Treball de Recerca

Superant l'Humà en "Flappy Bird"

Un Estudi sobre l'Aprenentatge Automàtic

Marc Pérez Fusco, Jan Ferrer Paramio,
Teo Clerici Jurado
26-9-2024

ÍNDEX

ABSTRACT

En aquest treball, explorem les possibilitats de la Intel·ligència Artificial (IA) per a jugar al joc "*Flappy Bird*" utilitzant algoritmes avançats d'aprenentatge adaptatiu. L'ús combinat de mutacions i altres factors permet l'obtenció d'un model neuronal més eficient i capaç de realitzar càlculs ràpids. El model neuronal desenvolupat en aquest treball és capaç d'aprendre i adaptar-se a les condicions del joc. En aquesta recerca es demostra com la corba de l'aprenentatge és logarítmica i depèn fortament dels diferents comportaments de la IA a partir de les variables de mutacions, inputs i població.

INTRODUCCIÓ

Darrerament, les tècniques neuroevolutives s'han popularitzat molt en el món de la IA, i en aquesta recerca hem volgut aplicar aquestes tècniques a un videojoc anomenat "*Flappy Bird*", molt conegut entre els aficionats als videojocs. El treball vol mostrar com aprèn una IA que juga a un joc poc complex i com es pot millorar aquest aprenentatge, a partir de la variació de les condicions inicials del joc, per això s'analitza com és de logarítmic l'aprenentatge de la IA i com varia segons la configuració establerta variant la població, els inputs i els tipus de mutació.

A partir de l'anàlisi de les dades obtingudes d'aquest aprenentatge, també es planteja trobar aquells paràmetres que fan que la IA dugui a terme la tasca de la millor manera, així com aquelles condicions més eficaces per entrenar-la, com ara quants agents són òptims pel seu aprenentatge i quantes generacions són necessàries per entrenar una IA perquè pugui dur a terme la tasca encomanada de forma perfecta.

Per obtenir els resultats que ens han permès dur a terme a aquesta anàlisi, s'ha construït a tall de laboratori digital un "*Flappy Bird*" específic en *Godot Engine*[1] per poder inserir-hi una IA que hem creat exclusivament per aquest propòsit. Aquesta IA l'anomenem *Autonomous Player Fast Learning Yeld* (APFLY-IA). Per últim, hem dotat la IA d'un sistema de telemetria (logs) integrat, encarregat de recopilar les dades del comportament i aprenentatge que són les que s'han fet servir per fer l'estudi.

COM FUNCIONA LA APFLY-IA

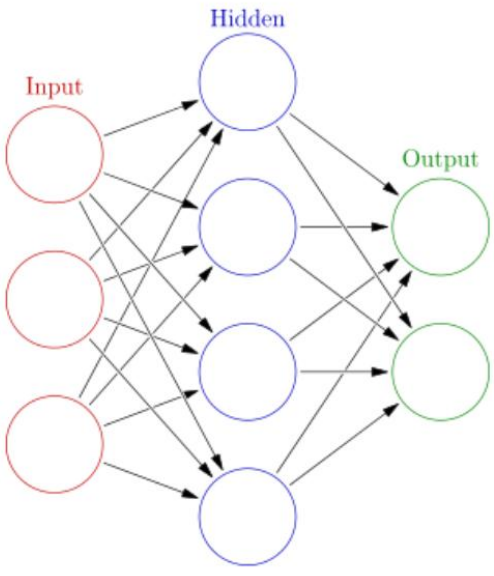

Figura 1: Exemple d'una xarxa neuronal [2]

Les xarxes neuronals són un sistema de ML (*Machine Learning* o Aprenentatge Automàtic) que es basa en la biologia, en l'estructura neuronal dels animals. Aquestes xarxes es representen en forma de graf, on els vèrtexs són les neurones encarregades d'emmagatzemar la informació i on les arestes són les connexions encarregades de transformar aquesta informació i transmetre-la a les neurones connectades. La primera columna, vermella amb títol d'*Input* són les neurones on col·locarem la informació necessària per fer funcionar la xarxa neuronal; la segona columna, blava amb títol *Hidden*, són les neurones encarregades d'emmagatzemar la informació que encara no s'ha acabat de processar (per problemes que són lineals aquesta capa és prescindible), i la tercera columna, verda amb títol *Output*, són les neurones encarregades de proporcionar les dades de sortida. Cada connexió té associat un valor entre el -1 i l'1 que li direm pes o importància. Per saber el valor d'una neurona s'ha de sumar les multiplicacions entre les neurones estrictament precedents i el pes de la seva connexió.

En cas d'un problema no lineal els resultats de les neurones ocultes s'haurà de passar per una funció d'activació, com pot ser una funció sigmoide. En el cas del APFLY-IA, s'haurà de passar la neurona de

sortida per aquesta funció per acotar el valor resultant entre 0 i 1, si el valor és superior a 0,7 el personatge saltarà.[3]

Per entrenar la xarxa neuronal utilitzem una versió d'un algorisme genètic, inspirat en la teoria Darwiniana, anomenat NEAT (*Neuro Evolution of Augmenting Topologies*). En primer lloc, es crearà un nombre N (a definir) de xarxes neuronals, que anomenem de forma individual "agents" i de forma col·lectiva "població" i per cada població l'anomenarem segons el seu ordre d'aparició com a generació. La primera població que apareix serà la generació 0 (zero), la segona 1, i així successivament. Tots els agents d'una població s'executaran alhora en la simulació.

Les connexions de les neurones de cada agent de la generació 0 serà totalment aleatòria i l'estructura de la xarxa neuronal serà predeterminada: una neurona per cada *input* existent, una neurona de sortida i connexions que connectin les neurones *input* amb la neurona *output*. Per la següent generació agafarem el 10% més 1 dels millors agents de la generació anterior, a través d'una puntuació anomenada *fitness* que segons cada tasca té els seus criteris de puntuació, i la resta de població restant es crearà a partir de "mutar" els agents escollits de la generació anterior o de barrejar dues xarxes neuronals que s'han escollit de la població anterior.

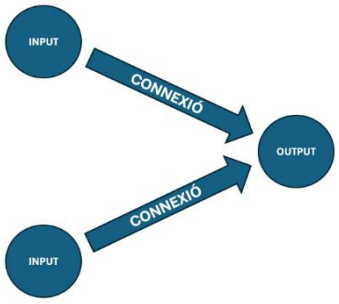

Figura 2: Esquema de l'estructura predeterminada de la xarxa neuronal d'APFY-IA

La barreja de dos agents, o *crossover (Crs)*, consisteix en crear una nova xarxa neuronal a partir de dues xarxes neuronals amb la mateixa estructura que anomenarem pares (mateix nombre de neurones i igual connectades). Per aquesta nova xarxa neuronal s'agafarà la mateixa estructura i pel pes de les connexions s'agafarà la mitjana aritmètica del pes de les connexions dels pares.

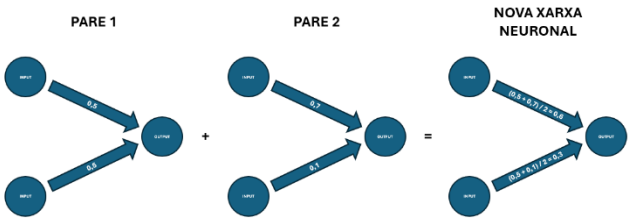

Figura 3: Exemple de Crossover (Crs)

Per la mutació d'un agent, el procediment és més complex perquè pot mutar de diferents maneres (la manera s'escollirà de forma aleatòria), de forma no estructural (MNE) o de forma estructural (ME). La no estructural consisteix a canviar el pes d'una connexió per un altre valor aleatori. En canvi, les mutacions estructurals modifiquen el graf: poden crear una neurona nova o destruir-la i també poden crear una nova connexió, o destruir-la.

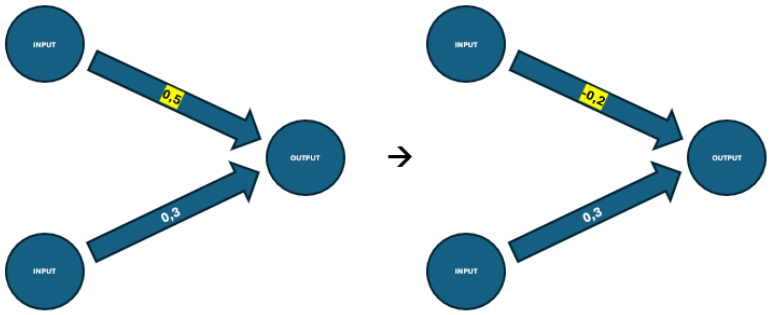

Figura 4: Exemple de la mutació no estructural (MNE)

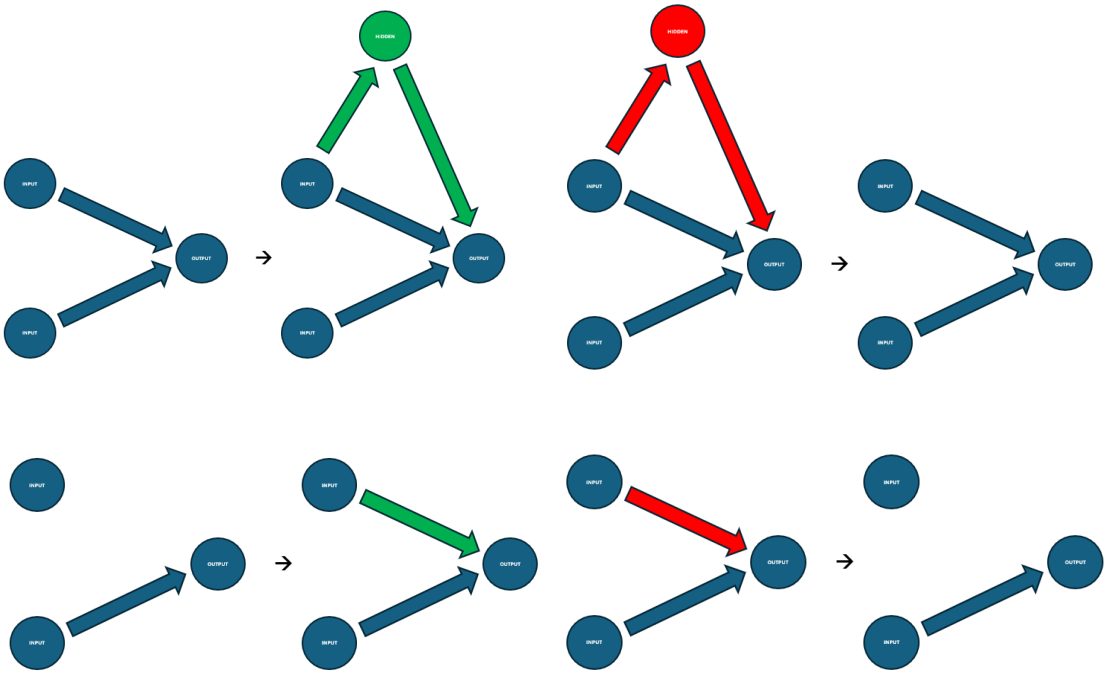

Figura 5: Exemples de les mutacions estructurals

Quan es crea una neurona nova, és necessari connectar-la amb la xarxa, aleshores es crea una connexió que té com a destí aquesta nova neurona i té una altra connexió que parteix des d'aquesta neurona fins a una altra, i quan es destrueix una neurona també han de desaparèixer totes les connexions amb les quals interactuava directament.

Un cop s'ha creat la nova generació es repeteix el procés fins a aconseguir un agent que assoleix fer la tasca de forma correcta.

LA APFLY-IA DINS DEL CONTEXT "*FLAPPY BIRD*"

FLAPPY BIRD

Flappy Bird és un joc que es va popularitzar a partir de l'any 2013 i que consisteix a controlar un ocell que ha de volar entre una sèrie de tubs cilíndrics que apareixen a la pantalla. Per mantenir-lo en vol, el jugador ha de tocar la pantalla enlairant l'ocell lleugerament, i si no es toca, l'ocell cau a causa de la gravetat. L'objectiu principal és passar entre els tubs sense xocar-hi ni caure a terra, i cada cop que l'ocell passa amb èxit entre dos tubs, el jugador guanya un punt.

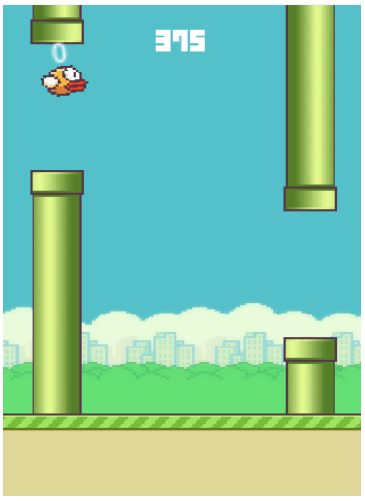

Figura 6: Flappy bird

És un dels videojocs més senzills del qual una IA pot aprendre a jugar. Es tracta d'un problema lineal, ja que el seu objectiu és no caure per sota d'una línia horitzontal imaginària col·locada a la base superior d'obstacle, de manera que pugui passar per dins de l'obstacle sense xocar. Aquesta línia es pot aconseguir amb una xarxa neuronal lineal, ja que quan la seva posició en l'eix vertical sigui inferior o igual al de la base de l'obstacle l'agent saltarà.

FITNESS

El *fitness* és un sistema de puntuació que s'utilitza per saber quina xarxa neuronal ha acomplert millor la tasca. En la APFLY-IA s'entreguen 1000 punts per passar un obstacle, es penalitza amb -100 punts si mor per tocar el terra. Segons com a prop vola l'ocell de la posició Y del centre del forat (PYF) - com és mostra a la imatge - es recompensa l'ocell segons la zona amb 100, 50, 10 o 5 punts per cada segon de més a prop a més llunyà que vola. Finalment per cada segon que l'agent està viu se li entrega un punt. Després que totes les xarxes neuronals morin o arribin a la puntuació màxima establerta, per fer d'un joc infinit un de finit, després de passar per 150 tubs, es farà una llista de la millor a la pitjor xarxa neuronal segons els criteris anteriors per poder escollir les millors per la següent generació. Per tal que en perfeccionar-se la IA no estigués jugant a la mateixa partida de forma infinita, s'ha establert un màxim de 160.000 de fitness aproximadament (passades 150 canonades). Quan la IA arriba a aquesta puntuació *fitness* tres vegades seguides, es dona per suposat que ja ha après i acaba la partida.

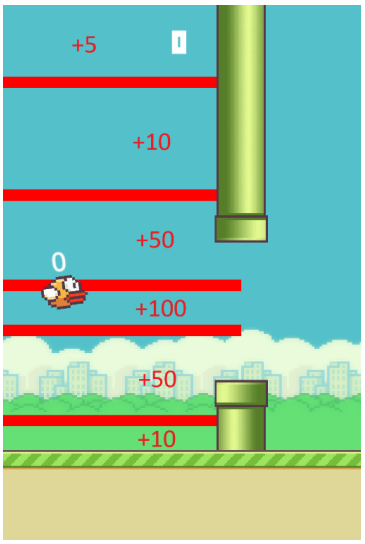

Figura 7: Esquema de la obtenció del fitness segons la posició y de l'ocell

INPUTS

Els *inputs* són aquelles dades que donem a la xarxa neuronal perquè porti a terme la seva tasca correctament. En la APFLY-IA donarem a cada xarxa neuronal 4 possibles dades que anirem variant per poder fer l'anàlisi. Aquestes són: la posició Y de l'ocell (PYO), la posició Y del centre del forat de l'obstacle (PYF), la posició X del centre del forat de l'obstacle (PXF) i la velocitat Y de l'ocell (VYO).

ALTRES CONSTANTS

Per la recollida de dades hi ha variables que canvien, que són els *inputs*, els tipus de mutació i la població per generació, en canvi, n'hi ha d'altres que no varien. Per cada configuració és fan entre 20 i 30 partides, per cada partida hi ha 125 generacions i la puntuació màxima establerta és de 150, és a dir que després d'haver passat per 150 tubs la generació actual morirà i s'iniciarà la següent generació.

L'APRENENTATGE

A l'inici d'aquest treball de recerca una de les hipòtesis plantejades era la forma que prendrien els gràfics que relacionen la puntuació dels agents d'intel·ligència artificial amb el pas les generacions al llarg d'un entrenament.

La nostra hipòtesi era que la forma seria logarítmica, i després de 3.798 partides i més de 450.000 generacions amb diferents paràmetres (població, mutacions i *inputs*) **hem vist que, efectivament, la forma del gràfic demostra que l'aprenentatge és logarítmic, i per confirmar-ho utilitzem un valor anomenat R^2.**

La R^2 (coeficient de determinació) en un gràfic logarítmic mesura com de bé s'ajusta una línia de tendència logarítmica a les dades. S'interpreta i calcula de manera similar a la R^2 en un gràfic lineal, però considerant la transformació logarítmica. El coeficient varia entre 0 i 1, on 1 suggereix que el model logarítmic explica perfectament la relació entre les variables, mentre que 0 indica que el model no explica gens aquesta relació. Un valor més proper a 1 implica que el model logarítmic captura millor la variabilitat de les dades en comparació amb altres models més simples, com el lineal. Es calcula amb la següent fórmula[4]:

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

Investigant més a fons podem veure que, tot i que la majoria dels gràfics tenen una forma logarítmica, no sempre segueixen exactament el mateix patró logarítmic. A la següent taula s'utilitza el valor mitjà de R^2 en una determinada configuració, relacionant les dades d'entrada i les possibles mutacions que es poden

aplicar a la IA, sense tenir en compte la població per generació, ja que es va comprovar que la població no modificava la forma del gràfic de manera significativa.

	-	PYO	PYF	PYO, PYF	PYO, PYF, PXY	PYO, PYF, VYO	PYO, PYF, PXY, VYO	INPUTS
-	0,535	0,21	0,228	0,638	0,759	0,764	0,608	
MNE	0,516	0,439	0,097	0,562	0,706	0,812	0,401	
ME	0,534	0,14	0,413	0,669	0,766	0,712	0,501	
MNE, ME	0,531	0,114	0,317	0,783	0,786	0,673	0,51	
MNE, Crs	0,55	0,159	0	0,539	0,747	0,893	0,755	
MNE, ME, Crs	0,549	0,166	0,245	0,586	0,799	0,776	0,724	
MUTACIONS								

Valors mitjans de R^2 per a diferents combinacions d'entrades (columnes) i mutacions (files). Escala monocromàtica: roig indica un ajust pobre una funció logarítmica (valors propers a 0) i blanc indica bon ajust (valors propers a 1).

A la taula es fa referència a les entrades i mutacions de forma abreviada, i aquestes són les corresponents equivalències:

MUTACIONS	ENTRADES
"**-**" → S'inclouen els valor de totes les possibles configuracions de mutacions. "**MNE**" → No estructural "**ME**" → Estructural "**Crs**" → *Crossover*	"**-**" → S'inclouen els valor de totes les possibles configuracions d'entrades. "**PYO**" → Posició Y de l'ocell "**PYF**" → Posició Y del forat del obstacle "**PXY**" → Posició X del forat del obstacle "**VYO**" → Velocitat Y de l'ocell

Les mutacions inclouen configuracions com No estructural (MNE), Estructural (ME) i Crossover (Crs), mentre que les entrades es refereixen a diferents paràmetres del joc, com la posició i velocitat de l'ocell i la posició dels obstacles. El símbol "-" indica que s'inclouen totes les possibles configuracions per a aquella categoria.

A la taula es pot observar que els aprenentatges menys logarítmics són els de les columnes sense les entrades **PYO** o **PYF**, això es deu al fet que aquestes entrades són imprescindibles perquè la IA pugui aprendre, i per això en aquestes columnes no es produeix cap mena d'aprenentatge. En canvi, la columna on s'utilitzen les entrades **PYO**, **PYF**, **VYO** és on el gràfic té més forma logarítmica i, per ser exactes, la columna on s'utilitzen les entrades **PYO**, **PYF**, **VYO** i les mutacions **MNE**, **Crs**; és on el coeficient és més alt, per tant, és el més logarítmic.

La mitjana del coeficient R^2 dels gràfics que presenten un aprenentatge, durant les 125 generacions, és de 0,73. Aquest valor és indicatiu d'un aprenentatge ràpid logarítmic, però, a què és degut aquest comportament logarítmic?

En les primeres generacions, la IA obté guanys significatius en la puntuació de *fitness*. Això es deu al fet que hi ha moltes possibilitats de millora i, petits canvis en l'estratègia poden portar a grans augments del *fitness*. De la mateixa manera, a mesura que la IA es troba a generacions més avançades, els canvis substancials que poden portar a una millora en el *fitness* són canvis més subtils i la seva millora respecte al *fitness* també és menor.

L'anàlisi d'aquesta tendència logarítmica a priori pot semblar poc útil, però té bastants aplicacions. Per exemple, permet predir els ràpids guanys inicials de *fitness* seguits d'un altiplà, de manera que es pot determinar quin és el millor moment per aturar l'aprenentatge o afegir nous reptes a l'aprenentatge per obtenir uns millors resultats.

COM AFECTEN LES VARIABLES A L'APENENTATGE

En aquesta secció, analitzarem com les diferents variables d'entrada (inputs) i les seves combinacions influeixen en l'aprenentatge i el rendiment de la xarxa neuronal en el joc Flappy Bird. A través d'aquest anàlisi, compararem escenaris en què la xarxa disposa de diferents tipus d'informació, per tal de determinar quina és més rellevant per a la seva capacitat de prendre decisions efectives. Això ens permetrà entendre millor les necessitats d'informació de l'IA i com aquestes afecten els seus resultats en el joc.

Com ja hem mencionat anteriorment, un dels objectius d'aquest treball és mostrar com aprèn una IA que juga a un joc poc complex i com es pot millorar aquest aprenentatge, a partir de la variació de les condicions inicials del joc. En aquest apartat passem a analitzar com la l'aprenentatge de la IA varia segons la configuració establerta variant la població, els inputs i els tipus de mutació.

Les diferents combinacions d'*inputs*, mutacions i poblacions generen diferents comportaments en l'evolució i l'aprenentatge de la xarxa neuronal. A partir de l'estudi realitzat, a continuació s'explica breument com es comporta la xarxa neuronal depenent de la variació dels paràmetres introduïts.

ANÀLISI DELS *INPUTS*

Quan la xarxa neuronal només coneix la posició Y de l'ocell (PYO), els resultats són completament aleatoris anant des de 0 fins a 4.000 de *fitness*. Mai assoleix una dada superior de 4.000 de *fitness*, perquè no té la informació necessària per arribar al seu objectiu, pel fet que amb les dades de què disposa no pot saber a quina posició es troba el forat de l'obstacle. L'única cosa que pot fer és evitar caure a terra i estar a una alçada en la qual pot haver-hi un forat. (La puntuació màxima com s'ha dit anteriorment és de 160.000 punts de *fitness* i una puntuació de 4.000 punts no presenta cap indici d'un aprenentatge rellevant).

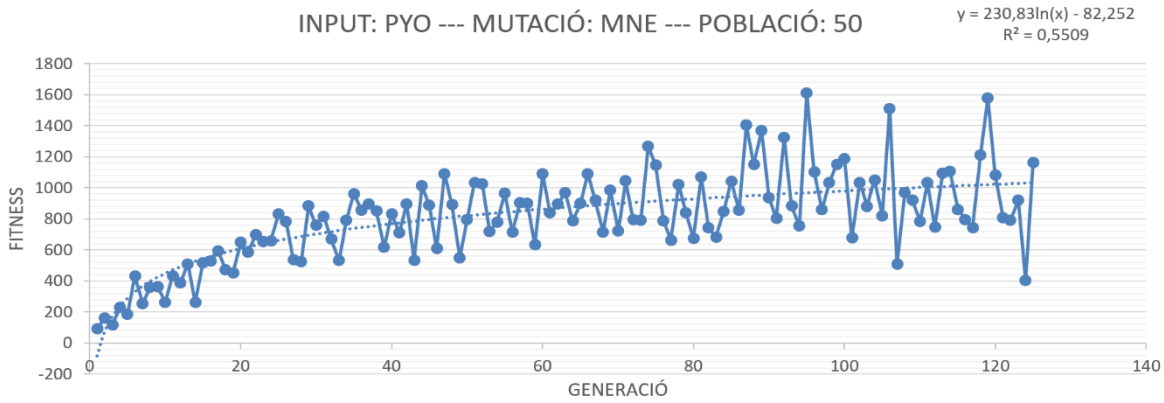

Figura 8: Evolució de la IA amb Input PYO (mutació MNE, població 50)

D’altra banda, quan només li donem la posició Y de l’obstacle (PYF), la xarxa neuronal només sap a quina alçada està el forat de l'obstacle i no la posició de l'ocell, per la qual cosa no és capaç de superar-lo ni de manera aleatòria, com sí que passava amb l'*input* PYO, perquè està jugant completament a cegues.

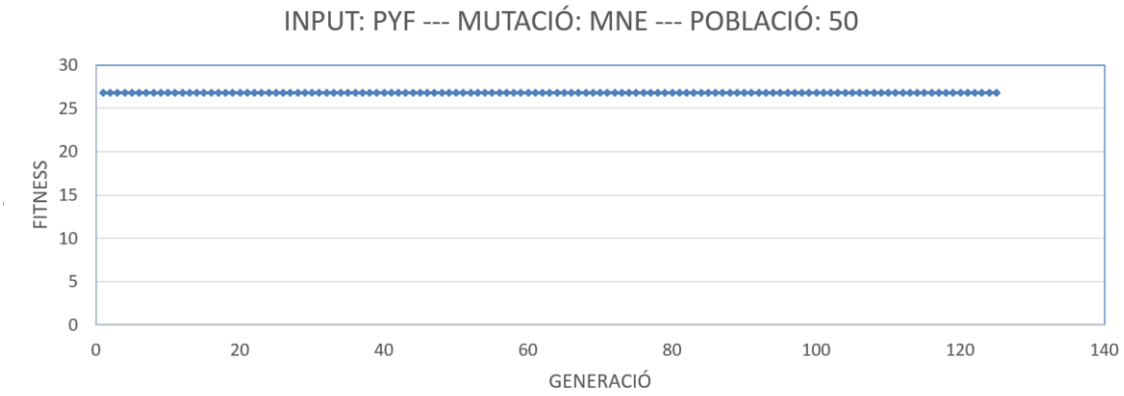

Figura 9: Evolució de la IA amb Input PYF (mutació MNE, població 50)

Quan la xarxa neuronal de la IA coneix la posició Y de l'ocell i la posició Y de l'obstacle, es veu l'aprenentatge de la xarxa és ràpid i efectiu, perquè té la informació necessària per completar la tasca (inputs PYO-PYF separats).

Però si en comptes de tenir aquestes dues variables per separat les relacionem de forma directa (a través de la seva diferència) es veu que el resultat és més eficient, millora la velocitat d’aprenentatge i redueix la complexitat de la xarxa neuronal, oferint uns millors resultats, acord amb la gràfica mostrada a continuació (inputs PYO-PYF junts).

Això és degut a que la importància del pes de les connexions entre neurones no recau en els valors numèrics absoluts en si mateixos sinó en la posició relativa establerta entre les dues variables. Per exemple el quadre que es mostra a continuació, correspon a una xarxa neuronal amb els *inputs* PYO i PYF on aquest 0,5 podria haver estat un 0,3 o un 0,8 però el que realment importa és que l’altra connexió sigui un -0,5; un -0,3 o un -0,8 respectivament, és a dir que conservin la mateixa relació entre ells.

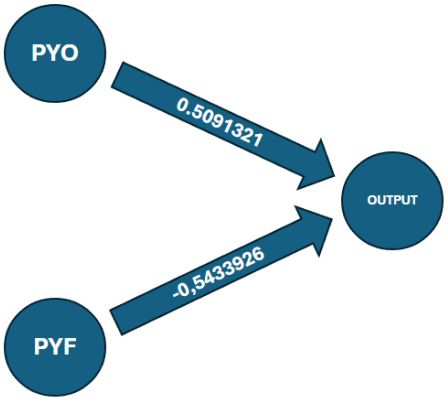

Figura 10: Exemple d’una IA perfecte

Aleshores podem concloure que amb un sol *input* que relaciona aquestes dues variables de forma directa, l'aprenentatge és més eficient, millorant la velocitat i reduint la complexitat de la xarxa neuronal, oferint uns millors resultats d'acord amb les gràfiques *fitness* - generació.

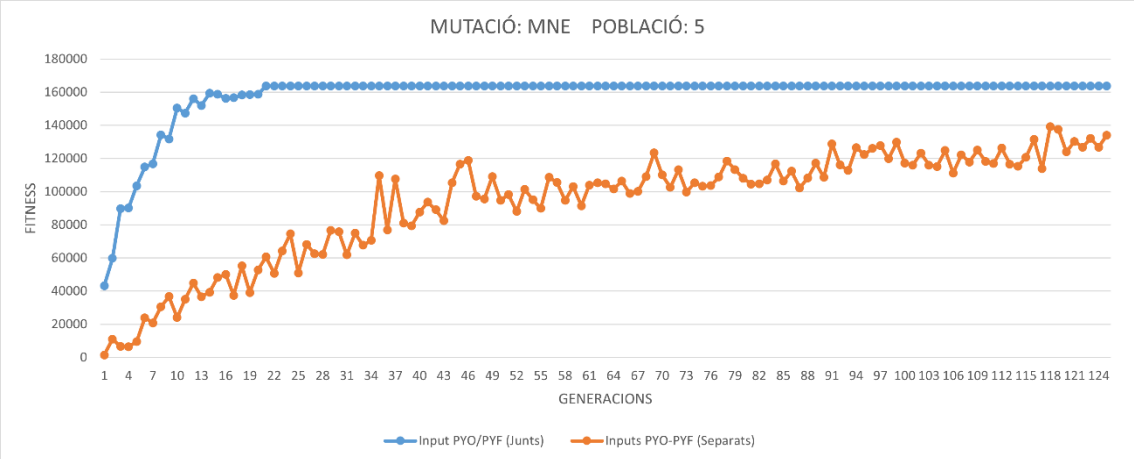

Figura 11: Evolució de la IA segons el si l'input PYO-PYF estan junts o separata (mutació MNE, població 5)

Quan oferim la posició Y de l'ocell i la posició Y i posició X del forat de l'obstacle (PYO-PYF-PXF) el temps d'aprenentatge de la IA augmenta respecte a la configuració d'*inputs* (PYO-PYF) a causa de l'augment de la complexitat de la xarxa neuronal, encara que una de les opcions de la IA sigui ignorar o eliminar aquesta informació (PYF), també té la possibilitat d'integrar-la a la solució, podent resoldre aquest problema de dues maneres diferents. En canvi, si combinem la posició Y de l'ocell, la posició Y de l'obstacle i la velocitat Y de l'ocell (PYO-PYF-VYO), passa a ser una configuració menys efectiva que la anterior, ja que l'aprenentatge s' alenteix més per la presència de la velocitat Y, un *input* que la xarxa ha de ignorar o eliminar.

Quan ajuntem tots els *inputs* posició Y de l'ocell, posició Y i X de l'obstacle i velocitat Y de l'ocell (PYO-PFY-PXY-VYO) alenteix encara més l'aprenentatge i passa a ser una de les xarxes neuronals menys bones, traient de banda les que no tenen prou informació- com que ha d'ignorar l'excés d'informació.

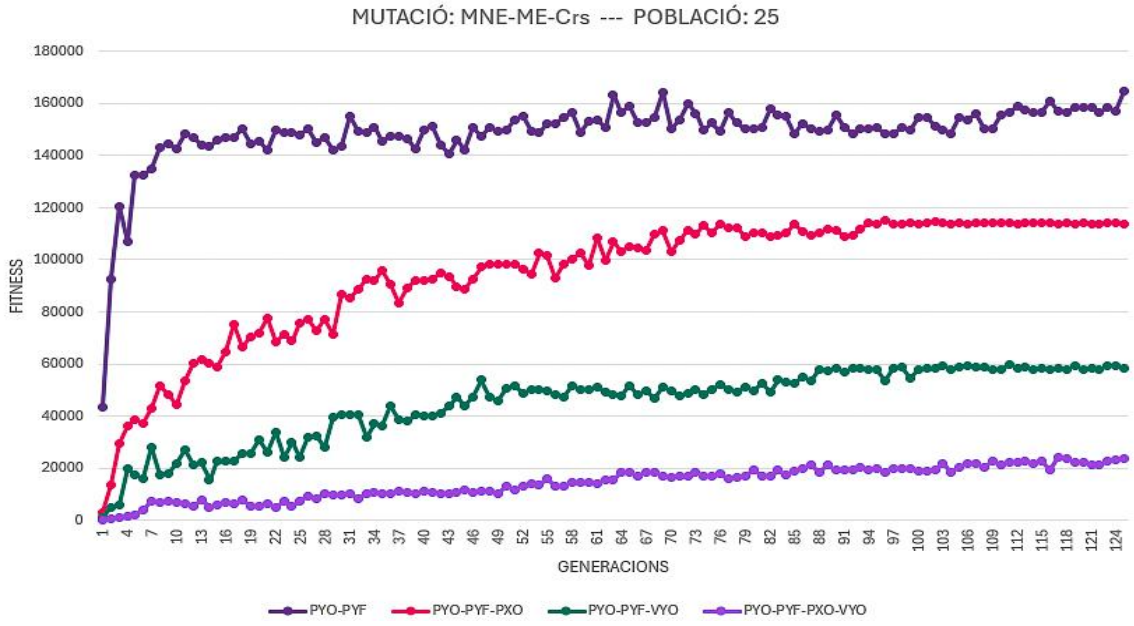

Figura 10: Evolució de la IA segons l'input (mutació MNE-ME-Crs, població 25)

És a dir, com més informació innecessària rep la xarxa neuronal, i més complexa és aquesta, l'aprenentatge s'alenteix i/o es limita. Com que la xarxa neuronal és lineal i aquesta pensada per resoldre problemes lineals no massa complexos, com el *Flappy Bird*, l'aprenentatge s'alenteix i/o es limita més que altres xarxes neuronals pensades per resoldre problemes més complexos.

ANÀLISI DE LES MUTACIONS

L'aplicació de la mutació no estructural (MNE), en general provoca unes puntuacions de *fitness* més altes i un aprenentatge més ràpid quan l'apliquem sola. Però quan es tracta de generacions on es dona un excés d'informació o on falta informació, l'aprenentatge de la mutació s'alenteix perquè no pot eliminar ni crear connexions i ha de perdre temps aproximant el valor d'una connexió a 0.

D'altra banda, quan ajuntem la MNE amb més mutacions, perd eficiència (MNE-ME, MNE-Crs i MNE-ME-Crs).

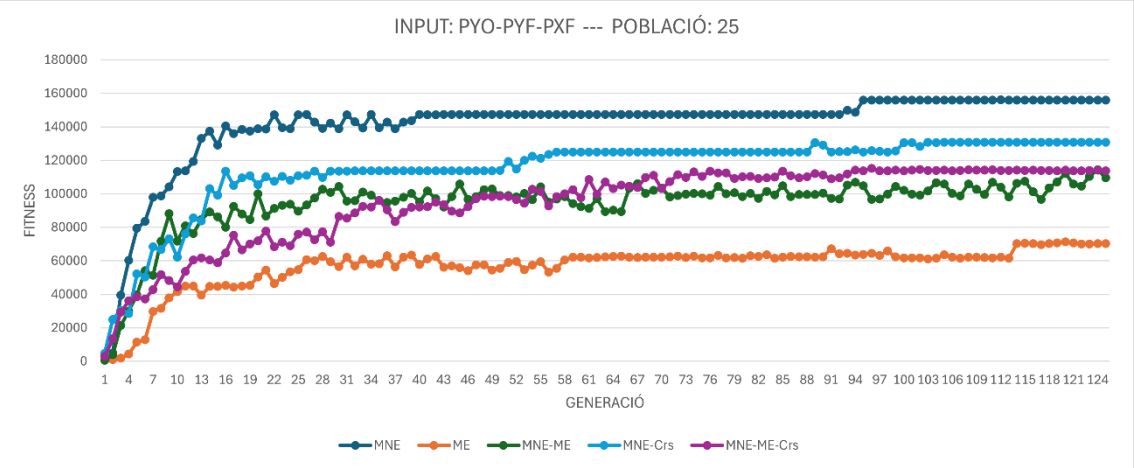

Figura 11: Evolució de la IA segons la mutació (input PYO-PYF-PXF, població 25)

Quan apliquem la mutació estructural (ME) només genera un resultat per sobre de les altres mutacions quan es tracta d'aplicar aquesta mutació a generacions amb excés o falta d'*inputs*. Però la ME és superada per altres mutacions (MNE) al cap d'unes quantes generacions, pel fet que la ME inicia amb un primer salt, però després s'estanca per fer que la xarxa neuronal sigui més complexa. D'altra banda, quan ajuntem la ME amb més mutacions (MEN-ME i MNE-ME-Crs), perd eficiència per generacions amb *inputs* no gaire bons, però guanya eficiència quan es donen bons *inputs* d'entrada, assegurant que superi a quant n'hi ha la mutació sola.

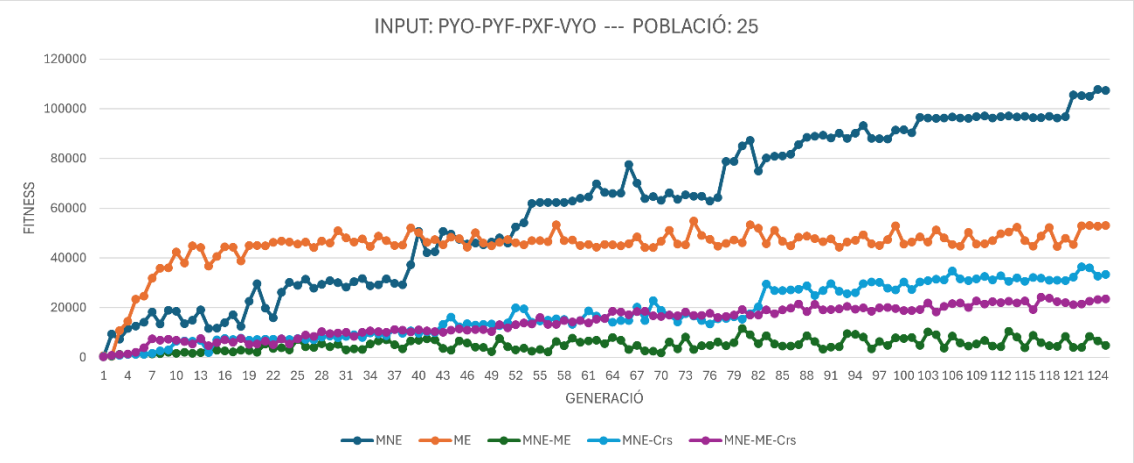

Figura 12: Evolució de la IA segons la mutació (input PYO-PYF-PXF-VYO, població 25)

D'altra banda, la implicació del *crossover* (MNE-Crs MNE-ME-Crs) no provoca canvis notables en la majoria de partides i a més a més no segueix un patró aparent, la mateixa execució en moments diferents provoca resultats diferents sense una explicació plausible del seu perquè.

ANALISI DE LES POBLACIONS

Referent a la població, veiem que com més IAs estiguin jugant a la vegada, el seu aprenentatge serà més ràpid perquè al ser al mostra més gran la probabilitat de resultats eficients també és més elevat i com que les següents generacions parteixen d'aquests resultats el seu aprenentatge s'escurça

Cal recalcar que no tots els gràfics permeten corroborar aquestes conclusions al peu de la lletra per culpa de la quantitat de variables que poden afectar com es comporta cada generació, però l'anàlisi precís dels gràfics fets anteriorment busca ser el més possible.

CONCLUSIÓ

Podem concloure que les variables que permetran a la APFLY-IA aprendre de forma més ràpida són els *inputs* posició Y de l'ocell i posició Y del forat (PYO-PYF), la mutació no estructural i la població més alta que et permeti el dispositiu que utilitzis per entrenar la IA sense alentir la velocitat de la simulació (en la nostre anàlisi la població més alta és de 100).

Com que podem dir que els resultats de l'aprenentatge segueixen la forma d'una funció logarítmica, podem marcar un límit recomanat de generacions per la millor configuració trobada, PYO-PYF, per poder aconseguir la IA que sàpiga executar la tasca sense haver de fer les 125 generacions. Segons les dades recollides a partir de la 8a generació sempre s'obtindrà una IA perfecta (tot i que és possible que en casos extraordinaris encara necessiti alguna generació més).

Som conscients que les possibilitats de noves anàlisis i noves conclusions poden entendre's més enllà d'aquest treball, i per això deixem a disposició de tothom per seguir investigant, el laboratori digital[5] que hem dut a terme en el marc d'aquest treball de recerca.

En aquest sentit, volem oferir unes primeres línies de futur en algunes de les quals s'està començant a treballar.

LÍNIES DE FUTUR

Mirant cap al futur, la recerca realitzada en aquest treball obre la porta a noves línies d'investigació que podrien ampliar aspectes en que no s'ha pogut aprofundir o donar-los suficient rellevància, i oferir una continuïtat i ampliació a la recerca fins ara feta.

El primer aspecte a ampliar consistiria a passar d'una xarxa neuronal lineal, plantejada en un inici per resoldre problemes simples, a una xarxa neuronal complexa, i estudiar les diferències entre els dos tipus de xarxes i els resultats obtinguts. En aquest sentit creiem que a l'utilitzar una xarxa neuronal complexa ens permetria:

1. Plantejar nous tipus de paràmetres i variables a analitzar, la qual cosa faria que a la vegada els problemes i jocs que podrien ser analitzats també podrien ser més complexos.
2. Poder aplicar més d'una mutació per IA i amb això calibrar el seu nombre per estudiar la millora en l'aprenentatge.

Per exemple, per fer un problema més complex a analitzar, es podria implementar en el joc de Flappy Bird una funció en què aparegués una moneda en una alçada aleatòria i que l'ocell hagués de veure si és viable agafar la moneda o si en agafar-la es moriria, i en funció d'això decidir que fer. Per fer això, s'hauria d'adaptar la APFLY-IA i dissenyar una forma més correcta d'administrar el *fitness (*el sistema de puntuació que s'utilitza per saber quina xarxa neuronal ha acomplert millor la tasca) i també adaptar la forma

d'entrenar la IA, si per etapes separades per cada dificultat concreta o si directament amb totes les dificultats alhora.

Augmentar les dificultats i sobretot passar d'una xarxa lineal a una complexa implica poder disposar de molta més capacitat de processament de dades (que els equips informàtics dels quals disposem fins ara no ens poden proporcionar) per l'obtenció dels resultats. Per agilitar aquest procés s'hauria de millorar la forma de la recollida de dades, ordenant i visualitzant-les d'una forma més senzilla i recopilant-les de forma més ràpida i eficient. Per agilitar aquest procés es proposen dos principals camins a seguir:

- Disposar d'equips informàtics més potents i amb molta més capacitat de processament matemàtic (per exemple ampliant la seva GPU – Graphic Processing Unit).
- Passar d'un processament i recopilació de les dades de resultats centralitzat a un processament i recollida de dades descentralitzada o distribuïda.

La primera opció és la més immediata i no requeriria pràcticament la modificació dels processos actuals més enllà de les optimitzacions que es creguessin convenients. La principal dificultat d'aquesta opció és el seu cost econòmic i la poca escalabilitat que ofereix a la llarga.

La segona opció, tot i que requereix més esforços de desenvolupament, permetria a un baix cost, obtenir molta més capacitat de processament al repartir de forma distribuïda en diferents sistemes de baixa capacitat la tasca d'obtenció dels resultats.

Per dur a terme aquesta segona opció, s'hauria de repartir, com en més col·laboradors voluntaris millor, una còpia del joc ja parametritzat amb uns paràmetres definits. Els resultats obtinguts es podrien transferir en format JSON (ja implementat al sistema actual) a un sistema central que s'encarregaria de recollir les dades per processar-les, visualitzar-les i analitzar-les en conjunt. Aquest mecanisme es podria fer cíclic per tal d'anar realimentant els jocs amb nous paràmetres a partir dels resultats analitzats.

Tot i afegir més complexitat i requerir més esforç aquesta segona opció permetria a la llarga obtenir més capacitat de processament. Si el projecte busqués no només l'anàlisi de resultats sinó també l'aprenentatge de la IA de forma contínua, potser la complexitat requerida la podria fer inviable i s'hauria d'optar per la primera opció o una combinació de les dues.

Una darrera línia a futur podria consistir a adaptar la IA del joc en forma de mòdul de tal manera que es pogués utilitzar en altres programes de recerca o directament en jocs dels quals es volgués dotar d'una IA per exercir funcions de personatges no jugadors. Per assolir aquest objectiu a banda d'extreure la IA del joc i donar-li forma de mòdul, s'hauria d'establir un estàndard per tal que el mòdul es pogués acoblar en els sistemes de destí. L'adaptació universal seria molt complexa, però creiem que l'adaptació a plataformes com *Godot Engine* no hauria de ser excessivament difícil.

INTEGRACIÓ D'UN ODS

8. Promoure el creixement econòmic sostingut, inclusiu i sostenible; l'ocupació plena i productiva, i el treball digne per a totes les persones. Obre en una nova finestra.

Promou el creixement econòmic, ja que aquesta anàlisi ajuda a fer un ús més responsable, per part de les empreses, de les noves IA i pot eliminar llocs de treballs repetitius on la salut mental del treballador es pot veure afectada.

BIBLIOGRAFIA

1. Engine, G. Godot Engine - Free and open source 2D and 3D game engine. *Godot Engine* https://godotengine.org/.
2. Glosser.ca. *English: Artificial Neural Network with Layer Coloring*. (2013).
3. McIntyre, A., Kallada, M., Miguel, C. G., Feher de Silva, C. & Netto, M. L. neat-python. (2024).
4. D, E. Looking at R-Squared. *Medium* https://medium.com/@erika.dauria/looking-at-r-squared-721252709098 (2019).
5. FerrerJan/Treball-de-Recerca-IA. https://github.com/FerrerJan/Treball-de-Recerca-IA.