Reinforcement Learning: FlappyBird

Kenneth Obando Rodríguez, *Instituto Tecnológico de Costa Rica*
Alejandro Pacheco Quesada, *Instituto Tecnológico de Costa Rica*

***Resumen*—**

***Index Terms*—neural network, gradient descent, softmax, cross-entropy, relu, mnist, machine learning, image classification**

I. Introducción

EXISTEN problemas computacionales que requieren que un programa realice una acción ante un estado determinado, con la agravante que estos estados pueden tener muchas variables diferentes que lo convierten en una tarea prácticamente imposible para una aproximación estática de la solución. Ante estos problemas, surge toda una rama del campo de la Inteligencia Artificial que se llama Aprendizaje Por Reforzamiento *("Reinforcement Learning")*, que consiste, en términos generales, en entrenar un algoritmo utilizando premios *("reward"))* positivos o negativos, según sea el resultado de la acción tomada en un estado dado.

Esta investigación tiene como objetivo implementar un algoritmo de *Reinforcement Learning* conocido como "Policy Gradient" para entrenar un modelo computacional para que juegue con éxito "FlappyBird", para ello se utilizan herramientas como PyTorch,

II. Trabajo relacionado

Un dato interesante es que el famoso visionario Turing (1948,1950) propuso una aproximación del aprendizaje por reforzamiento aunque no estaba totalmente convencido de su efectividad: *"... el uso de castigos y premios puede, en el mejor de los casos, ser parte de un proceso de enseñanza."* [**?**], [**?**]. Se sospecha que la primera investigación exitosa en este campo es la de Arthur Samuel, el cual contiene una buena parte de las ideas modernas en parendizaje por reforzamiento, incluyendo la aproximación utilizando una función [**?**].

Desde principios de la década de los 90 se realizaron aportes en el campo del aprendizaje por reforzamiento utilizando *"Policy Gradient"* inicialmente gracias a Ronald Williams que desarrolló esta familia de algoritmos. Trabajos posteriores hicieron grandes aportes, aunque fue con Papavassiliou y Russel que se describe un nuevo tipo de reforzamiento que converge con cualquier tipo de aproximador de función [**?**]. A partir de estos estudios, se han hecho bastantes aproximaciones, y se ha llevado a una rápida y amplia implementación gracias a herramientas como PyTorch, TensorFlow y hardware como las GPU mediante librerías como CUDA.

Una buena síntesis sobre los métodos que existen actualmente en Policy Gradient se encuentra en [**?**].

III. Metodología

En términos generales, este proyecto se divide en diferentes etapas para el entrenamiento del modelo computacional: FlappyBird contiene toda la lógica del juego, el renderizado de las imágenes y recibe las acciones del agente; la etapa de preprocesamiento, la Red Convolucional y los reportes mediante gráficos e imágenes generadas utilizando la herramienta PLT.

III-A. Juego: FlappyBird

FlappyBird es un juego que consiste en hacer que un ave pase en medio de una serie de dos tuberías que se van aproximando constantemente, para ello, el jugador debe apretar una tecla (normalmente *"espacio"*) que provoca que el pájaro salte en el aire. El jugador debe ser lo suficientemente hábil para coordinar los saltos, la distancia y el tiempo.

Este juego contiene características especiales que permiten ser un candidato para ser utilizado para entrenar un agente mediante Aprendizaje por Reforzamiento:

- Sólo requiere dos acciones de parte del agente: saltar o no realizar ninguna acción.
- El agente sólo debe preocuparse por la ubicación del pájaro y la cercanía de las tuberías más próximas.
- Las figuras de las imágenes son sencillas, con colores sólidos, lo que facilita la etapa de preprocesamiento.

III-B. Preprocesamiento

Para facilitar la etapa de aprendizaje, se realiza un preprocesamiento de cada imagen generada por el juego, el cual consiste en aplicar un filtro a una capa del espectro visual. Mediante diferentes pruebas realizadas, se comprobó que en la capa verde los elementos importantes para el aprendizaje *"el pájaro y las tuberías"* mantienen una tonalidad semejante, lo que permite filtrar por los valores de esta tonalidad con un costo computacional mínimo. Además, se recorta las secciones de la imagen que no tienen un valor significativo para el aprendizaje. En la figura **??** se puede apreciar las diferentes etapas del preprocesamiento.

III-C. Red Neuronal

Para esta investigación se utilizó una red convolucional, que de base para los experimentos tiene tres capas, cada una con una capa de *"MaxPooling"*, la última capa es una red neuronal *"FullyConnected"* con una salida con SoftMax. Las características de cada capa se detallan en el Cuadro **??**. Cabe decir que, como parte de la experimientación, se probó con

Capa	Descripción	Cantidad de Kernels	Tamaño del Kernel	Stride	Activación
1	Convolucional	10	5	1	ReLu
2	MaxPooling	N/A	2	1	N/A
3	Convolucional	12	7	1	ReLu
4	MaxPooling	N/A	2	1	N/A
5	Convolucional	12	7	1	ReLu
6	MaxPooling	N/A	2	1	N/A
7	DropOut	N/A	N/A	N/A	N/A
8	Red Fully Connected	N/A	N/A	N/A	N/A

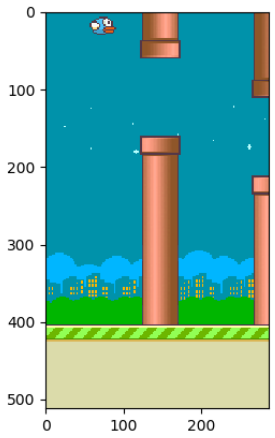
Cuadro I: Descripción de las capas de la Red Convolucional.

varias convinaciones de red convolucional para comprobar los resultados.

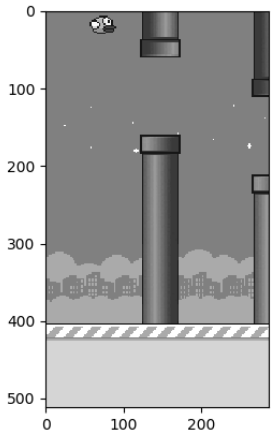
III-D. Entrenamiento y testing

III-E. Red neuronal

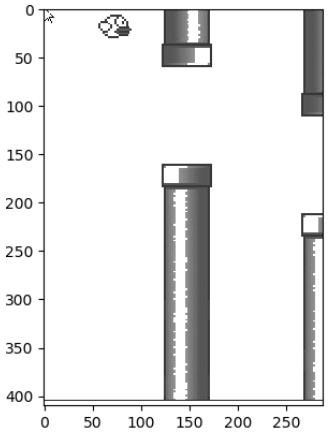
ANEXOS



(a)



(b)



(c)

Figura 1: Imágenes las diferentes etapas del preprocesamiento: