

UNIVERSIDAD DE GUANAJUATO

León, Gto., a 13 de enero de 2023
Asunto: **Revisión de Tesis**

DR. DAVID YVES GHISLAIN DELEPINE
DIRECTOR
DIVISIÓN DE CIENCIAS E INGENIERIAS
CL -UNIVERSIDAD DE GUANAJUATO

A través de la presente constato que he revisado la tesis del C. **Ángel Isaac Gómez Canales** con el fin de obtener el grado de Licenciatura en Ingeniería Física. Su trabajo de tesis se titula **"Estudio de aprendizaje por refuerzo para procesamiento de imágenes médicas"**. En su trabajo de investigación, Ángel presenta un desarrollo de procesamiento de imágenes usando una red neuronal por refuerzo basada en "Deep learning" para discriminación de ruido y de patrones. Esto con la finalidad de usarse en la interpretación de imágenes médicas para un mejor diagnóstico. El trabajo de titulación satisface con la completez y solidez de un proyecto de titulación a nivel licenciatura. Ángel ha realizado las correcciones recomendadas al documento de la tesis. Además, he cuestionado a Ángel en los temas relacionados a su trabajo de tesis, demostrando su dominio en los temas abordados en su trabajo de tesis. Por lo que considero que ya puede proceder con los trámites para la disertación de tesis.

Sin más por el momento le envío saludos cordiales.

Atentamente

Dr. Carlos Herman Wiechers Medina
Profesor-Investigador

Tel. +52 (477) 7885100 Ext. 8467
Cel. +52 (477) 1080605
e-mail 1: carherwm@fisica.ugto.mx
e-mail 2: ch.wiechers@ugto.mx

UNIVERSIDAD DE GUANAJUATO

León, Guanajuato a 06 de diciembre 2022
Asunto: revisión de tesis

Dr. David Yves Ghislain Delepine
Director, División de Ciencias e Ingenierías, UGTO

Estimado Dr. Delepine,
Por este medio le informo que, después de haber revisado la tesis titulada "Estudio de aprendizaje por refuerzo para procesamiento de imágenes médicas" presentada para cubrir los requisitos necesarios para obtener grado académico de Licenciado en Ingeniería Física del estudiante Ángel Isaac Gómez Canales, apruebo la tesis arriba mencionada para su defensa ante el jurado asignado por la secretaría Académica de la DCI. Ángel presento su trabajo correctamente y además realizó las correcciones pertinentes que le fueron sugeridas por mí, terminando en un documento de rigor científico necesario para su nivel.

Sin más por el momento me despido cordialmente enviándole un saludo

Atentamente

Dra. Natalia Rincón Londoño
Profesora Asociada "C"
Departamento de Ingeniería, UGTO
r.rincon@ugto.mx

Universidad de Guanajuato

Asunto: Carta aval de sinodal

León, Gto., Noviembre 17, 2022

Dr. David Yves Ghislain Delepine
Director
División de Ciencias e Ingenierías
Estimado Dr. Delepine:

Por medio de la presente hago constar que he revisado la tesis titulada: "**Estudio de aprendizaje por refuerzo para procesamiento de imágenes médicas**, que para obtener el grado de Licenciado en Ingeniería Física presenta el **Sr. Ángel Isaac Gómez Canales.**

En dicho trabajo se presenta una implementación novedosa para la reducción de ruido en imágenes médicas mediante el empleo de técnicas de Machine Learning. Cabe destacar que este trabajo es importante por las potenciales aplicaciones que en diversos campos tienen estas técnicas, particularmente en el diagnóstico médico.

Le comunico que he discutido cuidadosamente dicha tesis con el sustentante, a quien le he hecho llegar mis comentarios y correcciones. Le expreso además que en lo general me parece un buen trabajo por lo que avalo su presentación.

Sin otro particular por el momento, aprovecho para reiterarle las seguridades de mi consideración más distinguida.

Atentamente

DR. MODESTO ANTONIO SOSA AQUINO
PROFESOR TITULAR "C"
Sinodal

DEPARTAMENTO DE INGENIERÍA FISICA,
DIVISION DE CIENCIAS E INGENIERÍAS, CAMPUS LEÓN
Loma del Bosque 103, Fracc. Lomas del Campestre, C.P. 37150 León, Gto., México. Tel. (477) 788-5100, Fax: (477) 788-5100 ext. 8410, http://www.fisica.ugto.mx