UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Grau en Enginyeria Informàtica

Exploració de l'ús de la regla 110 dels autòmats cel·lulars elementals com a reservori en sistemes de reservoir computing en problemes de classificació

Treball de Fi de Grau

Memòria

Universitat politècnica de Catalunya

Facultat d'Informàtica de Barcelona

Autor: Ferran Esteban Ripoll
Director: Jordi Delgado
Especialitat: Computació
2020 – 2021 Q2

22 de juny de 2021

Índex

1 Introducció i contextualització

El món de la informàtica ha crescut immensament en les últimes dècades i han aparegut moltes branques que han aportat grans sistemes que avui en dia afecten a la nostra societat. Gran part d'aquest creixement s'ha basat en l'experimentació, en prova i error. Molts dels sistemes que tenim avui en dia han tingut una evolució al llarg dels anys gràcies a persones que han investigat i experimentat amb altres mètodes per buscar millors solucions.

La classificació d'imatges ha sigut una de les principals àrees d'investigació en el camp de la *Visió per Computador* i en els darrers anys hi ha hagut una gran quantitat d'estudis, experiments i una diversitat de tècniques dintre del camp del *Machine Learning* (aprenentatge automàtic) per analitzar la fiabilitat d'imatges reconegudes per l'ordinador.

Aquest treball de fi de grau de modalitat A se situa en el marc de la Facultat d'Informàtica de Barcelona i pertany a l'especialitat de *Computació*. Té com a objectiu experimentar i investigar si els autòmats cel·lulars elementals són capaços de servir com a *reservoir* en sistemes de *reservoir computing* (reservori computacional) dedicats a problemes de classificació, posant èmfasi en el *benchmark* estàndard *MNIST*.

1.1 Definició de conceptes

En els següents apartats es tractaran tots els conceptes necessaris per entendre i contextualitzar el problema i l'experiment que és realitzarà.

1.1.1 Autòmats cel·lulars

Un dels problemes que molts científics s'han trobat al llarg dels segles es que modelar el món real és molt difícil. Una de les branques de les matemàtiques discretes que intenta aclarir l'aleatorietat i el soroll innat dels sistemes complexos es coneix com autòmats cel·lulars [1].

John Von Neumann, en l'època dels anys 40, va buscar la resposta a:

Era teòricament possible dissenyar un robot que es repliqués a si mateix?

Coneguda com a autoreplicació, és la propietat d'un sistema dinàmic que produeix la construcció d'una còpia idèntica de si mateix. *Von Neumann* va compartir el problema amb un altre investigador, *Stanislaw Ulam*, qui va suggerir treballar amb una configuració de gelosia o reixeta [2]. En intentar modelar sistemes dinàmics van derivar a un marc de cel·les dintre de quadrícules; cada cel·la, amb múltiples estats possibles en funció de les cel·les veïnes. Així van néixer els autòmats cel·lulars.

Els **autòmats cel·lulars** (*cellular automata*) són una col·lecció de cel·les, amb estats, en una quadrícula de forma específica que evoluciona a través de diversos passos de temps discrets d'acord amb un conjunt de regles. L'estat de cada cel·la és defineix pel resultat

d'alguna funció de les cel·les veïnes. Per tant, es té una quadrícula de $m \times n$ cel·les on cada cel·la pot mostrar un de dos o més estats amb un conjunt definit de veïns que afecten l'estat a través d'un conjunt de regles. Partint d'aquesta premissa, els investigadors es van preguntar quin seria l'escenari més simple possible.

1.1.2 Autòmats cel·lulars elementals

Aquest escenari tan simple, anomenat posteriorment elemental, és va definir en els tres mòduls que es troben a qualsevol autòmat cel·lular [3].

- Una quadrícula de cel·les unidimensional
- Dos estats tal que seran el 0 i l'1 (blanc i negre, respectivament, tal com es pot observar a la *Figura 1*)
- Un conjunt de regles que té en compte només els veïns adjacents (esquerra/dreta)

Per tant, aquest autòmats cel·lulars elementals es definiran per ser unidimensionals, binaris i amb un conjunt de regles que agafa els estats dels veïns adjacents.

Figura 1: ECA amb cel·les binàries [0,1,0,1,0,1,0]

Per cada cel·la es tindran $2^3 = 8$ possibles configuracions ja que es tindrà en compte si l'estat és 0 o 1 en 3 cel·les consecutives tal com es pot observar a la *Figura 2*.

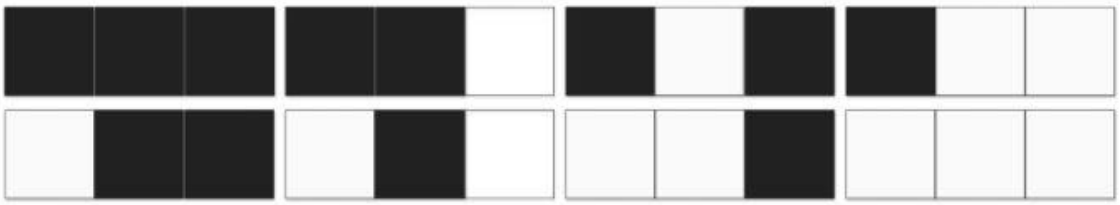


Figura 2: ECA amb les 8 configuracions possibles

Cadascuna d'aquestes 8 configuracions diferents generarà l'estat de la cel·la intermèdia que, al ser estats binaris, pot donar 0 o 1. Per tant, es tindrà $2^{2^3} = 256$ tipus diferents d'autòmats cel·lulars elementals. Stephen Wolfram va proposar un esquema, anomenat *Wolfram code,* per assignar cada regla un número del 0 al 255.

Wolfram va categoritzar cadascun d'aquests possibles autòmats cel·lulars en 4 classes diferents depenent del seu comportament. En la *Figura 3* es pot observar un exemple del comportament de cada classe.

- **Classe 1:** En gairebé tots els patrons inicials evolucionen ràpidament a un estat homogeni i estable. Qualsevol aleatorietat desapareix el patró inicial.

- **Classe 2:** En gairebé tots els patrons inicials evolucionen ràpidament a estructures estables o oscil·lants. Es pot filtrar part de l'aleatorietat en el patró inicial.
- **Classe 3:** En gairebé tots els patrons inicials evolucionen de manera pseudoaleatòria o caòtica. Qualsevol estructura estable que apareix es destrueix ràpidament.
- **Classe 4:** En gairebé tots els patrons inicials tendeixen a estructures complexes amb la formació d'estructures que poden sobreviure llargs períodes de temps. Els autòmats de classe 4 són capaços de computació universal, tot i que només s'ha pogut demostrar amb l'autòmat 110 i el *conway's Game of Life* [4].

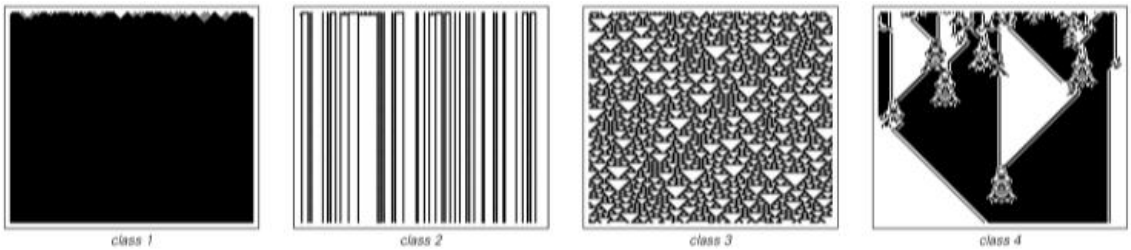


Figura 3: Exemple de les 4 classes [5]

Les regles dels autòmats cel·lulars elementals

Tal com s'ha explicat al punt anterior *Stephen Wolfram* va formalitzar 256 regles. Cada regla és defineix pel valor en binari de cada possible nou estat. Per exemple:

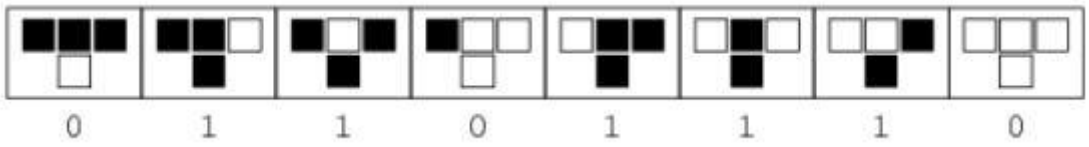


Figura 4: Regla 110

Partint de la *Figura 4* aquest nou estat serà el número en binari 0110 1110 que equival al número 110 en decimal, per tant s'haurà definit la regla 110.

Tot i que hi ha 256 regles, moltes d'elles són trivialment equivalents entre si i a partir de dues transformacions [6] veurem que ens quedaran 88 regles independents entre elles:

- La primera transformació serà la reflexió a través d'un eix central. El resultat d'aplicar aquesta transformació a una regla s'anomena regla reflectida. En total hi ha 64 regles equivalents a les seves regles reflectides.
- La segona transformació consisteix en intercanviar les funcions de 0 i 1 en la definició. El resultat d'aplicar aquesta regla s'anomena regla complementària. En total hi ha 16 regles que són equivalents a les seves regles complementàries.
- Si a una regla se li apliquen les dues transformacions s'obtindrà una regla complementària reflectida. En total hi ha 16 regles equivalents a les seves regles complementàries reflectides.

1.1.3 La regla 110

La regla 110 és una de les 256 regles descrites per Stephen Wolfram. Aquesta regla és particular amb una característica única que cap de les altres té o que encara no s'ha pogut demostrar.

- És un autòmat cel·lular elemental, un patró d'una dimensió de cel·les negres i blanques.
- És Turing complet [7] que, per definició, por simular una *màquina de Turing universal*.
- És molt similar i és compara amb el conegut Joc de la Vida dissenyat pel matemàtic *John Horton Conway* l'any 1970.

Per la definició del primer punt consistiria en una infinita fila de cel·les tal que $\{C_i \mid i \in \mathbb{Z}\}$. Cada cel·la pot ser un dels dos estats $\{0,1\}$ i per cada pas s'actualitza amb el valor de la funció de cada cel·la veïna tal que:

$$\forall i, C_i' = F(C_{i-1}, C_i, C_{i+i})$$

En la *Figura 5* es pot observar l'exemple d'un autòmat cel·lular elemental utilitzant la regla 110 en el pas de 15 iteracions i l'estat inicial de totes les cel·les blanques excepte la del punt mig que es negre.

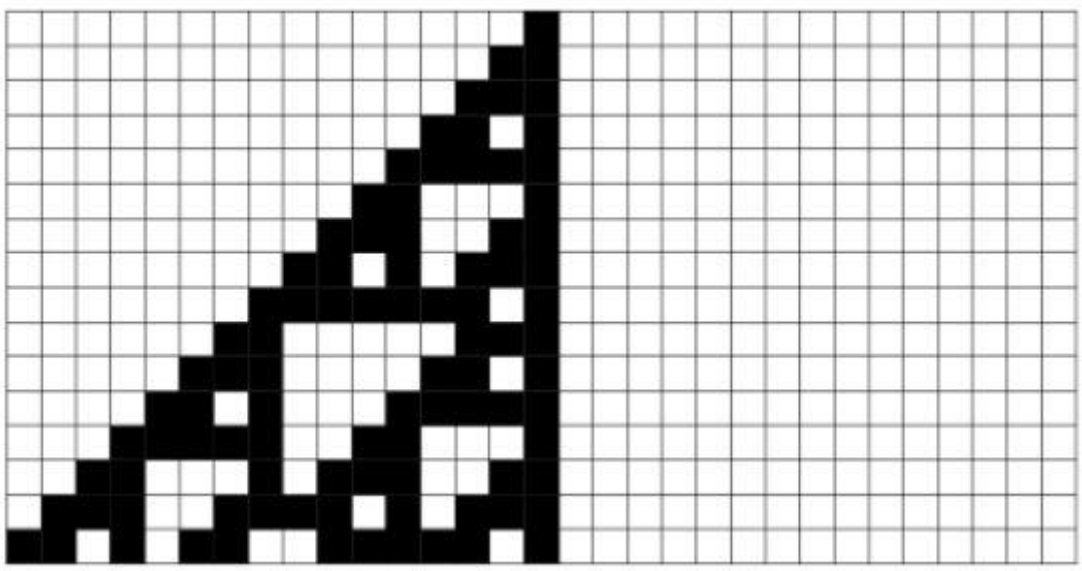


Figura 5: ECA 110 amb 15 iteracions partint d'una cel·la negra inicialment

1.1.4 Reservori

La computació de reservori és un framework que es distingeix en 3 capes: una d'entrada, el reservori que és la xarxa que s'ocupa de connectar els nodes entre si i, a vegades, entre ells i una de sortida [8] (*Figura 6)*.

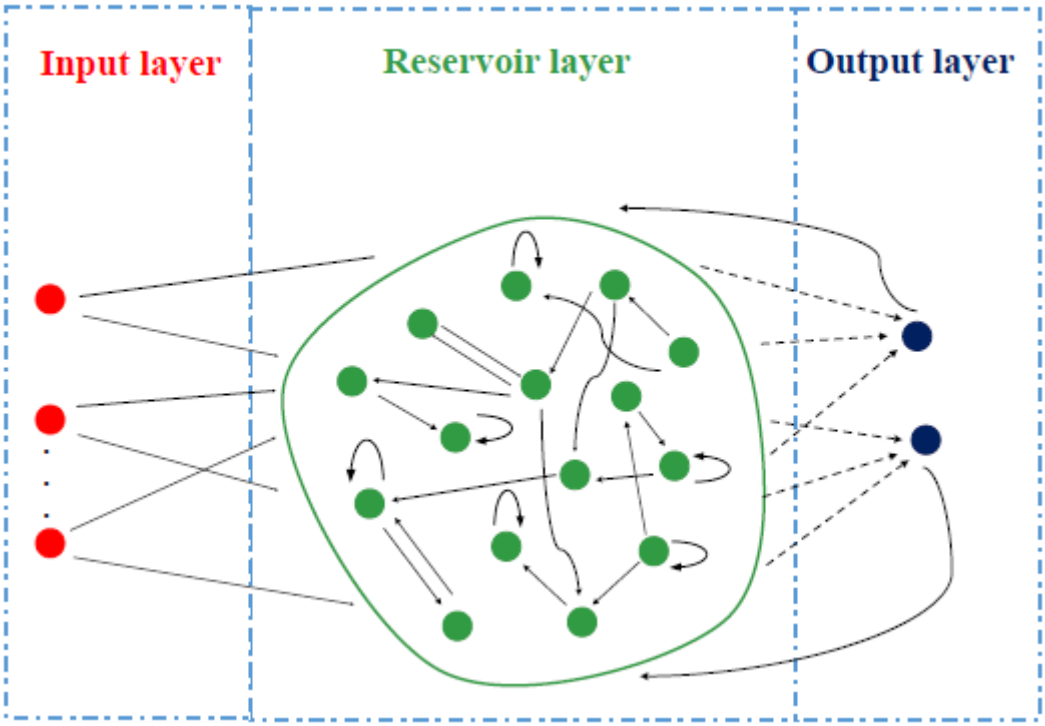


Figura 6: Les 3 capes que constitueixen l'estructura de la computació de reservori [9]

La diferencia amb altres tècniques d'aprenentatge automàtic és que tant l'entrada com el reservori és mantenen fixades i l'únic que s'actualitza són les connexions de sortida. La importància d'aquest mètode és que no es perd temps en millorar les connexions internes i no cal modificar l'estructura interna i, de vegades, no cal saber-la. A més a més, es poden fer varies feines en paral·lel.

Per a què aquesta tècnica funcioni s'han de complir 3 característiques: projecció d'una entrada d'una dimensió major, una funció no-lineal i memòria a curt termini:

- Per projectar una entrada amb moltes dimensions calen molts nodes per a que cadascun d'ells ens ofereixi una projecció diferent de la informació que estem introduint.

- Cal una funció no-lineal perquè si es tenen una sèrie de funcions lineals qualsevol combinació lineal de totes elles seria altre cop una funció lineal, però pels reservoris de computació calen comportaments més complexos.

- Cal memòria a curt termini perquè es necessita que la informació es quedi dintre de la xarxa i s'aconsegueix mitjançant connexions entre nodes.

Els reservoris de computació estan dividits en diferents tipus depenent d'algunes característiques que es modifiquen al voltant de la senyal d'entrada i dels nodes del reservori. Dels diferents models que es poden trobar se'n destacaran els 2 principals: Els E*cho State network* i els L*íquid State Machine*.

Els E*cho State Network (ESN)* [10] són un tipus de reservori de computació que utilitza el sistema de les xarxes neuronals recurrents que evita la no convergència i grans costos computacionals en els mètodes de descens de gradients. Consisteix en una senyal d'entrada, el reservori amb un gran número de neurones connectades entre elles i la senyal de sortida. Els pesos de la senyal d'entrada i del reservori es mantenen fixats i els pesos de la senyal de sortida es poden entrenar i s'obtenen a partir de resoldre problemes de regressió linear.

Els *Líquid State Machine (LSM)* [11] són un tipus de reservori de computació que utilitza el sistema de les xarxes neuronals d'impulsos [12]. El que fa que es diferenciï principalment de les altres xarxes neuronals es que l'entrada es presenta distribuïda en el temps i el model de la neurona almenys té una variable d'estat que representa el potencial de la neurona.

1.1.5 Base de dades MNIST

La base de dades MNIST (*Modified National Institute of Standards and Technology database*) és un gran conjunt de dígits escrits a mà (*Figura 7)* utilitzats normalment com a entrenament de problemes de classificació. El MNIST conté 60.000 imatges d'entrenament i 10.000 imatges per testejar [13].

Figura 7: MNIST dataset

1.2 Descripció del problema

En aquest treball es portarà a terme una investigació sobre l'ús de la regla 110 dels autòmats cel·lulars elementals com a reservori en problemes de classificació.

Actualment hi ha molts sistemes que intenten millorar o provar la classificació d'imatges i en el nostre treball volem centrar-nos en una nova línia d'investigació per analitzar els resultats.

1.3 Actors implicats

Aquest treball, al ser una experimentació d'un tema molt concret, els principals actors implicats seran el professor director del treball, Jordi Delgado, i l'alumne que portarà a terme l'experimentació, Ferran Esteban.

Altres actors implicats serien totes aquelles persones que treballin o investiguin en el món dels problemes de classificació i puguin trobar interès o utilitat als resultats obtinguts.

2 Justificació de l'alternativa de resolució escollida

L'objectiu d'aquest projecte és, tal com s'ha definit els apartats anteriors, crear, investigar i experimentar sobre problemes de classificació a partir d'una branca de les matemàtiques discretes.

La base d'aquest projecte serà dissenyar una nova línia d'investigació partint com a base els autòmats cel·lulars elementals. En els últims anys s'ha investigat sobre les diferents regles (ja mencionades anteriorment) i dels reservoris com a framework, però no hi ha gaire informació sobre utilitzar tots aquest recursos per a problemes de classificació [14].

Un dels articles que s'ha posat més èmfasi a l'hora de plantejar-nos portar a terme aquest projecte és el *Energy-Efficient Pattern Recognition Hardware with Elementary Cellular Automata* [15]. En aquest article detallen l'experimentació dels autòmats cel·lulars i els reservoris de computació amb els problemes de classificació, utilitzant el MNIST per fer les proves. Partint de les dades proposades i l'experiment final es portarà a terme un projecte propi on experimentarem utilitzant una regla en concret i poder comparar les conclusions dels experiments.

El segon article que hem posat èmfasi es el conegut *Reservoir Computing using Cellular Automata* [16] de l'*Ozgur Yilmaz*. Es fa referència aquest article perquè molts dels projectes i experiments dintre d'aquest camp utilitzen aquest experiment com a base per a portar a terme els seus propis experiments i poder comparar resultats. S'explica de manera detallada cada terme que s'ha explicat en el punt anterior i utilitza el problema 5-bit (tal com es farà en aquest projecte) per testejar.

En l'actualitat hi ha hagut molts projectes utilitzant el dataset MNIST com a base per investigar diferents mètodes i propietats i com afecten a l'hora d'utilitzar-los com a classificadors. Per això ens centrarem en els dos conceptes definits anteriorment:

- Regla 110 dels autòmats cel·lular elementals
- *Reservoir computing* (computació de reservori)

Per desenvolupar un sistema que resolgui problemes de classificació, posant èmfasi en el dataset *MNIST*.

3 Abast del projecte

3.1 Objectiu

L'objectiu principal és, tal com s'ha definit al principi del projecte, explorar l'ús de la regla 110 dels autòmats cel·lulars elementals com a reservori en problemes de classificació.

Per tant principalment es tractarà d'investigar en profunditat els autòmats cel·lulars elementals i la regla 110 i com podem utilitzar-la en l'apartat de computació de reservori.

3.2 Subobjectius

Els subobjectius seran els següents:

1. Es realitzarà tot el procés d'implementació del sistema del reservori amb la regla 110 dels autòmats cel·lulars elementals.
2. Testar i depurar el codi.
3. Definir quin o quins problemes de classificació hem de provar (com a mínim es testejarà el MNIST).
4. Fer els experiments i analitzar els resultats.

En funció dels resultats

5. Revisar el treball fet fins el moment i tornar a experimentar.

3.3 Requeriments

Alguns requeriments seran necessaris per definir un projecte de bona qualitat.

- Seleccionar i definir correctament els mètodes utilitzats per establir l'entrada (input) i sortida (output) dels problemes de classificació.
- Optimitzar el codi i utilitzar bones pràctiques de programació.

3.4 Obstacles i riscos

Com tot gran projecte ens podem trobar amb diferents obstacles i possibles riscos que haurem de tenir en compte a l'hora de portar a terme el projecte.

1. *El temps.* El temps sempre és un risc i un obstacle a l'hora de portar a terme projectes amb una data límit, tal com s'explicarà a l'apartat de planificació
2. *La falta de coneixements de l'alumne.* És essencial per portar a terme un projecte de dimensions considerables un bon coneixement del tema. En aquest cas l'autor del projecte parteix d'una base mínima en problemes de classificació, però parteix de 0 amb els reservoris i els autòmats cel·lulars.

3. *El covid*. Mai ens havíem trobat en una situació que fes parar tot el país, però avui en dia ens trobem en mig d'una pandèmia que està dificultant la mobilitat de la gent. Això causarà que les reunions hagin de ser a distància amb la dificultat que això comporta. Per altre banda i a causa de l'alta propagació es podria donar el cas de contagiar-se i estar-se dies sense poder avançar, dificultant la progressió del projecte i modificant les dates previstes a l'hora de planificar el projecte.

4 Metodologia i rigor

Aquest projecte es basa en una experimentació on haurem de fer diferents proves, avaluar resultats i seguir provant en un cicle, per tant buscarem una metodologia que ens permeti portar a terme aquest procés de la manera més òptima possible.

4.1 Metodologia àgil

La metodologia que ens servirà per aquest projecte són les anomenades metodologies àgil [17]. En particular utilitzarem la metodologia *Scrum* [18].

La metodologia *Scrum* és un framework per portar a terme projectes de llarga durada de manera iterativa, fragmentant-ho en sprints de curt termini i poder readaptar el contingut del projecte.

Figura 8: Fases de les metodologies ágil [19]

Per portar a terme aquest projecte aplicant una metodologia àgil es faran reunions continues amb el director del projecte on s'analitzarà l'estat actual del projecte i les modificacions que calguin per seguir amb l'experimentació.

Inicialment es portarà a terme una fase de lectura d'articles i autoaprenentatge. Fet aquest pas inicial, es planificaran tots els passos a fer per portar a terme la investigació. Després de la reunió amb el director del projecte es valoraran els possibles canvis a fer.

Per portar a terme al projecte de manera que puguem valorar a cada sprint que hem fet i que cal modificar utilitzarem el software *Git* [20] . Git permet fer un control de versions. En particular, utilitzarem *Github* [21].

5 Planificació temporal

L'objectiu d'aquest projecte en la planificació temporal és portar-lo a terme en el temps que comptabilitzem un quadrimestre. Un treball de final de grau consisteixen en 18 crèdits: 3 crèdits distribuïts en la part de *Gestió de projectes* i 15 crèdits més per al que seria el *Treball final de grau*.

L'inici del projecte va començar a principis de setembre, però a causa de que l'autor del projecte va haver de passar per una intervenció quirúrgica al canell va decidir posposar-lo fins a mitjans de gener per fer-lo al quadrimestre de primavera, per tant hem considerat el moment de l'inici de projecte l'11 de Gener del 2021.

Partint de la normativa de la FIB on indica que cada crèdit equival a 30 hores [22], ajustarem el treball per fer aleshores 450 hores i unes 90 hores per portar a terme la gestió del projecte.

La idea principal a l'hora de portar a terme aquest projecte és distribuir aquestes 450 hores durant els mesos que dura el quadrimestre, per tant s'espera fer 4 hores diàries els dies feiners. Això equival a 80 hores al mes i en 5 mesos a 400 hores. Per poder arribar a les 450 hores s'afegiran unes entre 2 o 3 hores als caps de setmana de manera planificada.

6 Descripció de les tasques

Les descripcions de les tasques són un component indispensable a l'hora de gestionar un projecte, per tant en aqueta secció s'explicarà de quina manera és gestionarà cada tasca i la seva durada.

6.1 Gestió del projecte

La gestió del projecte és una part indispensable a l'hora de planificar un projecte. Dividirem la gestió del projecte en diferents subapartats: contextualització i abast, planificació, pressupost, sostenibilitat i reunions.

- [T1] – Contextualització i abast

  Per contextualitzar el concepte i l'abast del projecte calen unes hores de recerca i investigació per ser capaç de donar una resposta elaborada i definir de manera que es justifiqui el propòsit del projecte.

  El tema proposat per portar a terme aquest projecte conté molta informació i calen hores de recerca, per tant s'han necessitat 25 hores.

- [T2] – Planificació temporal

  La planificació és una part imprescindible de qualsevol projecte i per portar-la a terme han calgut unes hores de reunió per decidir com ho farem i de quina manera tirarem endavant els *sprints*. Per la planificació s'han necessitat 15 hores.

- [T3] – Pressupost

  Es realitzarà un informe per calcular si caldrà algun hardware opcional i els softwares necessaris. S'estima unes 15 hores

- [T4] – Informe de sostenibilitat

  Es realitzarà un informe respecte com afectarà i quin serà el possible impacte del projecte. S'estima unes 15 hores.

- [T5] – Reunions

  Com tot projecte caldran reunions, i més en un projecte d'investigació on cal avaluar els experiments de manera regular. S'estima unes 20 hores

6.2 Aprenentatge i lectura d'articles – Sprint 1 [SP1]

En aquesta segona fase i després d'haver estudiat de manera superficial les principals característiques és portarà a terme una fase d'autoaprenentatge de manera exhaustiva i es llegirà i discutirà amb el tutor del projecte alguns articles.

Cada article conté molta informació, per això s'ha decidit separar la lectura de cadascun en diferents tasques per poder analitzar-los correctament.

- [T6] – Reservoir computing using cellular automata

  Lectura i discussió de l'article *"Reservoir Computing using Cellular Automata"* de l'*Ozgur Yilmaz* [16]. S'estima unes 20 hores.

- [T7] – Investigation of Elementary Cellular Automata for Reservoir Computing

  Lectura i discussió de l'article "Investigation of Elementary Cellular Automata for Reservoir Computing" de l'*Emil Tailor Bye* [23]. S'estima unes 20 hores.

- [T8] - Feed-forward versus recurrent architecture

  Lectura i discussió de l'article "*Feed-forward versus recurrent architecture and local versus cellular automata distributed representation in reservoir computing for sequence memory learning*" del *Mrwan Margem* i *Osman S. Gedik* [24]. S'estima unes 10 hores.

- [T9] – Long Short-Term Memory in Echo State Networks: Details of a Simulation Study

  Lectura i discussió de l'article "*Long Short-Term Memory in Echo State Networks: Details of a Simulation Study*" del *Herbert Jaeger* [25]. S'estima unes 10 hores.

- [T10] – Altres articles

  S'espera que a la tercera fase es provi de mirar de fer més experiments si el MNIST ens dona bons resultats, per tant s'assignarien unes hores més a la lectura d'algun article necessari. S'estima unes 20 hores.

6.3 Implementació i proves – Sprint 2 [SP2]

En aquesta segona fase es portarà a terme tot el gruix del projecte on s'implementarà tot el codi necessari.

- [T11] – Preparació

  La Primera fase on prepararem l'estructura del projecte: software i hardware necessari. S'estima unes 10 hores

- [T12] – Implementació

  La segona fase serà la de desenvolupar els algorismes necessaris que ens permetin començar a fer proves. S'estima 40 hores.

- [T13] – Proves

  A partir de que ja tenim un sistema implementat es realitzaran una sèrie de proves per assegurar-nos de que funciona correctament. S'utilitzaran *benchmarks* habituals com el *5-bit problem* (T8). S'estima unes 30 hores.

- [T14] – Reunió del sprint

  Al final de les proves s'avaluarà amb el cap de projecte el treball fet i és discutirà si cal fer cap modificació. S'estima 10 hores

- [T15] – Documentació

  Redactar la documentació del codi i les proves generades. S'estima 20 hores.

6.4 Experimentació – Sprint 3 [SP3]

La fase final del projecte és experimentar amb diferents problemes i estudiar els resultats. Inicialment només és tractaria d'atacar el problema del MNIST, però depenent dels resultats donaríem el vistiplau a altres problemes.

- [T16] – Proves al MNIST

  Fase final on provarem el problema MNIST i detallarem els resultats. S'estima 40 hores.

- [T17] – Reunió de l'sprint

  Reunió per fer una avaluació final del problema i decidir si es proven altres problemes. S'estima 10 hores.

- [T18] – Proves opcionals a altres problemes.

  Tasca opcional on es provarien altres problemes que siguin factibles per l'experimentació. S'estima 40 hores

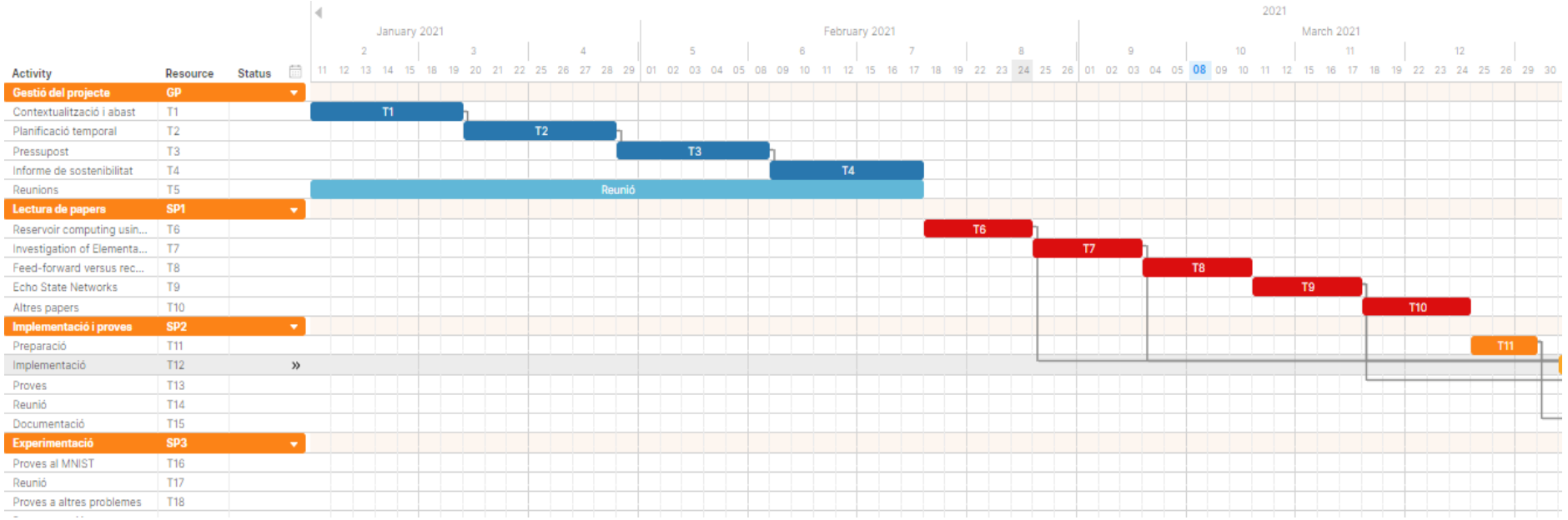
- [T19] – Documentació

  Redactar tota la investigació i experiments fets. S'estima 30 hores.

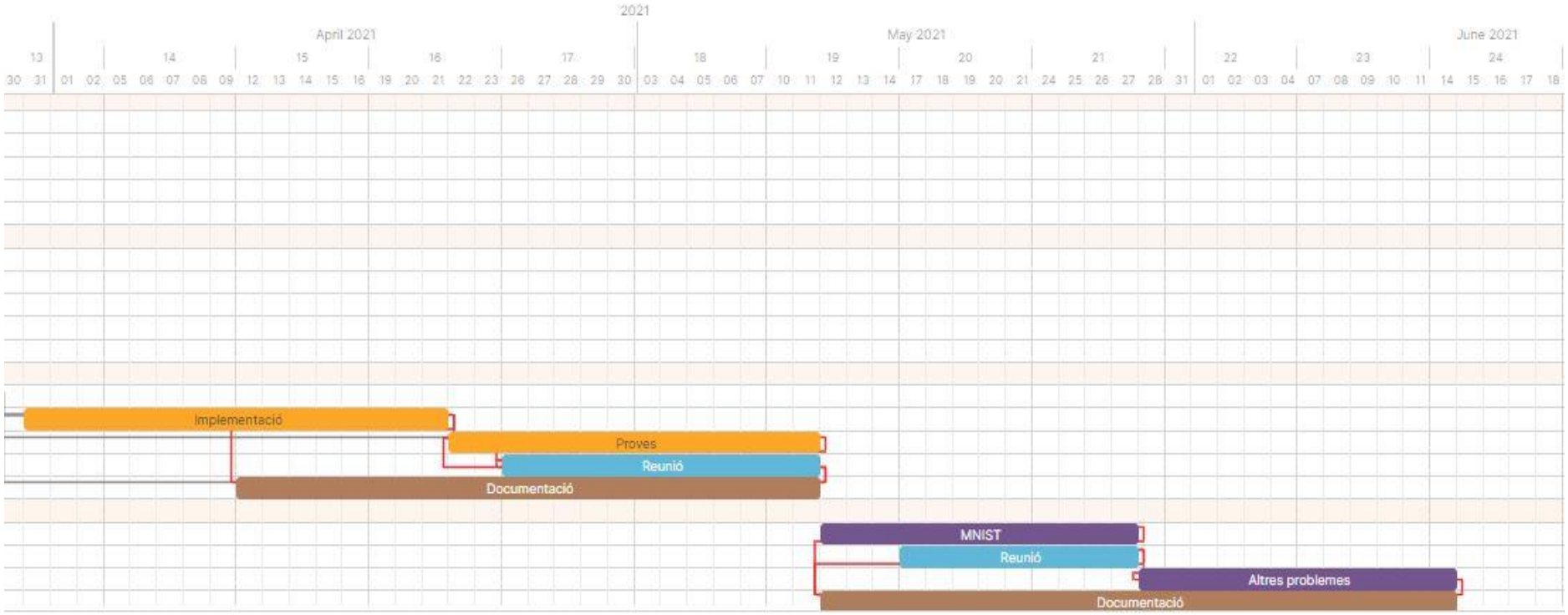
En la *Taula 1* podem veure cada tasca i les dependències entre elles.

Id	Tasca	Temps estimat (*h*)	Dependències
GP	**Gestió del projecte**	**90**	**-**
T1	Contextualització i abast	25	-
T2	Planificació temporal	15	T1
T3	Pressupost	15	T2
T4	Informe de sostenibilitat	15	T3
T5	Reunions	20	-
SP1	**Lectura d'articles**	**100**	**-**
T6	Reservoir computing using CA	20	**-**
T7	Investigation of Elementary Cellular Automata	20	-
T8	Feed-forward versus recurrent architecture	20	-
T9	Echo State Networks	20	-
T10	Altres articles	20	-
SP2	**Implementació i proves**	**140**	**-**
T11	Preparació	10	T6, T7
T12	Implementació	60	T6, T7, T8, T11
T13	Proves	40	T9, T12
T14	Reunió	10	T13
T15	Documentació	20	T11, T12, T13, T14
SP3	**Experimentació**	**120**	
T16	Proves al MNIST	40	T13, T14
T17	Reunió	10	T16
T18	Proves a altres problemes	40	T17
T19	Documentació	30	T16, T17, T18

Taula 1: Distribució de tasques



Taula 2: esquema de Gantt (part 1)



Taula 3: esquema de Gantt (part 2)

7 Gestió del risc: Plans alternatius i obstacles

Com tot projecte, cal analitzar i valorar els possibles risc que podem trobar a l'hora de portar-lo a terme. A continuació analitzarem quins risc tenim i com prevenir-los.

1. *El temps* – Com ja havíem dit és un risc important a l'hora de portar a terme un gran projecte. Al ser un projecte d'investigació i la idea principal és el problema MNIST, si s'anés just de temps altres problemes de classificació es descartarien.
2. *Falta de coneixements* – Algunes tasques s'han afegit més hores per donar temps a entendre o a buscar més informació en cas que fos necessari.
3. *El covid* – El coronavirus ens està afectant a tots a l'hora d'haver de mobilitzar-nos i tot el que siguin reunions o anar a biblioteques s'ha dificultat molt més. Per evitar qualsevol situació de risc s'han afegit més hores per si l'alumne o el director estiguessin indisponibles uns dies i preparat softwares per poder fer reunions de manera telemàtica.

8 Pressupost

Un cop finalitzat el procés de fer la gestió del projecte i planificar cada tasca i les hores necessàries es calcularà quin serà el cost per portar-les a terme.

8.1 Identificació i estimació dels costos

En qualsevol projecte on es requereixen una sèrie de recursos tant tecnològics com humans que cal identificar i calcular les despeses necessàries per poder desenvolupar-lo correctament.

8.1.1 Recursos humans

La primera part a l'hora d'identificar el pressupost serà investigar el salari necessari per gestionar els treballadors d'aquest projecte. En aquest projecte participen 2 persones: el director i, per tant, cap de projecte i l'alumne que serà l'encarregat d'investigar i programar.

Segons la web *glassdoor* [26] el sou d'un cap de projecte està al voltant d'uns 47.000 – 48.000€ l'any que serien uns 23€/h. Per la pàgina web de *indeed* hem buscat el sou mitjà d'un programador [27] i estaria al voltant d'uns 27.500€ l'any que serien uns 13,22€/h.

En la *Taula 4* podem veure totes les dades respecte els sous.

Rol	**Sou brut (€/h)**	**Retribució Seguretat Social**
Cap de projecte	23,00	30,00
Programador	13,50	17,55

Taula 4: Sou brut i retribució a la seguretat social per rol del projecte

Ara que ja es tenen els sous per cada rol implicat en aquest projecte es pot calcular el cost total de cada tasca i el cost final

Id	Tasca	Temps estimat (*h*)	Cap de projecte (€)	Programador (€)
GP	**Gestió del projecte**	**90**	**2070**	**270**
T1	Contextualització i abast	25	575	-
T2	Planificació temporal	15	345	-
T3	Pressupost	15	345	-
T4	Informe de sostenibilitat	15	345	-
T5	Reunions	20	460	270
SP1	**Lectura d'articles**	**100**	**-**	**1350**
T6	Reservoir computing using CA	20	-	270
T7	Investigation of Elementary Cellular Automata	20	-	270
T8	Feed-forward versus recurrent architecture	20	-	270
T9	Echo State Networks	20	-	270
T10	Altres articles	20	-	270
SP2	**Implementació i proves**	**140**	**230**	**1890**
T11	Preparació	10	-	135
T12	Implementació	60	-	810
T13	Proves	40	-	540
T14	Reunió	10	230	135
T15	Documentació	20	-	270
SP3	**Experimentació**	**120**	**230**	**1620**
T16	Proves al MNIST	40	-	540
T17	Reunió	10	230	135
T18	Proves a altres problemes	40	-	540
T19	Documentació	30	-	405
Total		**450**	**2530**	**4860**

Taula 5: Cost de cada tasca i cost final dels recursos humans

8.1.2 Recursos materials

Ara que ja s'han assignat els sous necessaris per les persones implicades en el projecte, es calcularà els recursos materials.

Inicialment ja es tenia un ordinador de sobretaula per portar a terme el projecte de manera eficient, però seguint la normativa d'Hisenda el qual el hardware es permet utilitzar en 3 – 4 anys caldrà demanar un nou ordinador.

Per aquest projecte s'ha decidit utilitzar l'ordinador de sobretaula un HP OMEN OBELISK DT875 per a l'oficina i un portàtil HP 255 G8. El gruix d'hores s'utilitzarà a l'ordinador de sobretaula ja que majoritàriament la feina es farà a casa. Pel portàtil s'han assignat 70 hores.

Per calcular l'amortització seguirem la fórmula:

$$\frac{CostDispositiu}{VidaÚtil * 220(dies\ hàbils) * 4(hores\ dedicació)} * HoresDedicació$$

Hardware	Preu (€)	Vida útil (en anys)	Amortització (€)
Ordinador	1.065,34	4	115,00
Portàtil	520,30	4	10,34

Taula 6: Costos materials

8.1.3 Recursos indirectes

Per l'apartat de recursos indirectes partirem de la suposició que la investigació es porta a terme en una oficina de treball. Per buscar un lloc adient hem decidit utilitzar *aurea* [28]. En aquest espai de treball treballarà 1 persona utilitzant els recursos materials mencionats a la *Taula 6* i amb serveis d'electricitat, calefacció, aire condicionat, internet, una sala de reunions i l'accés durant els 7 dies de la setmana. Per tant, s'ha decidit contractar el *Full Access* on tindrem tots aquest serveis coberts.

Servei	Preu/mes +IVA (€)	Temps (en mesos)	Total (€)
Oficina coworking	210+44,10	4	1017

Taula 7: Costos indirectes

8.1.4 Contingències

Per aquest projecte d'investigació on només treballarà una persona a temps complert i un parell d'ordinadors s'ha decidit que hi hagi un sobre cost de contingència d'un 10%

Costos de recursos	Cost (€)	Contingència	Cost contingència (€)
Personals	7.390	10%	739
Materials	1.585,64	10%	176,56
Indirectes	1.017	10%	101,7

Taula 8: Costos per contingències

8.1.5 Imprevistos

Per aquest projecte pocs imprevistos haurien d'haver-hi ja que poca gent hi participa i tampoc hi ha massa material, però tal com s'ha analitzat en la gestió de riscos ens trobem en situació de pandèmia i podria haver-hi la possibilitat que hi hagués alguna baixa per coronavirus [29], per tant s'haurien de retardar els lliuraments afegint hores al final. El càlcul sortiria per unes 20 hores de més amb una possibilitat del 10%.

Una altre possibilitat seria que els recursos material s'espatllessin i haver de canviar-los. S'estima la possibilitat en un 5%.

Imprevistos	Cost (€)	Risc	Cost imprevistos (€)
Augment hores	270,00	10%	27,00
Ordinador	1.065,34	5%	53,27
Portàtil	520,30	5%	26,01

Taula 9: Costos imprevistos

8.1.6 Pressupost final

Ara que ja s'ha calculat tots els possibles costos podem fer un pressupost final a la *Taula 10*.

Tipus de cost	Cost
Personal	7.390
Material	1.585,34
Indirectes	1.017,00
Contingència	1.017,26
Imprevistos	106,28
Total	11.215,88

Taula 10: Costos totals

8.2 Control de gestió

A fi de poder gestionar d'una manera més eficient qualsevol desviació en el pressupost final al moment que es finalitzi una tasca s'anotaran els gestos extres i es compararan amb els calculats al pressupost. Per portar a terme aquest càlculs s'utilitzaran les següents fórmules:

- Desviació hores consumides per tasca

$$(Hores\_estimades - Hores\_reals) * Cost\_estimat$$

- Desviació dels costos en recursos humans per tasca:

$$(Cost\_estimat - Cost\_real) * Hores\_reals$$

- Desviació total dels recursos materials

$$Cost\_material\_estimat - Cost\_material\_real$$

- Desviació total dels recursos indirectes:

$$Cost\_indirecte\_estimat - Cost\_indirecte\_real$$

- Desviació total dels imprevistos:

$$Cost\_imprevistos\_estimat - Cost\_imprevistos\_total$$

- Desviació total d'hores

$$Hores\_estimades - hores\_reals$$

9 Desenvolupament

En aquest apartat es profunditzarà més en alguns conceptes dels autòmats cel·lulars elementals i els reservoris i com es pot implementar a partir d'algorismes mencionats en els articles de l'apartat 6.2.

9.1 Autòmats cel·lulars elementals com a reservori

En l'*estat de l'art* s'ha explicat que són els autòmats cel·lulars i els reservoris computacionals i com funcionen, però en aquest projecte es treballarà la utilització d'autòmats cel·lulars com a reservoris.

El primer en estudiar aquest sistema i desenvolupar un algorisme funcional va ser l'*Ozgur Yilmaz* en l'article mencionat a l'apartat 6.2 "*Reservoir computing using Cellular automata"* [16] a l'any 2014. La idea d'utilitzar aquest sistema és que les neurones del reservori siguin cada estat de cada entrada de l'autòmat. La nomenclatura utilitzada és un sistema ***ReCA*** *(Reservoir Cellular Automata)*.

Anteriorment s'ha explicat com el sistema estava dividit en 3 parts: la senyal d'entrada, el reservori (en aquest cas seran els autòmats cel·lulars elementals) i la senyal de sortida tal com es pot observar a la *Figura 9*. En següents apartats es detallarà quins mecanismes s'utilitzaran per l'entrada, com s'enviaran les dades a l'autòmat i la manera en que es treballaran els pesos de la senyal de sortida ja que per fer el test i tenir un algorisme funcional s'utilitzarà el sistema de l'*Ozgur* detallat en el seu article i per treballar amb les imatges del MNIST s'utilitzarà unes modificacions.

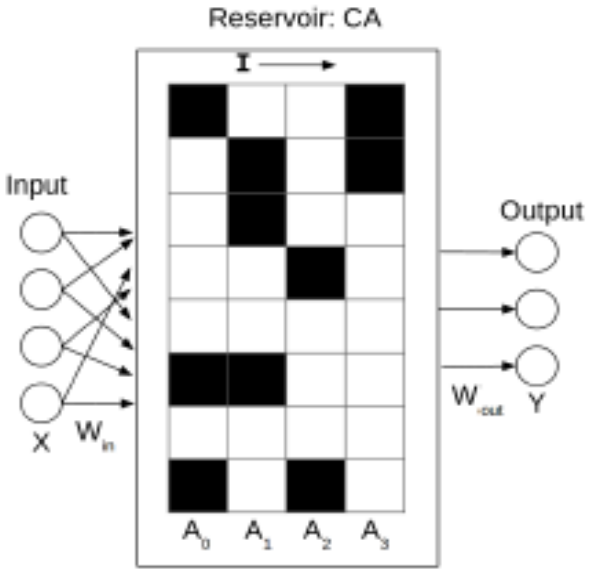


Figura 9: Sistema ReCA amb la senya d'entrada, el reservori i els pesos en la senyal de sortida

10 Problema del 5 bit

El problema del 5 bit va ser descrit per *Hochreiter* i *Schmidhuber* l'any 1997 [30] i és característic per ser un problema difícil de resoldre amb xarxes neuronals recurrents [31].

10.1 Introducció al problema del 5 bit

El sistema està definit per 4 variables d'entrada i 3 de sortida (en algun article es parlen de 4 variables de sortida, però no es té en compte en cap sistema ja que el valor de l'última variable és sempre 0). Els valors poden ser 0 o 1 en el pas del temps i complint una sèrie de regles. Les variables d'entrada es defineixen com $a_1, a_2, a_3 \; i \; a_4$ i les variables de sortida com $y_1, y_2 \; i \; y_3$

El problema del 5 bit es divideix en 3 blocs. El primer bloc són els 5 primers passos en el temps on es defineix el patró inicial del problema. De manera aleatòria l'$a_1 \; i \; l'a_2$ seran 1 o 0 respectivament i $l'a_3 \; i \; l'a_4$ seran 0 en el pas del temps. La sortida per aquest 5 passos per l'$y_1 \; i \; y_2$ serà 0 i $l'y_3$ serà 1. En la T*aula 11* es pot veure un exemple d'una possible entrada.

n	Input				Output		
	a_1	a_2	a_3	a_4	y_1	y_2	y_3
1	**1**	**0**	0	0	0	0	1
2	**1**	**0**	0	0	0	0	1
3	**0**	**1**	0	0	0	0	1
4	**0**	**1**	0	0	0	0	1
5	**1**	**0**	0	0	0	0	1

Taula 11: Patró inicial del 5-bit

Els valors en vermells són els bits que conformen el patró inicial ja que són els únics bits que tenen un valor diferent de manera aleatòria. L'a_3 i l'a_4 no tenen importància en aquest bloc ja que sempre seran 0 per qualsevol combinació d'$a_1 \; i \; a_2$.

Com l'entrada dels 2 primers bits és aleatòria i com els valors d'a_1 i a_2 no poden ser iguals en el mateix pas de temps es pot arribar a tenir un màxim de 32 patrons ja que en aquest 5 passos (on s'ha definit una variable n per agilitzar la lectura dels passos en el temps) poden ser fins a 2^5 combinacions diferents per la totalitat de l'entrada.

El segon bloc se l'anomena "distractor" i són tota una sèrie de passos idèntics tots ells excepte l'últim pas, anomenat *cua* que indica el final del distractor. Per aquest distractor els valors d'entrada per $a_1, a_2 \; i \; a_4$ seran 0 i el valor de a_3 serà 1 i els valors de sortida per y_1, y_2 seran 0 i el valor de y_3 serà 1. En l'últim pas del distractor, la cua, els valors de sortida es mantenen igual i els valors d'entrada es modifiquen quedant $l'a_1, a_2 \; i \; a_3$ en 0 i $l'a_4$ en 1.

En la *Taula 12* es pot veure un exemple on el valor del distractor t_d és 15 i l'últim pas del distractor és la cua amb el valor $a_4 = 1$. Arribats a aquest punt es tindrien 20 passos en el temps.

n	Input				Output		
	a_1	a_2	a_3	a_4	y_1	y_2	y_3
1	**1**	**0**	0	0	0	0	1
2	**1**	**0**	0	0	0	0	1
3	**0**	**1**	0	0	0	0	1
4	**0**	**1**	0	0	0	0	1
5	**1**	**0**	0	0	0	0	1
6	0	0	**1**	0	0	0	1
7	0	0	**1**	0	0	0	1
8	0	0	**1**	0	0	0	1
...	...				...		
19	0	0	**1**	0	0	0	1
20	0	0	0	**1**	0	0	1

Taula 12: Distractor i cua del 5 bit per 15 passos

El tercer i últim bloc es defineix la sortida com la còpia idèntica del patró inicial vist en l'entrada del bloc 1 tenint en compte els bits remarcats en vermell de la *Taula 11*, per tant en cadascuns dels 5 passos en els bits d'entrada (a_1 *i* a_2) del patró inicial resultaran en el patró de sortida en els bits y_1 i y_2. El valor de y_3 serà 0 i els valors de l'entrada seran idèntics als valors vist al distractor exceptuant el valor de la cua en tots els passos del bloc final. En la *Taula 13* es pot veure un exemple del bloc final.

n	Input				Output		
	a_1	a_2	a_3	a_4	y_1	y_2	y_3
20	0	0	0	**1**	0	0	1
21	0	0	1	0	**1**	**0**	0
22	0	0	1	0	**1**	**0**	0
23	0	0	1	0	**0**	**1**	0
24	0	0	1	0	**0**	**1**	0
25	0	0	1	0	**1**	**0**	0

Taula 13: Bloc final amb els bits de sortida com el patró inicial

En aquest últim exemple es pot veure com els valors de sortida són idèntics als valors d'entrada que es poden en la *Taula 11*. Els bits d'entrada tornen a seguir l'estructura del *distractor* (sense tenir en compte l'últim pas en el temps, la *cua*).

En la T*aula 14* es pot veure com queda el resultat final d'ajuntar les 3 taules.

	n	Input				Output		
		a_1	a_2	a_3	a_4	y_1	y_2	y_3
Patró inicial	1	1	0	0	0	0	0	1
	2	1	0	0	0	0	0	1
	3	0	1	0	0	0	0	1
	4	0	1	0	0	0	0	1
	5	1	0	0	0	0	0	1
Distractor	6	0	0	1	0	0	0	1
	7	0	0	1	0	0	0	1
	8	0	0	1	0	0	0	1
	...	...				...		
	19	0	0	1	0	0	0	1
Cua	20	0	0	0	1	0	0	1
Sortida del patró inicial	21	0	0	1	0	1	0	0
	22	0	0	1	0	1	0	0
	23	0	0	1	0	0	1	0
	24	0	0	1	0	0	1	0
	25	0	0	1	0	1	0	0

Taula 14: Exemple d'un cas del problema 5 bit

Donat l'exemple de la *Taula 14* on el *distractor* és 15, s'obté el patró inicial amb els 5 primers passos, marcat en color blau; el *distractor*, juntament amb el seu últim pas en el temps, la *cua*, marcats en verd i vermell respectivament; i la sortida amb la còpia del patró inicial marcat en blau.

Aquest seria un dels 32 possibles exemples del problema 5-bit on es trobaria des del cas que el bit $a_1 = 1$ i el bit $a_2 = 0$ són idèntics fins al cas $a_1 = 0$ i $a_2 = 1$ en els 5 primers passos en el temps respectivament.

El nombre de passos en el temps el qual es mostrarà el *distractor* serà l'únic paràmetre que tindrà el problema. Per provar l'eficàcia del sistema s'afegeixen valors més elevats i s'analitzen quants casos és capaç de reconèixer al 100% el sistema. Per donar per vàlid un reconeixement l'algorisme ha de ser capaç de predir amb exactitud la sortida

completa vista en la *Taula 14* tenint en compte tots els valors dels tres blocs en conjunt i no només els de l'últim bloc.

10.2 Implementació del problema 5 bit

La implementació del problema del 5 bit s'ha portat a terme en el llenguatge de programació *Python*, exactament s'ha utilitzat la versió 3.9 [32] i utilitzant alguns dels paquets disponibles com *Numpy* [33] pel tractament dels valors i *Scikit-learn* [34] per portar a terme l'entrenament de la senyal de sortida.

La idea d'utilitzar *Python* en aquest projecte no era la més òptima en termes d'eficiència ja que hi ha altres llenguatges, com per exemple el llenguatge *C* que són molt més ràpids [35], però per una altre banda *Python* ofereix unes eines per estructurar les dades i l'entrenament de l'autòmat de manera més clara i fàcil amb la idea d'agilitzar la generació de codi i dedicar-li més temps als experiments posteriors.

Tal com s'ha comentat anteriorment la computació de reservori està dividida en 3 blocs. La senyal d'entrada, el reservori i la senyal de sortida. L'algorisme implementat estarà dividit en 3 blocs similars per poder diferenciar clarament les dades de cadascun.

10.2.1 La senyal d'entrada

La senyal d'entrada es pot diferenciar en dues estratègies diferents. Generar tots els casos del 5 bit i afegir-los a l'entrada o treballar cas per cas individualment.

Per aquest projecte s'ha decidit treballar, inicialment[1], cas per cas de manera individual al reservori. Per generar els 32 casos del 5 bit es crearà un algorisme que generi cada cas partint del primer cas on $a_1 = 1$ i $a_2 = 0$ en els 5 primers passos en el temps fins l'últim cas on $a_1 = 0$ i $a_2 = 1$. Per generar una estructura de dades amb el 5 bit cal passar-li un paràmetre t_d per definir la quantitat de *distractors* que es volen generar.

Tal com es pot observar en la *Taula 15* quedarien 32 taules amb tots els possibles valors agrupats amb un $t_d = 15$.

El sistema s'inicialitzarà amb tots els valors de cada fila del cas escollit tenint fins a un total de $t_d + 10$ entrades (5 passos inicials + t_d + 5 passos finals).

[1] A partir de les dades i el temps es valorarà l'altre opció

q	n	Input				Output		
		a_1	a_2	a_3	a_4	y_1	y_2	y_3
1	1	1	0	0	0	0	0	1
	2	1	0	0	0	0	0	1
	3	1	0	0	0	0	0	1
	4	1	0	0	0	0	0	1
	5	1	0	0	0	0	0	1
	6	0	0	1	0	0	0	1
	7	0	0	1	0	0	0	1
	8	0	0	1	0	0	0	1
	...	...				...		
	19	0	0	1	0	0	0	1
	20	0	0	0	1	0	0	1
	21	0	0	1	0	1	0	0
	22	0	0	1	0	1	0	0
	23	0	0	1	0	1	0	0
	24	0	0	1	0	1	0	0
	25	0	0	1	0	1	0	0

• • •

q	n	Input				Output		
		a_1	a_2	a_3	a_4	y_1	y_2	y_3
32	1	0	1	0	0	0	0	1
	2	0	1	0	0	0	0	1
	3	0	1	0	0	0	0	1
	4	0	1	0	0	0	0	1
	5	0	1	0	0	0	0	1
	6	0	0	1	0	0	0	1
	7	0	0	1	0	0	0	1
	8	0	0	1	0	0	0	1
	...	...				...		
	19	0	0	1	0	0	0	1
	20	0	0	0	1	0	0	1
	21	0	0	1	0	0	1	0
	22	0	0	1	0	0	1	0
	23	0	0	1	0	0	1	0
	24	0	0	1	0	0	1	0
	25	0	0	1	0	0	1	0

Taula 15: Els 32 casos del problema 5-bit

El pas final per concloure amb l'entrada seria passar-li un dels casos al reservori. Per veure com evolucionen els diferents casos del 5 bit s'ha generat un bucle per veure els resultats dels 32 casos de manera paral·lela i comparar millor els resultats amb altres paràmetres que s'explicaran en el següent apartat.

10.2.2 El reservori

El reservori consisteix en una sèrie d'autòmats cel·lulars amb un conjunt de cel·les cadascun que seran els nodes del reservori. Aquestes cel·les de manera inicial estaran buides i s'afegiran els possibles 1 o 0 depenent de si rep l'entrada del problema o els estats següents de l'autòmat.

El primer pas al reservori s'anomena *Encode stage* on es codificarà l'entrada a partir de 2 paràmetres que se li passaran al sistema: els paràmetres *R* i *C*. El paràmetre *R* indicarà la quantitat de vegades que es maparà l'entrada de manera aleatòria i el paràmetre *C* indicarà el número que s'utilitzarà per multiplicar la mida del vector. Se li podria passar un valor qualsevol i no multiplicar res, però així queda més fàcil per automatitzar el mapeig i no quedi vectors de mida diferents.

En la *Figura 10* es pot observar un exemple de l'article *Reservoir Computing Using Non-Uniform Binary Cellular Automata* [36] amb els paràmetres $R = 2$ i $C = 2$ amb el primer pas de l'entrada X_0. La mida de l'autòmat serà inicialment la mida de l'entrada, que serà el valor 4 (els 4 bits de cada pas en el temps) multiplicat pel valor C que com és 2 la mida serà de 8 cel·les. De manera aleatòria s'afegiran els 4 bits a les cel·les de l'autòmat i les cel·les que no tinguin cap bit automàticament seran 0. Aquest pas es repetirà R vegades. Com R és 2 es tindrà 2 autòmats de mida 8. L'últim pas de la codificació és concatenar els 2 autòmats quedant un sol autòmat de mida 16: X_0^P.

$$X_0^p = [X_0^{P_1}; X_0^{P_2}; \dots; X_0^{P_R}]$$

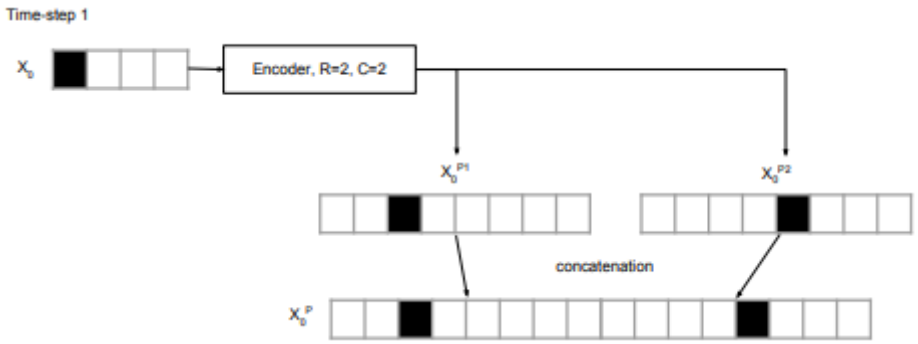


Figura 10: Entrada amb els paràmetres R i C

Tal com es pot veure poden quedar cadenes més o menys llargues depenent d'una sèrie de paràmetres que es consideren de manera inicial i que s'utilitzen per analitzar els resultat de l'algorisme emprat i com d'efectiu és utilitzar els autòmats com a reservori en problemes d'aquestes característiques.

El següent pas és desenvolupar l'autòmat amb la regla 110, tal com s'ha explicat en l'estat de l'art del projecte. L'autòmat començarà en la posició 0, agafarà els valors de la posició actual, anterior i posterior i aplicarà la regla creant un nou autòmat de la mateixa mida però amb els nous valors. En la *Taula 16* es pot observar com quedarà l'autòmat després d'haver aplicat la regla 110 una vegada.

X_0^P	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0
A_0	0	1	1	0	0	0	0	0	0	0	0	1	1	0	0	0

Taula 16: Nou vector després d'aplicar la regla al vector inicial

La regla 110 es pot aplicar tantes vegades calguin o de la manera que es vulgui provar l'eficàcia del sistema donant-li més o menys informació. Per tant per una funció *F* que indica l'aplicació de la regla:

$$A_0^0 = F(X_0^P)$$

$$A_1^0 = F(A_0^0)$$

$$A_2^0 = F(A_1^0)$$

$$\dots$$

$$A_I^0 = F(A_{I-1}^0)$$

El paràmetre *I* indicarà el número d'iteracions que l'autòmat portarà a terme. Com tota la informació dels nodes interessa per la senyal de sortida s'emmagatzemarà la informació en un vector de manera concatenada:

$$A^0 = [A_0^0; A_1^0; A_2^0; \dots; A_I^0]$$

Aquesta informació és el que se li acaba enviant a la senyal de sortida amb la variable *W*. Tots aquests passos de l'algorisme han sigut per la primera entrada, per a les següents entrades es portarà a terme una modificació.

L'algorisme procedeix de la mateixa manera inicialment passant-li l'entrada a l'autòmat, mapant *R* vegades i amb la mida del vector multiplicada pel valor *C*. La diferència radica en l'últim pas abans d'aplicar la regla 110.

Aquest pas, anomenat *Fase de transició*, es basa en mirar bit a bit la cel·la de la nova entrada i el de la cel·la de l'últim pas de l'autòmat anterior i es decideix el valor final d'aquest nou autòmat de la manera següent:

- Si la suma dels 2 valors és 2, el valor de la cel·la serà 1
- Si la suma dels 2 valors és 0, el valor de la cel·la serà 0
- Si la suma dels 2 valors és 1, el valor de la cel·la serà 1 o 0 de manera aleatòria.

Un altre manera que es pot fer i s'implementarà en el sistema del MNIST és amb una operació XOR. D'aquesta manera es manté certa informació de cada pas i no es perd totalment en cada entrada de l'autòmat.

Amb l'autòmat ja definit es pot aplicar la regla 110 i crear una nova sèrie d'autòmats cel·lulars. L'algorisme seguirà aplicant-se fins arribar a l'últim pas del paràmetre I.

Per tant, de manera resumida es te un algorisme que agafa la primera entrada, el modifica partir d'uns paràmetres *R* i *C* i li aplica la regla 110 tantes vegades el paràmetre I indiqui. La resta d'entrades es fa de la mateixa manera tenint en compte l'últim autòmat.

10.2.3 La senyal de sortida

La senyal de sortida, d'una manera molt més senzilla es poden aplicar els algorismes que proporciona el mòdul de *Python Scikit-learn* per passar-li els pesos que s'han aconseguit anteriorment amb la variable W. Aquestes W s'utilitzaran juntament amb la seva etiqueta o "label" corresponent (els bits de sortida del 5-bit y_1, y_2 i y_3 amb una funció del *Scikit-learn LinearRegression* [37] el qual ens proporciona les possibles prediccions del problema.

La regressió lineal s'adapta a un model lineal amb coeficients $W = (W_1, \dots, W_p)$ per minimitzar la suma residual dels quadrats entre els objectius observats al conjunt de dades i els objectius predits per l'aproximació lineal. La predicció ve donada per la funció següent

$$\begin{cases} 0 & si\ x < 0{,}5 \\ 1 & altrament \end{cases}$$

En les *Figures 11 i 12* es pot veure un parell d'exemples de com quedaria un autòmat cel·lular elemental a l'acabar totes les iteracions del reservori.

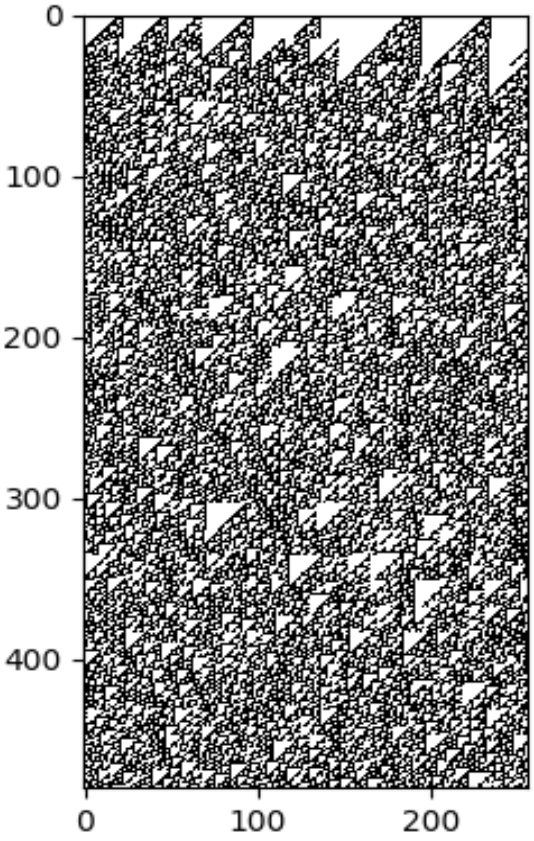


Figura 11: autòmat cel·lular per la primera entrada

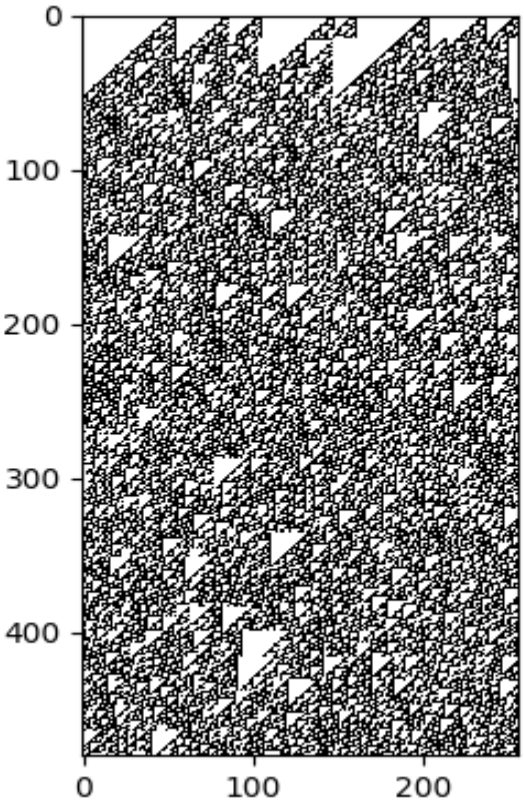


Figura 12: autòmat cel·lular elemental per la tercera entrada

10.3 Experimentació del problema 5 bit

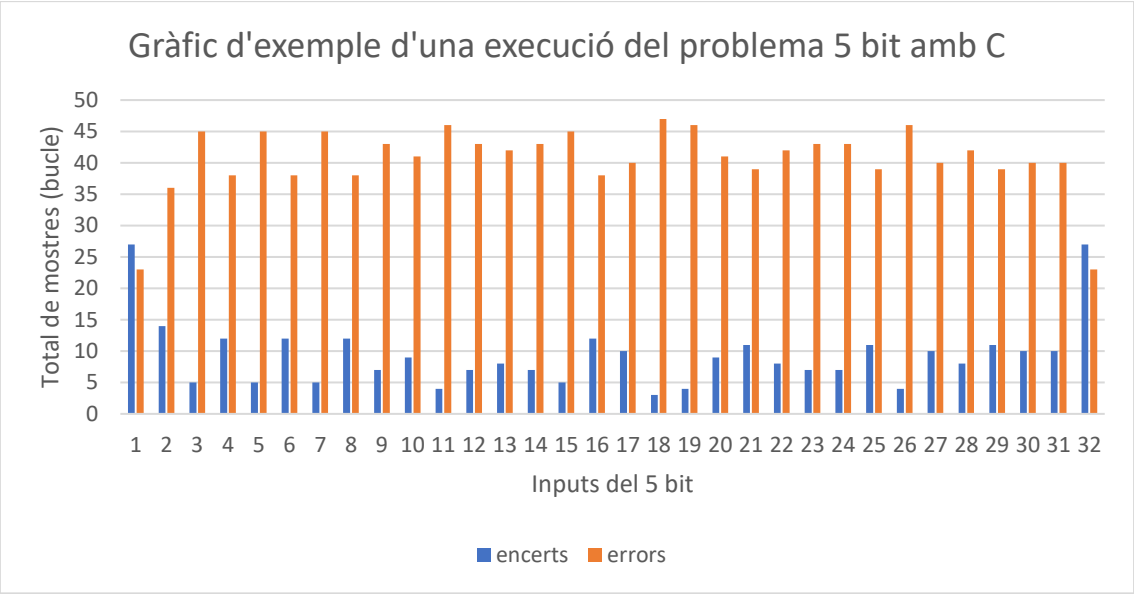
Les dades de l'experiment s'han obtingut provant de diferents valors dels paràmetres anomenats a l'apartat anterior: *R* (mapar les entrades de manera aleatòria), *C* (valor a multiplicar la mida de l'entrada), *I* (nombre d'iteracions que la regla 110 portarà a terme) i t_d (la quantitat de distractors dels 5 bit). Un últim paràmetre s'utilitzarà per provar en bucle les mateixes dades i quantes prediccions és capaç de resoldre. Una predicció serà correcte si tots els bits de la predicció coincideixen amb els bits de la sortida exactament igual.

Els resultats es mostraran en una sèrie de gràfics comparant com evoluciona el percentatge d'encerts i errors i finalment es farà una comparació amb els resultats de l'article de l'*Ozgur Yilmaz*.

Per portar a terme els experiments s'ha utilitzat 2 models diferents segons el paràmetre *C*. Un model el paràmetre *C* serà sempre 1 i els altres paràmetres es modificaran. En l'altre model el paràmetre C es modificarà de manera que el valor de C sigui més gran que 1. Per tots els experiments els paràmetres *R*, *I* i t_d seran sempre majors que 1. Hi haurà un 5è paràmetre, però que es deixarà fixat. Aquest paràmetre anomenat *b* executa tantes vegades el programa per aquest valor. Això ajuda a comprovar l'eficàcia del sistema i comprovar si és capaç de generar una sortida correcte per totes les execucions. Aquest paràmetre *b* estarà fixat a 15 (exceptuant pels 2 exemples que es trobaran més endavant, 1 per cada mètode).

10.3.1 Primer mètode

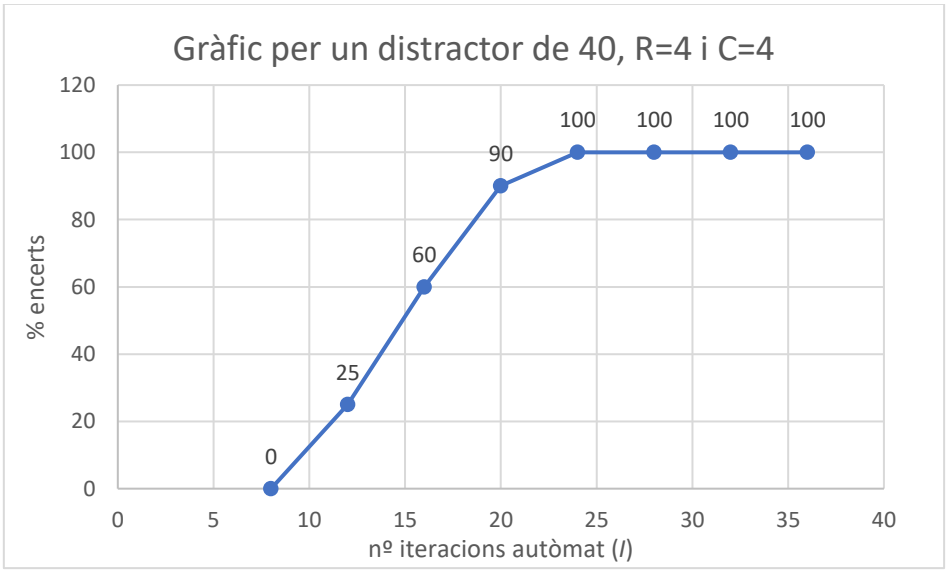
Un exemple de com es calcularan els percentatges totals per cada execució serà a partir d'un exemple del *Gràfic 1.* En aquest gràfic es tenen els paràmetres $I = 20,\ R = 4,\ C = 4$ per un bucle de 50 intents i una mostra de 40 *distractors*. Com es pot observar en general són uns resultats bastant dolents tenint en compte que el nombre d'encerts (barra blava) és molt baix per gairebé totes les entrades exceptuant per la primera i última que està més o menys al 50% tenint en compte que només s'han utilitzat 40 *distractors* com a mostra i es busca poder comparar amb mostres més altes.



Gràfic 1: Exemple d'una execució del 5 bit amb les 32 entrades per un distractor de 40, I=20, R=4 i C=4

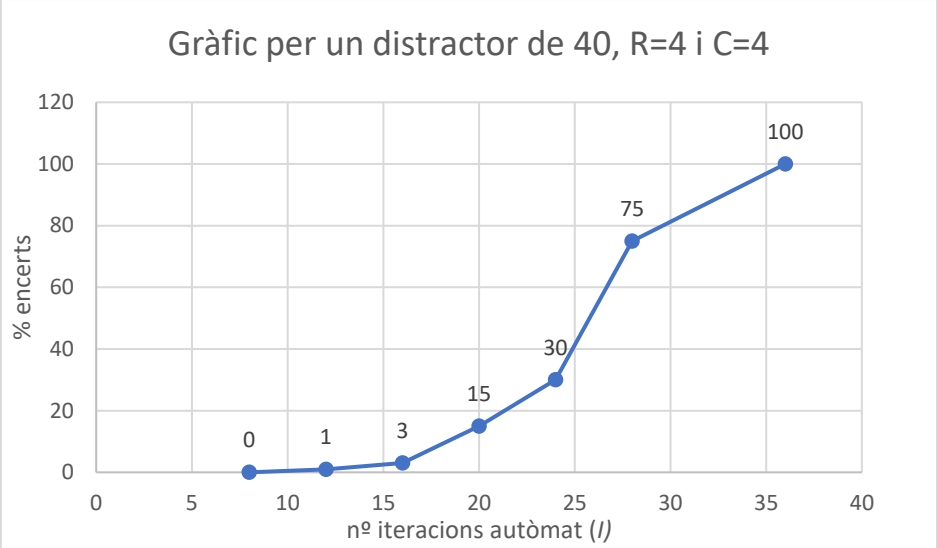
Per la resta de gràfics es calcularà un sol percentatge a partir dels 32 inputs i s'analitzarà per diferents mostres d'altres paràmetres. El paràmetres que més interessen analitzar són el nombre d'iteracions de l'autòmat ja que és on més informació útil s'obté del sistema, i els *distractors* que s'afegeixen.

Per una $C = 4$, $R = 4$ i un *distractor* de 20:



Gràfic 2: Percentatge d'encerts per un distractor de 20

Per una $C = 4,\ R = 4$ i un *distractor* de 40:



Gràfic 3: Percentatge d'encerts per un distractor de 40

Amb pocs casos analitzats es pot observar que els resultats utilitzant aquest sistema no són gaire bons ja que per uns *distractors* de mides petites necessita moltes iteracions de l'autòmat per aconseguir un 100% d'encerts. Tal com es veu en els gràfics, a més iteracions porta a terme l'autòmat, millors resultats. Per aconseguir encerts en *distractors* més elevats calen un gran número d'iteracions (més de 100) i/o combinar-ho amb més valors de R computacionalment no és eficient.

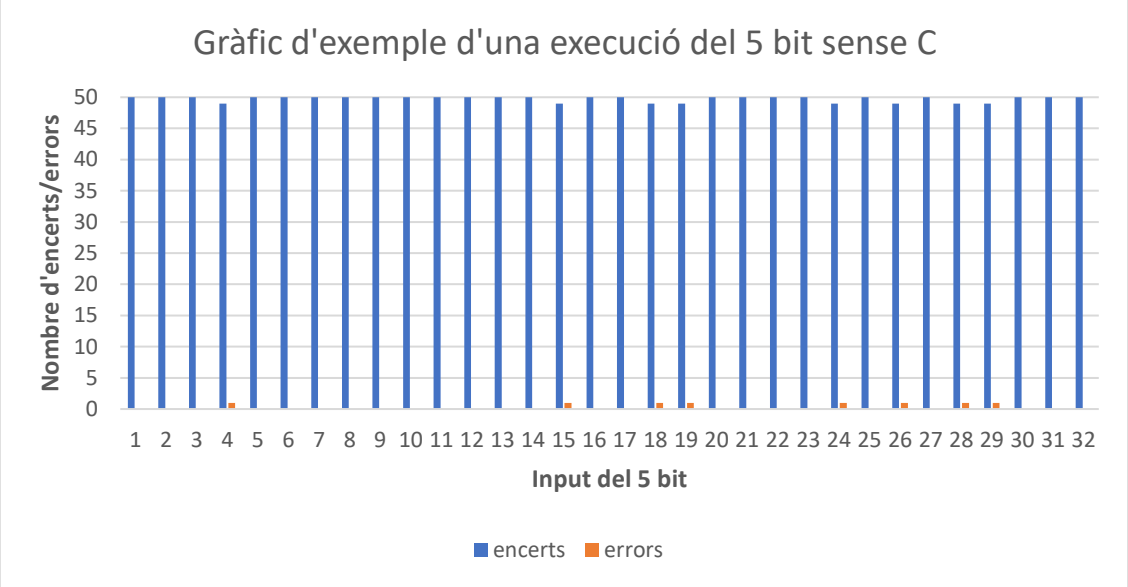
Les proves que l'*Ozgur* va porta a terme són amb *distractors* de 200 i 1000, per tant pels propers experiments s'utilitzarà el segon mètode[2] per veure si es pot aproximar més a les dades que ells obtenen.

10.3.2 Segon mètode

Per aquest segon mètode s'ha modificat lleugerament el sistema per a que no es tingui en compte el paràmetre *C* ja que com s'ha pogut analitzar per ara no ha donat bons resultats i dels articles analitzats [16][38][39] per portar a terme aquests experiments la gran diferència està entre utilitzar i no utilitzar aquest paràmetre. En l'article del *Stefano Nichele* i *Magnus S. Gundersen* utilitzen també la combinació de regles, però aquest apartat no es tindrà en compte ja que per aquest projecte únicament s'utilitza la regla 110.

Per poder comparar amb el primer model s'han utilitzat els mateixos paràmetres per la *I*, la *R,* el *distractor* i la *b* que serà 50. En el *Gràfic 4* es poden veure els resultats.

[2] Mètode on el valor del paràmetre *C* serà 1 (i per tant no es tindrà en compte)



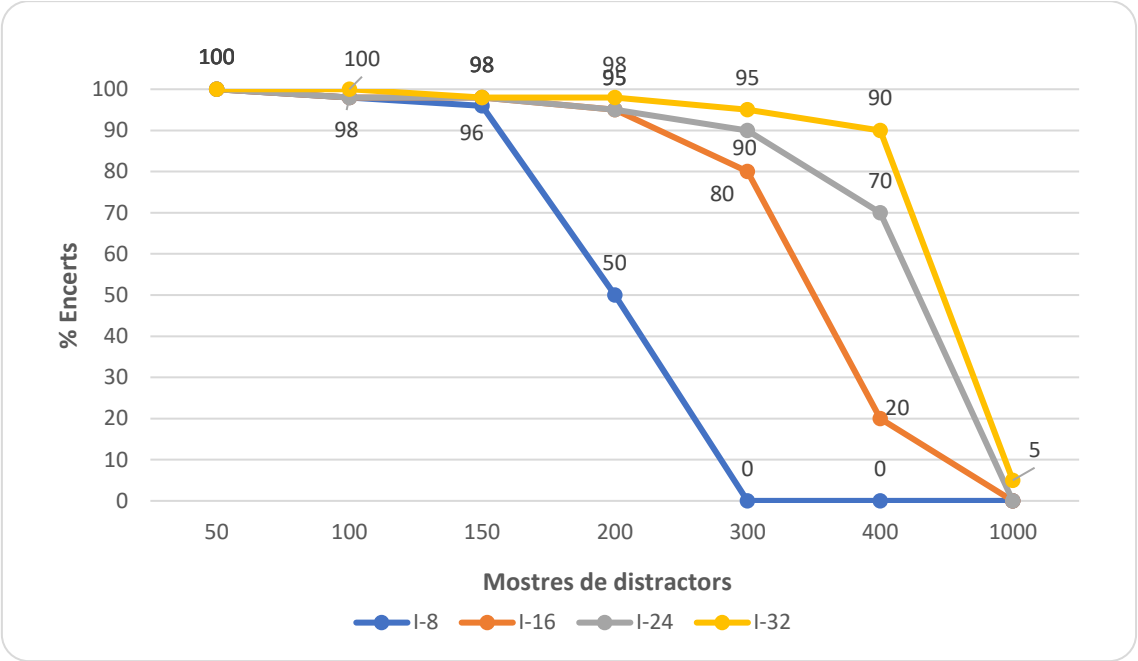
Gràfic 4: Exemple d'una execució del 5 bit amb les 32 entrades per un distractor de 40, I=20 i R=4

Com es pot observar els resultats són molts millors ja que pels mateixos valors que en el *Gràfic 4* el nombre d'encerts es gairebé del 100% en totes les entrades exceptuant en 8 d'elles (i de les 50 execucions només s'equivoca 1 vegada), per tant es pot argumentar que incrementar la mida de l'autòmat únicament afegint 0s creava més dades errònies a l'hora de predir la sortida.

Pels propers experiments s'analitzaran les prediccions obtingudes per diferents combinacions dels paràmetres *I, R* i *distractor* i es buscarà comprovar quins valors donen millors resultats pels *distractors* més elevats. Com la quantitat de proves ha estat més elevada pels bons resultats inicials l'anàlisi dels gràfics es portarà a terme de manera més detallada comparant per cada *R* diferent, el percentatge d'encerts per la combinació de les iteracions i els distractors. Pels *distractors* s'han utilitzat mostres de 50 en 50 fins a 200 i unes mostres més elevades entre 300 i 2000 depenent de com evolucionaven els resultats per iteracions petites.

Pel primer experiment s'ha escollit el valor de $R = 4$ ja que per valors més petits donarien molt poca informació perquè la mida del vector no donaria prou dades útils a l'executar l'autòmat. En el *Gràfic 5* es pot observar els resultats obtinguts. L'eix d'abscissa mostra els diferents valors de distractors escollits i l'eix d'ordenada mostra quin es el percentatge d'encerts.

Igual que en el mètode 1 a partir del segon gràfic s'aproxima el percentatge a partir de les 32 entrades i es dona un sol valor per no tenir tants valors diferents. Es pot donar el cas que una entrada no doni el 100% d'encerts i les altres si, però no es tindrà en compte ja que aquesta part del treball és més per experimentar, aprendre i veure com els autòmats funcionen i no buscar dades detallades.



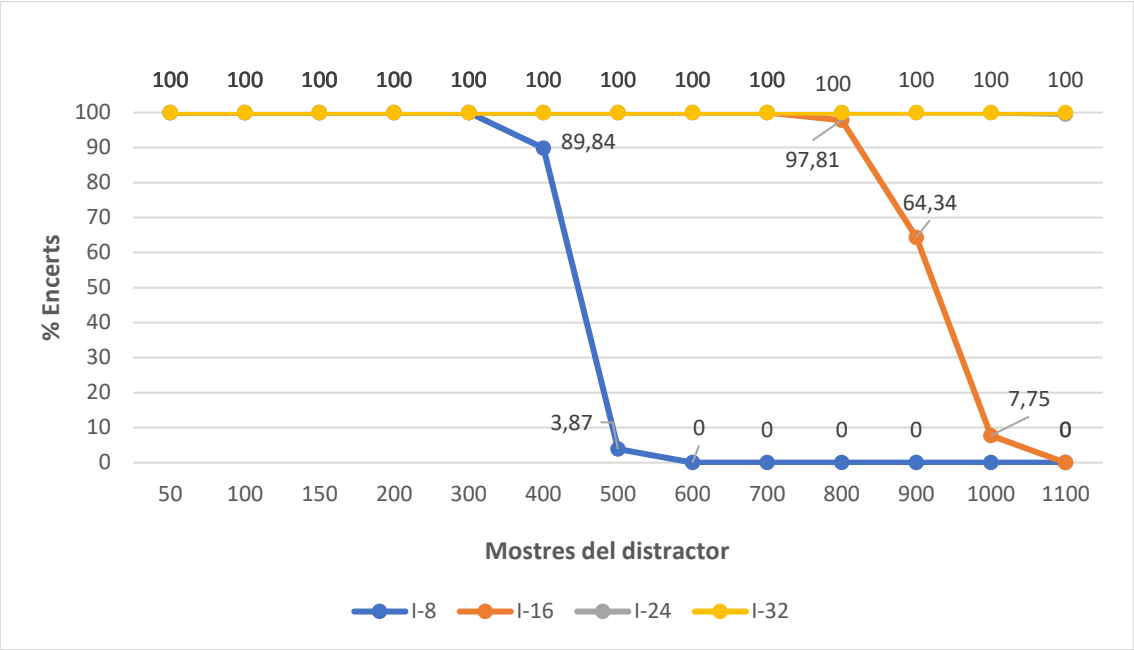
Gràfic 5: bucle de 20 i R de 4

Per una $R = 4$ s'han fet experiments amb 4 valors diferents del paràmetre *I*. Les línies del *Gràfic 5* mostren cada valor diferent: 8 (blau), 16 (taronja), 24 (gris) i 32 (groc).

Tal com es pot observar i comparar amb el *Gràfic 2* els resultats són molt millors ja que aquí ja s'aconsegueixen un 100% d'encerts amb 8 iteracions amb distractors molt més grans ja que en el gràfic 2 i 3 es mira per *distractors* concrets (20 i 40) i aquí es pot mirar amb més dades perquè els valors inicials dels *distractors* són gairebé el 100%.

Una dada important és que com més gran és el paràmetre *I*, millors resultats dona. Això té sentit perquè conté molta més informació per cada entrada i li és més fàcil fer prediccions amb *distractors* més elevats, tot i que per 1000 valors en el *distractor* només quan l'autòmat porta a terme 32 iteracions és capaç d'aconseguir algun encert, però mínim (5%).

Pel següent experiment es tindrà una $R = 8$ amb els mateixos paràmetres tant d'iteracions com de distractors. En el *Gràfic 6* es pot veure els resultats.

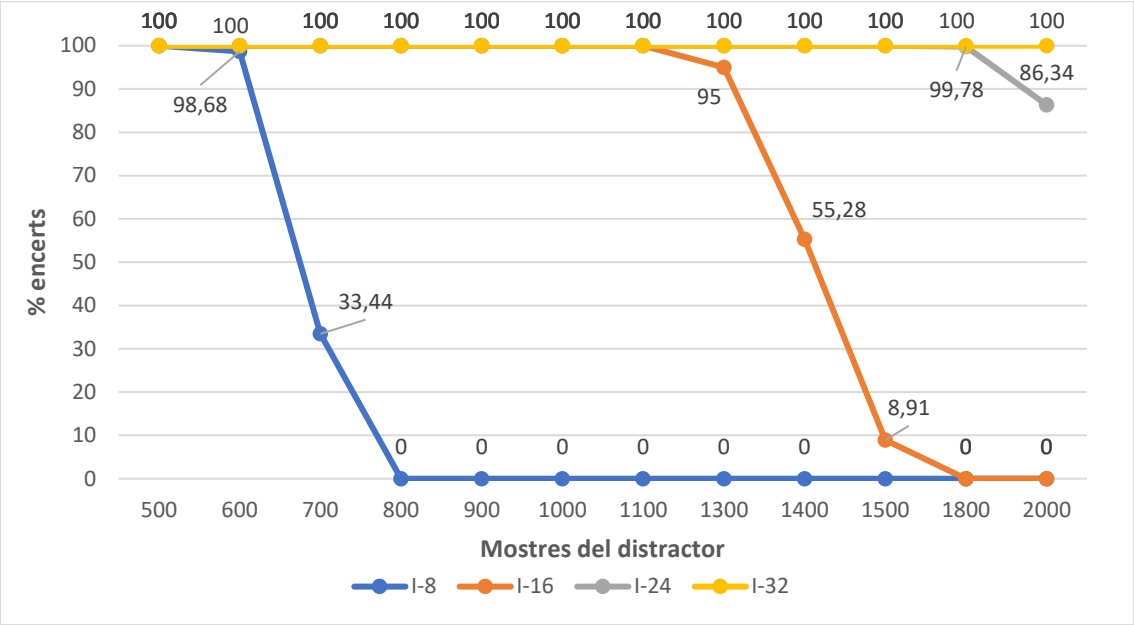


Gràfic 6: bucle de 20 per una R de 8

Per un increment de *R* es pot observar que els resultats milloren bastant més. Es pot observar que per una $I = 24$ i una $I = 32$ s'aconsegueix un 100% d'encerts per les 20 execucions del programa per tots els *distractors* estudiats, per tant demostra la fiabilitat de les dades. Per una $I = 16$ comença a produir algun cas erroni als 800 distractors i als 1000 distractors no és capaç de produir cap encert. Pel menor cas s'aconsegueix un 100% d'encerts fins a un distractor de 400 i a partir de 500 ja dona errors i no aconsegueix cap encert amb els distractors més alts.

Aquest resultats demostren l'eficàcia d'afegir informació extra a l'autòmat. Tot i que la informació de l'entrada es mapa de manera aleatòria, l'autòmat és capaç de produir prou informació útil per a que sigui capaç de predir les sortides a *distractors* elevats.

L'últim experiment es portarà a terme amb un valor de $R = 12$. En el *Gràfic 7* es poden veure els resultats.



Gràfic 7: bucle de 20 per una R de 12

Per l'últim experiment s'ha utilitzat mostres de *distractor* més grans que les anteriors ja que inicialment els resultats eren més bons i no es podria comparar amb exactitud.

Es pot observar que per una $R = 12$ s'aconsegueixen molt bons resultats ja que fins a un *distractor* de 600 no es troba el primer error per una *I* petita. A diferència de l'apartat anterior per una $I = 16$ si s'aconsegueix arribar als 1000 *distractor* amb una totalitat d'encerts.

Per finalitzar s'ha volgut experimentar en quin punt comença a no tenir una totalitat d'encerts. Per una $I = 24$ s'ha vist que inclús amb 2000 *distractors* s'aconsegueix un 86,34% d'encerts, per tant entre els pròxims 100 i 300 disminuiria a 0 possiblement, però no s'ha superat el límit el 2000 perquè la duració de les proves era excessiva.

10.4 Conclusió

El problema del 5 bit inicialment s'ha utilitzat per testejar els autòmats, la seva funcionalitat i els valors que s'aconseguien, però en cap moment s'ha plantejat fer un estudi exhaustiu del problema.

Tot i així, hi ha hagut prou temps per escriure un codi de programació funcional que, tal com s'ha observat en els gràfics del model 2, s'ha aconseguit uns molt bons resultats comparant-los amb els de l'article de l'Ozgur.

Aquest casos pràctics han servit per aprofundir en la teoria dels autòmats cel·lulars elementals, com evolucionen per passos de temps, els resultats que s'obtenen a partir de grans quantitats de dades i l'efectivitat que tenen a l'hora d'utilitzar-los per reconeixement de patrons.

En conclusió, s'ha pogut estudiar, entendre i desenvolupar un sistema útil en reconeixement de patrons. S'ha aconseguit extreure molt bons dades per diferents paràmetres i analitzar que és útil i que no és útil.

Tot aquest procediment servirà per treballar amb les imatges MNIST i fer un sistema que pugui predir amb un bon nivell de precisió les imatges disponibles.

El codi utilitzat per tot l'apartat del 5 bit està disponible al repositori del github i de manera pública a:

https://github.com/avanger9/RC-ECA-110/tree/main/implementacio/fivebitproblem

on els 2 fitxers amb el codi del problema són el **main.py** (per executar el programa amb els paràmetres) i el **reca.py** (on és portarà a terme tota la implementació).

11 Desenvolupament del MNIST

11.1 Introducció

El MNIST és un conjunt de dígits escrits a mà i són utilitzats normalment per provar i entrenar sistemes de reconeixement d'imatges. Està estructurat en 60.000 imatges d'entrenament i 10.000 per testejar. En la *Figura 13* es pot observar alguns exemples dels dígits que es poden trobar. Per visualitzar les imatges s'ha utilitzat el mòdul de *Python 3 Matplotlib* [40].

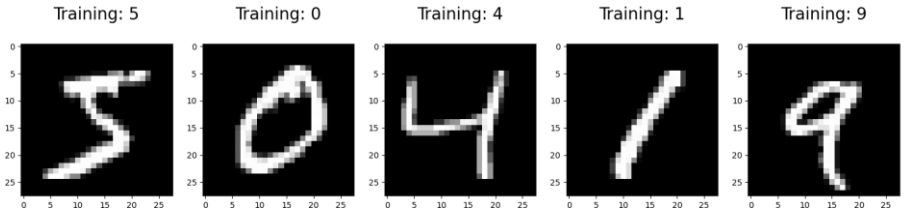


Figura 13: Diferents exemples de les imatge MNIST

Per portar a terme el desenvolupament d'un sistema capaç de reconèixer aquest conjunt d'imatges s'utilitzarà la mateixa idea que en el problema 5 bit: utilitzar els autòmats cel·lulars elementals com a reservori, però algunes fases vistes en el problema del 5 bit es modificaran per ajustar-se de manera més acurada al problema que es vol resoldre.

11.2 Desenvolupament

El desenvolupament de sistema del MNIST s'ha portat a terme utilitzant el llenguatge de programació *Python 3* i algunes llibreries com el *Numpy, Matplotlib* i *Scikit-learn* ja anomenats a l'apartat del 5-bit i la llibreria *Keras*.

Les imatges del MNIST estan estructures en tota una sèrie de files i columnes representades per píxels d'1 byte en una escala del 0 al 255 de grisos. Cada imatge està representada per 28 files i 28 columnes tenint un total de 784 píxels.

La implementació del sistema s'ha portat a terme utilitzant el treball de *"Energy-Efficient Pattern Recognition Hardware With elementary Cellular Automata"* [15] com a punt de partida per aprendre com codificar el MNIST amb els autòmats cel·lulars. A partir d'aquí s'ha portat a terme un desenvolupament del sistema propi i uns experiments amb altres dades per analitzar els resultats amb la regla 110.

La primera feina a portar a terme es decidir de quina manera es transformaran aquests bytes per tenir una representació de 0s i 1s per treballar amb els autòmats cel·lulars.

Un mètode és binaritzar la imatge, utilitzant algun mètode existent o decidint de manera personal un valor que decideixi en quina franja estan els valors que es transformaran a

0 i els valors que es transformaran a 1. Tot i que seria un mètode vàlid, per aquest projecte s'ha decidit utilitzar un altre mètode.

Aquest mètode és transformar aquest byte en valor decimal (valors que estan entre 0 i 255) en un número binari representat en 8 bits. Cada bit ocuparà lo que s'anomenarà una *"layer"* o capa. Per tant per cada imatge es tindran 8 capes on cada capa serà representada per un bit a cadascuna de les 28 files i columnes. Els bits estaran ordenats de manera que el bit de menys pes estarà a la capa 1 fins al bit de més pes que estarà a la capa 8. En la següent funció es defineix *B* com el número de capes, u com el valor decimal d'una imatge i $u^{(l)}$ cada valor binari de les 8 capes.

$$u = \sum_{l=0}^{B-1} 2^l u^{(l)} \qquad B = 8$$

Aquest sistema consistirà en la senyal d'entrada i, a diferència del problema 5 bit, no cal incrementar la mida del vector (el paràmetre *C*) (o en aquest cas matrius) ni fer un mapeig dels bits de manera aleatòria (el paràmetre *R*) ja que l'entrada ja es prou gran i conté molta informació. El paràmetre *I* si serà necessari per definir els passos en el temps de l'autòmat.

Un altre diferència amb el problema del 5 bit es que ara es tenen matrius, en comptes de vectors, per tant es tractarà de manera independent els bits en les files i els bits en les columnes a l'hora d'aplicar l'autòmat cel·lular.

Per definició, sigui la variable k els passos en el temps, la funció g^k que aplica k vegades la funció g i on $x^{(l)}(k)$ l'evolució d'$u^{(l)}$ en el temps.

$$\begin{aligned} x_r^{(l)} &= g_r^k\left(u^{(l)}\right) \\ &= \begin{cases} u^{(l)} & k = 0 \\ g_r\left(x_r^{(l)}(k-1)\right) & k > 0 \end{cases} \end{aligned}$$

Tal com s'ha dit, es tractarà de manera independent les files i les columnes. A més a més, cada capa serà independent una de l'altre. Per les files es tindrà una matriu $x_r^{(l)}(k)$ on s'aplicarà k vegades l'autòmat cel·lular a cada fila de la imatge $u^{(l)}$. De manera paral·lela es farà el mateix càlcul per les columnes on es tindrà la matriu $x_c^{(l)}(k)$ on s'aplicarà k vegades l'autòmat cel·lular a cada columna de la imatge $u^{(l)}$.

$$\begin{aligned} x_c^{(l)} &= g_c^k\left(u^{(l)}\right) \\ &= \begin{cases} u^{(l)} & k = 0 \\ g_c\left(x_c^{(l)}(k-1)\right) & k > 0 \end{cases} \end{aligned}$$

Aquests 2 sistemes $x_r^{(l)}(k)$ i $x_c^{(l)}(k)$ es combinaran utilitzant la funció XOR. Com la funció XOR es porta a terme bit a bit es tindrà una variable i i una variable j que defineixen cadascuna de les files i columnes respectivament on $\oplus$ és $el\ símbol\ de\ l'OR\ exclusiu$ [41].

$$\left(x^{(l)}(k)\right)_{ij} = \left(x_r^{(l)}(k)\right)_{ij} \oplus \left(x_c^{(l)}(k)\right)_{ij} \qquad i,j = 0,\dots,27$$

Arribat a aquest punt de manera senzilla s'ha portat a terme tots els nodes del reservori per cada capa, fila i columna. El proper pas es reintegrar les 8 capes de nou per tornar a tenir els valors en l'escala de grisos. Es porta a terme el pas inicial de manera inversa convertint el número binari en decimal tenint en compte que a $u^{(7)}$ esta el bit de més pes i a $u^{(0)}$ el bit de menys pes.

Per no tenir matrius tant grans es fa una *reducció de dimensionalitat* via *maxpooling* [42] transformant-les de 28 files i 28 columnes a 14 files i 14 columnes, per tant es passa de tenir 784 Bytes a 196 Bytes. Per no perdre excessiva informació la reducció de dimensionalitat mira cada 4 bytes en quadrats de 2 files i 2 columnes i d'aquest 4 bytes s'escull el valor més gran.

L'últim pas abans de poder fer l'entrenament es convertir aquesta matriu en un vector concatenant cadascuna de les files. La mida final del vector serà de $196 \times I$ on I és el paràmetre anomenat anteriorment per definir el número de passos que tindrà l'autòmat cel·lular.

El resultat de la sortida vindrà definida per l'equació següent:

$$\hat{y} = xW$$

On x és cada vector definit en el paràgraf anterior amb tota la informació de cada imatge comprimida en un vector de $196 \times I$. La variable W són els pesos del vector pels possibles 10 dígits diferents que es poden predir $(0,1,2,3,4,5,6,7,8,9)$ definits en una variable Q. Per cada dígit i, $\hat{y}_i$ serà l'aproximació donada per l'activitat del reservori.

Com és una aproximació s'ha de minimitzar el possible error que es doni respecte el valor esperat, per tant per poder optimitzar s'utilitzarà la funció *Softmax* [43].

La funció *Softmax* és una funció utilitzada en les regressions logístiques multinomials per normalitzar les sortides de les xarxes neuronals en probabilitats. Aquest mètode més conegut com un classificador de màxima entropia permet millorar la precisió del sistema.

Es defineix una funció *Softmax S* tal que:

$$S\left(\hat{y}^{(q)}\right) = \frac{exp\left(\hat{y}^{(q)}\right)}{\sum_{7q'} exp\left(\hat{y}^{(q')}\right)}$$

A partir de la funció *S* es construeix una matriu amb informació dels dígit que pertany a cada categoria q on $q = \{0,1,2,3,4,5,6,7,8,9\}$. La funció *S* conté 10 components que si funcionés perfectament tindria 9 zeros i un 1 i la posició del valor 1 indicaria el dígit de la imatge. Per posar un exemple, si la imatge u correspon al dígit 3, el vector de components *S* seria $S(\hat{y}_u) = \{0,0,0,1,0,0,0,0,0,0\}$ on es parteix de la posició 0 com la posició inicial. Tal com es pot observar a la posició 3 hi ha el valor 1, per tant indica que aquesta imatge pertany al dígit 3. Aquesta tècnica de convertir un valor en diferents components binàries s'anomena *"one-hot encoding"* [44].

Tots aquest passos es poden optimitzar i programar de manera senzilla gràcies a les llibreries de *Machine Learning* que proporciona el llenguatge Python. L'*Scikit-Learn* mencionat anteriorment i el *Keras* [45] són mòduls molt potents amb un gran ventall de funcionalitats que ajudaran a programar totes les funcionalitats de l'entrenament de manera òptima i amb diferents paràmetres per poder analitzar més profundament quins valors són més adequats per l'entrenament del sistema.

11.3 Entrenament i experimentació

La llibreria *Keras* del llenguatge Python es l'escollida per portar a terme la fase de l'entrenament i l'experimentació. Si els resultats no fossin prou raonables o si hi hagués més temps es miraria d'utilitzar l'*Scikit-Learn* o alguna altre llibreria.

11.3.1 Fase d'entrenament

Tal com s'ha explicat en l'apartat anterior s'executa tot el sistema programat fins aconseguir per cadascuna de les 60.000 imatges del MNIST una transformació de 784 bytes repartits en 28 files i 28 columnes en 196 bytes en una sola fila. El nombre final de bytes del vector vindrà definit pel paràmetre *I*. Per tenir diferents mostres s'ha decidit juntament amb el director del projecte provar amb 3 valors diferents: quan $I = 10$, quan $I = 13$ i quan $I = 16$.

L'entrenament del sistema es portarà a terme amb cadascun d'aquest 3 valors del paràmetre *I*, però abans de crear el model i executar-lo s'ha de dividir cada valor del vector per 255. Aquest pas és necessari ja que la funció *Softmax* treballa amb probabilitats i cal tenir els valors en tant per u.

El següent pas és crear un model amb els paràmetres corresponents. El *Keras* proporciona la classe *Sequential* [46] que permet crear el model i afegir diferents característiques que automatitzen els passos necessaris per l'entrenament.

El primer pas que el model ajuda a automatitzar és la funció *Softmax*. En l'apartat anterior s'ha explicat que era la funció i com s'aplicava matemàticament, però amb

l'ajuda de la classe *Dense* es pot aplicar directament el *Softmax* amb l'entrada corresponent.

La classe *Dense* [47] és una classe de la llibreria K*eras* que permet crear una "dense layer" o capa de neurones formals. Aquesta capa està connectada amb les capes posterior i anterior i cada neurona rep l'entrada de totes les neurones de la capa anterior.

En la *Figura 14* es pot veure un exemple de com es connecten cadascuna de les neurones capa rere capa, fins obtenir una sortida.

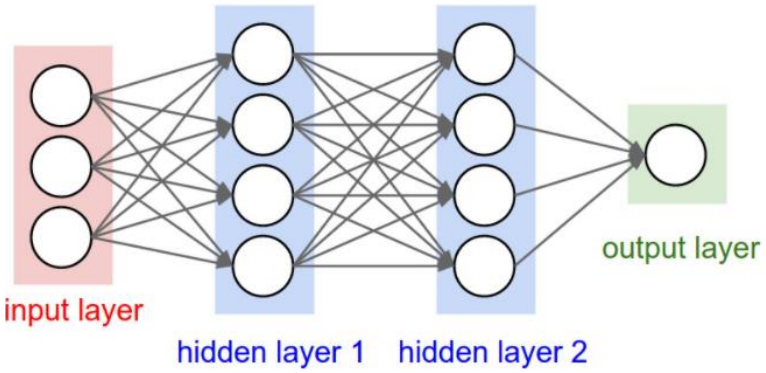


Figura 14: Exemple de les capes del Dense layer

Amb la funció d'activació aplicada i un número de capes escollit es construeix el model. En la *Figura 15* es pot observar el model construït amb la classe *Sequential*. Amb la classe *Dense* se li afegeix 10 neurones, la funció d'activació i la mida de l'input (per aquest exemple el paràmetre *I* és 16, per tant $196 * 16 = 3.136$). Finalment es visualitza executant una funció de la classe *Sequential Summary*.

Per aquest exemple es té un total de 3.136 neurones d'entrada cadascuna d'elles està connectada a cadascuna de les 10 neurones de sortida, per tant es tindran un total de 31.360 paràmetres. A més, cada neurona de sortida té el que s'anomena un biaix, per tant el total de paràmetres per aprendre seran de 31.370

```
 9   model = Sequential()    ## instance of Sequential class
10   model.add(Dense(10, activation='softmax', input_shape=(input_shape,)))  # output layer
11   model.summary()
```

```
(60000, 3136)
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
module_wrapper (ModuleWrappe (60000, 10)               31370
=================================================================
Total params: 31,370
Trainable params: 31,370
Non-trainable params: 0
```

Figura 15: Codi i visualització del model utilitzant Jupyter

El següent pas és compilar i entrenar el model. Per aquest 2 passos es tenen els principals paràmetres que es tindran en compte a l'hora de portar a terme els experiments corresponents, juntament amb els 3 valors del paràmetre *I*.

El mètode "*compile"* o compilar serveix per configurar el model. Aquest mètode té diferents arguments per afegir més informació, però per aquest entrenament únicament es tindran en compte 3: "*optimizer"* o optimització*, "loss"* o pèrdua i *"mètrics"* o mètrica. Tot i els 3 arguments, l'únic considerat paràmetre serà el d'optimització ja que proporciona diferents optimitzadors i interessa analitzar com evoluciona i si hi ha diferències significatives. Per la mètrica s'utilitzarà l'"*accuracy*" o precisió ja que el que realment interessa del projecte i que es pugui comparar amb altres tècniques és el percentatge d'encerts de les 10.000 imatges utilitzades per testejar el model. Per la funció d'error s'utilitzarà la de "categorical crossentropy". La fórmula que el programa porta a terme és la següent:

$$\mathcal{L}\left(S\left(\hat{Y}\right), L_T\right) = -\frac{1}{L}\sum_{u}\sum_{q=0}^{9} L_u^q \cdot \log\left(S\left(\hat{Y}_u^q\right)\right) + \frac{1}{C}\|W\|_F$$

El terme L indica el nombre d'imatges a treballar on L_u és cadascuna de les imatges individualment. El terme $S\left(\hat{Y}\right)$ és el vector de 10 components on, si funcionés perfectament, seria un vector de 9 zeros i un 1 a la posició de la classe que li correspon a u. L'últim terme és la *norma de Frobenius* [48] on si W és una matriu $m \times n$ es tindrà la següent fórmula:

$$\|W\|_F = \sqrt{\sum_{i=0}^{m}\sum_{j=0}^{n}\left|w_{ij}\right|^2}$$

Els optimitzadors que s'utilitzaran seran els següents: *Adam*, *Adamax, Nadam*, *RMSprop* i *SGD* [49]. Hi ha alguns més, però respecte a l'article de la web onlytojay [50] els mencionats anteriorment són els que millors resultats poden donar. Els optimitzadors tenen el paràmetre de *"learning rate"* que també serà utilitzat com un paràmetre.

L'últim pas de l'entrenament és utilitzar la funció *"fit"* de la classe *Sequential*. Aquesta funció serveix per tenir un model entrenat i funcional a partir de les dades d'entrada. En aquest punt s'utilitza el nou vector de les dades d'entrada ($196 \times I$) i les etiquetes de cada entrada. El procediment es que aprengui a reconèixer cadascuna de les 60.000 imatges passant-li un vector amb les dades d'entrada i el seu dígit corresponent. La funció *"fit"* també té 2 arguments més que es consideraran com paràmetres: "*batch size"* i *"epochs"*.

L'argument del *"batch size"* és un número que serveix per dir quantes mostres s'utilitzaran per actualitzar el gradient, així va entrenant en un bucle de mostres en comptes de fer-les totes a la vegada. Els *"epochs"* són cicles d'entrenament. Un sol *epoch* és quan es passa totalment un conjunt de dades per una xarxa neuronal. Més

epochs permeten entrenar més vegades la xarxa neuronal i ajustar l'error, tot i que masses *epochs* poden provocar més errors. Com es pot observar en la *Figura* 16 la idea és ajustar el nombre d'*epochs* perquè la corba quedi en la posició més òptima. Pocs *epochs* poden fer que la corba estigui en *"underfitting"* i el model no s'entreni correctament o masses *epochs* provoquen que la corba estigui en *"overfitting"* i excedeixi el nombre necessari pel model.

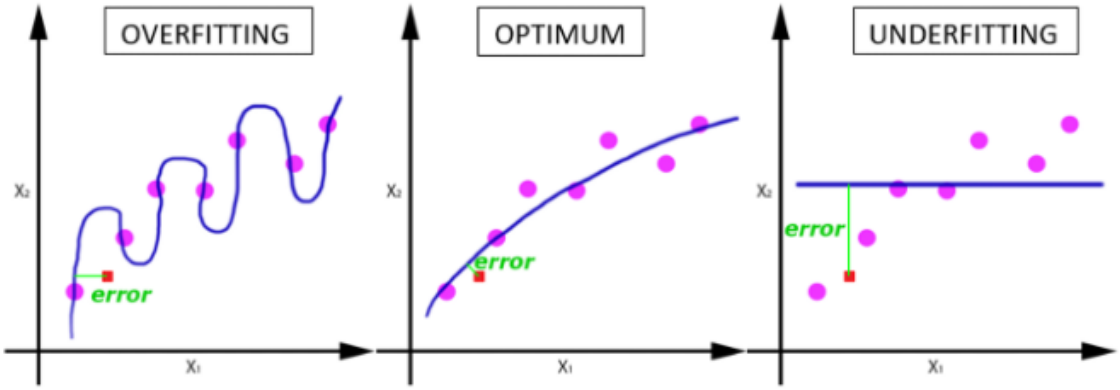


Figura 16: Corba dels canvis amb diferents epochs [51]

La idea a l'hora d'entrenar el model és ajustar el nombre d'*epochs* i *batch size* i buscar quins valors poden donar millors resultats.

Per tant, per la fase d'experimentació s'escollirà cadascun dels paràmetres *I* i es provaran els diferents optimitzadors amb un valor de *learning rate*. En el moment d'entrenar el model s'aniran provant diferent *batch size* i *epochs* i s'analitzarà la precisió d'encerts per les 10.000 imatges que es passaran com a test.

11.3.2 Fase d'experimentació

La fase d'experimentació es portarà a terme amb tots els paràmetres anomenats en el subapartat apartat anterior i modificant-los per veure els resultats dels experiments.

Per realitzar el experiments no es faran proves amb cadascun dels paràmetres explicats anteriorment ja que es tindrien masses dades que, per aquest projecte, no són necessàries. Al final la idea és veure com evoluciona l'aprenentatge a partir dels autòmats cel·lulars i quina precisió s'arriba a aconseguir. Es podria intentar investigar més i treballar exhaustivament amb cada paràmetre, però seria en un futur projecte de més complexitat.

La idea al darrera d'aquest experiments tant del professor que supervisa el projecte com de l'estudiant que el porta a terme és visualitzar uns quants resultats i extreure algunes conclusions preliminars.

Els 3 principals casos que s'investigaran serà per 3 valors del paràmetre I diferents: 10, 13 i 16. Per cada I s'analitzaran alguns dels optimitzadors anomenats en l'apartat

anterior i diferents *batch size*. Si hi hagués més temps es provarien amb alguns altres paràmetres.

Per cada optimitzador escollit es tindran diferents mides de *batch size*. Les mides escollides aniran de 5 a 2000 com a màxim.

Finalment, per cada optimitzador escollit i per cada paràmetre *I* s'executarà 10 vegades de manera consecutiva i es calcularà la mitjana i la desviació estàndard. Cada gràfic mostrarà el resultat final obtingut.

En la *Taula 17* es pot observar alguns dels paràmetres i paràmetres utilitzats. Per aquest exemple només l'optimitzador i les iteracions de l'autòmat es modificarien, la resta es mantindria igual per tots els casos.

paràmetres del model	
Regla de l'autòmat	110
iteracions de l'autòmat	10
Optimitzador	Adam
Learning rate	0.008
Epochs	40
Loss	Categorical Crossentropy
Activation	Softmax
Metrica	Precisió

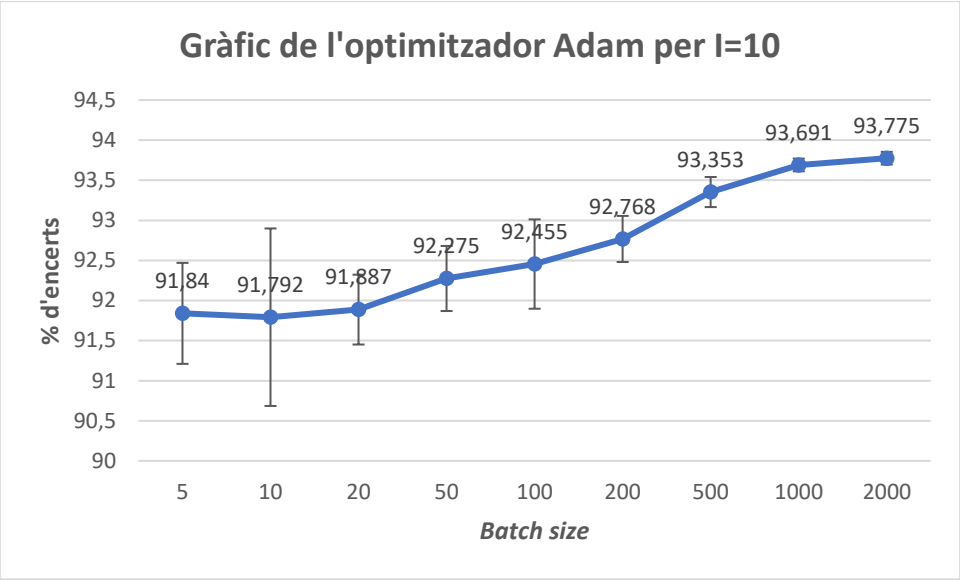
Taula 17: Exemple de paràmetres

11.3.2.1 Adam

El primer optimitzador que s'analitzarà serà l'*Adam* que, tal com diu l'article de la web *onlytoyjoy* anomenat a l'apartat anterior, és un dels que dona millors resultats. Tot i així a l'utilitzar una tècnica diferent i experimental aquest projecte no es centrarà en comparar amb els resultats de l'article sinó directament amb un cas base que serà el cas que no se li passi cap iteració a l'autòmat cel·lular.

Adam (abreviatura de *Adaptive Moment Estimation*) és un optimitzador que calcula les taxes d'aprenentatge a partir de la mitjana del *momentum* dels gradients. El *momentum* és una tècnica molt utilitzada en el món del *Machine learning* on per cada pas de propagació de l'algorisme acumula el gradient per determinar la direcció corresponent.

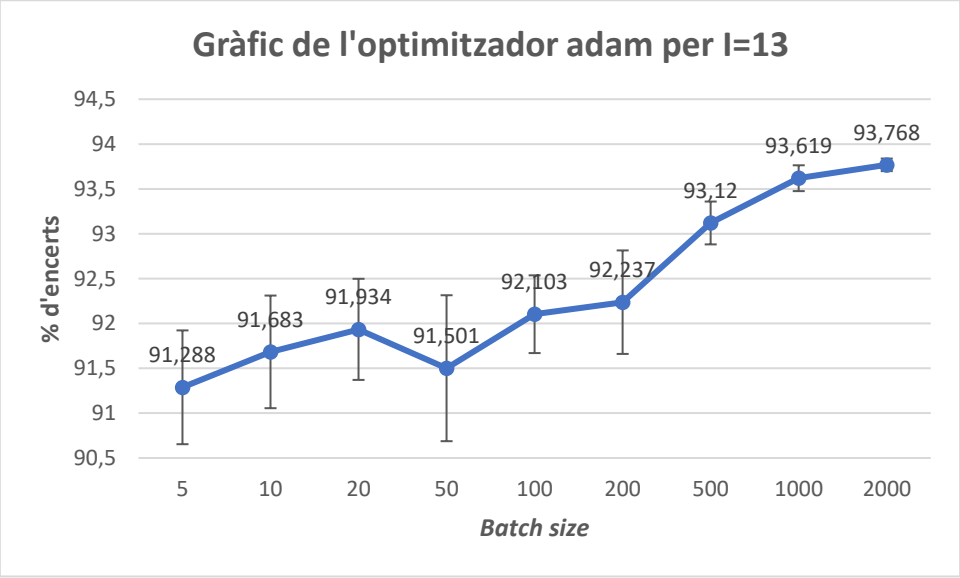
En el *Gràfic 8* es pot observar els resultats obtinguts a l'executar el software amb les funcionalitats del *Keras* i amb els diferents *batch size* en l'eix d'abscisses per una $I = 10$.



Gràfic 8: Gràfic de l'optimitzador Adam per I=10

Com es pot observar els nivells de precisió estan entre un 90 i un 94% amb un pic de 93,775%. Per l'optimitzador *Adam* seria interessant si en *batch*s més elevats s'aconsegueix millorar una mica més el percentatge. A més que es por apreciar que a l'augmentar els *batchs* hi ha menys diferències entre cada execució, ja que la desviació dels primers *batchs* és bastant elevada donant resultats diferents entre cada execució.

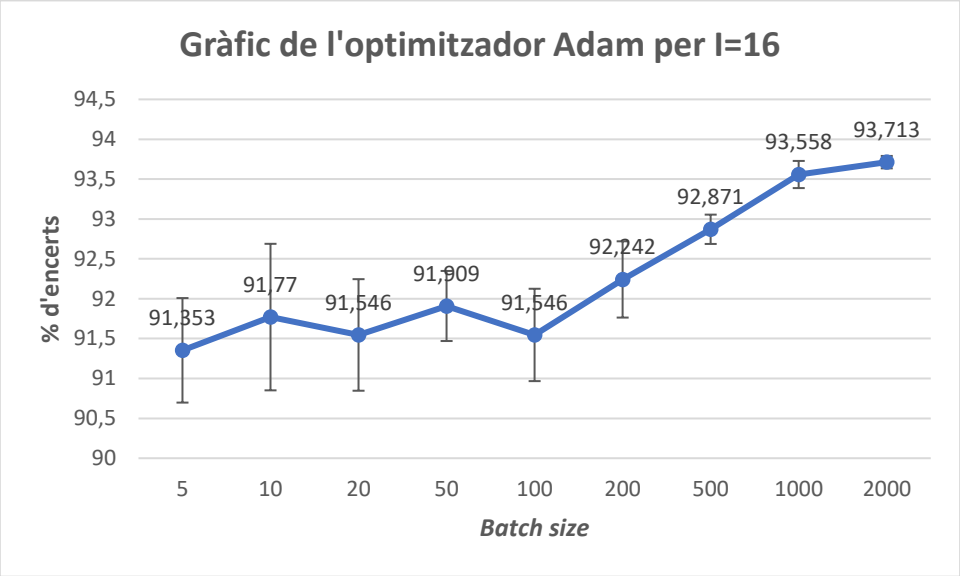
En el *Gràfic 9* es pot observar el mateix optimitzar per una $I = 13$.



Gràfic 9: Gràfic de l'optimitzador Adam per I=13

Com es pot observar tant els nivells de precisió com la desviació estàndard són molt similars al cas anterior. Per un costat, ens indica que a l'augmentar l'autòmat 3 iteracions més no s'aconsegueix un resultat visiblement diferent, però segueix donant uns bons resultats amb els *batchs* més elevats.

L'últim cas a analitzar de l'*adam* és per $I = 16$. En el G*ràfic 10* es pot observar els resultats de l'execució.



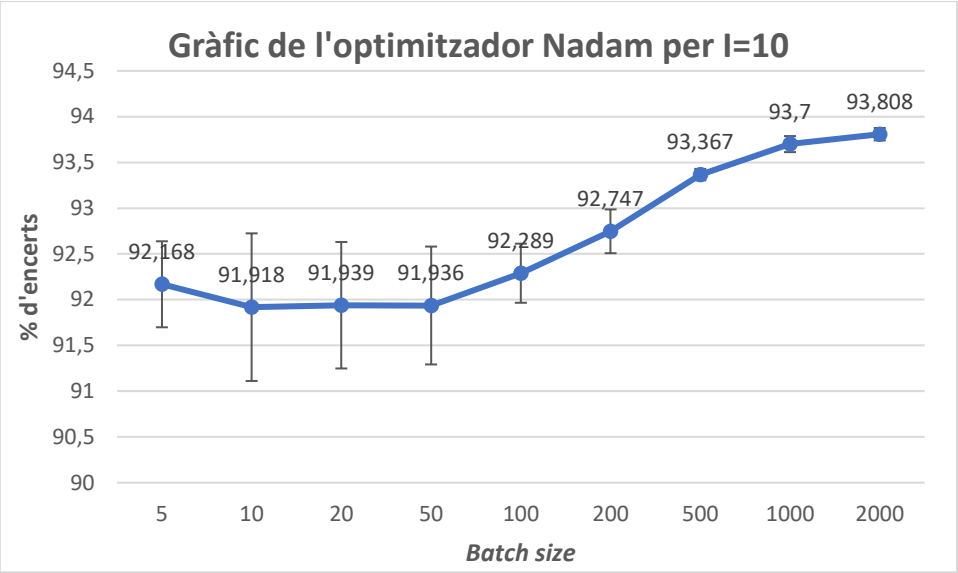
Gràfic 10: Gràfic de l'optimitzador Adam per I=16

Els resultats en els 3 casos d'I són gairebé similars i els nivells de precisió actuen de la mateixa manera augmentant amb majors nivells de *batch size* i amb una desviació estàndard molt petita i on els casos de *batchs* menors tant la precisió com la desviació són més irregulars.

11.3.2.2 Nadam

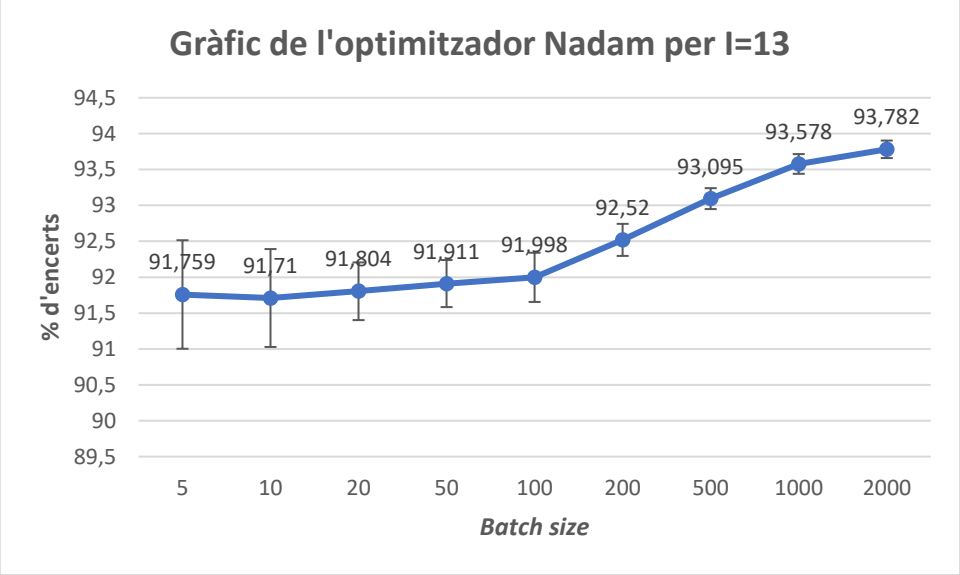
El següent optimitzador a analitzar serà el *Nadam*. El *Nadam* és un optimitzador que, a diferència de l'*adam*, accelera l'estimació del moment adaptatiu. A més, no intenta predir la següent posició sinó que actualitza directament la direcció del gradient actual dos vegades en la posició actual.

En el G*ràfic 11* es pot observar els resultats per una $I = 10$.

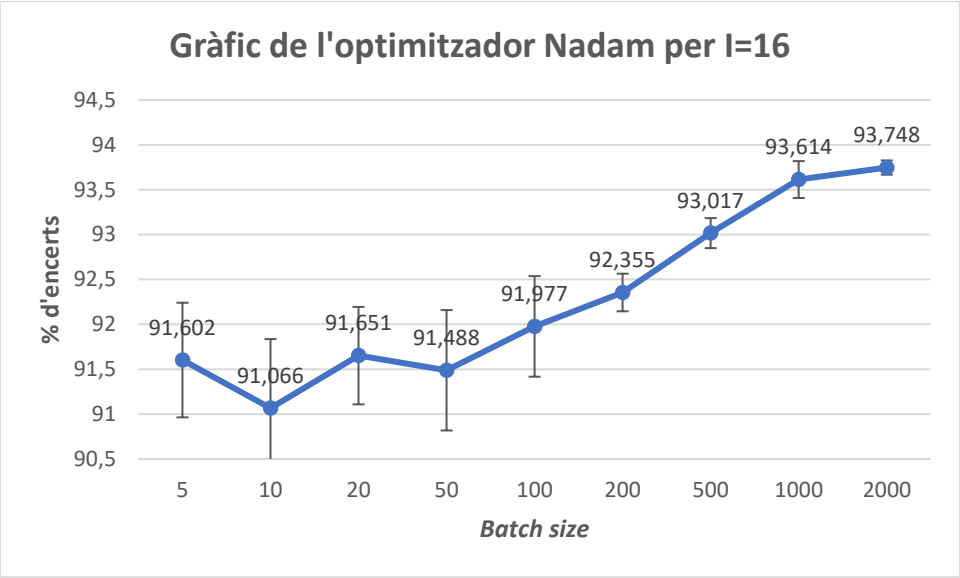


Gràfic 11: Gràfic de l'optimitzador Nadam per I=10

En els G*ràfics 12* i 13 es pot observar els resultats de l'execució per una $I = 13$ i una $I = 16$.



Gràfic 12: Gràfic de l'optimitzador Nadam per I=13



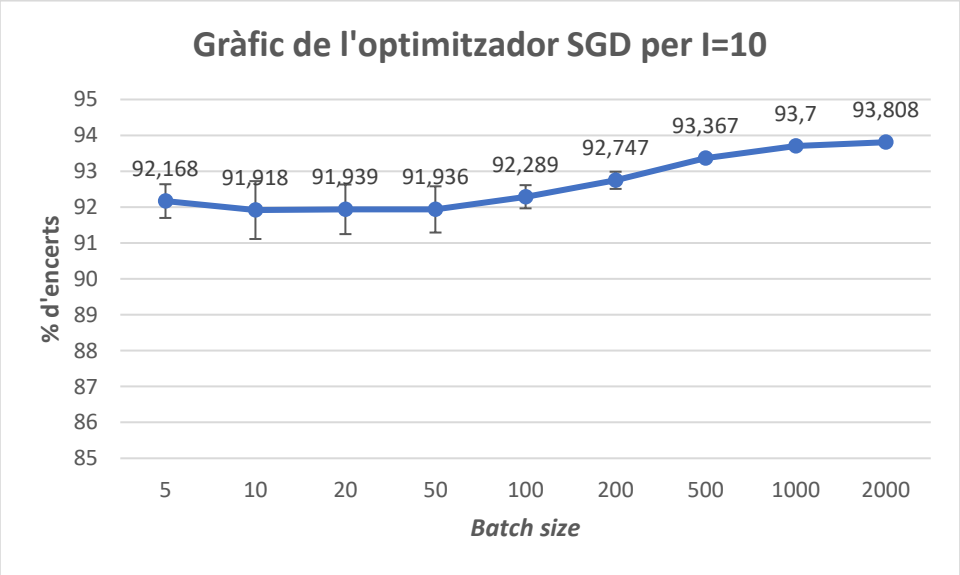
Gràfic 13: Gràfic de l'optimitzador Nadam per I=16

Es pot observar que per l'optimizador *Nadam* s'aconsegueix millors resultats amb *batchs* més elevats donant valors de precisió pròxims al 94% i amb una desviació de les dades molt petita, a diferència dels casos més petits on es pot observar desviacions més elevades i pitjors resultats.

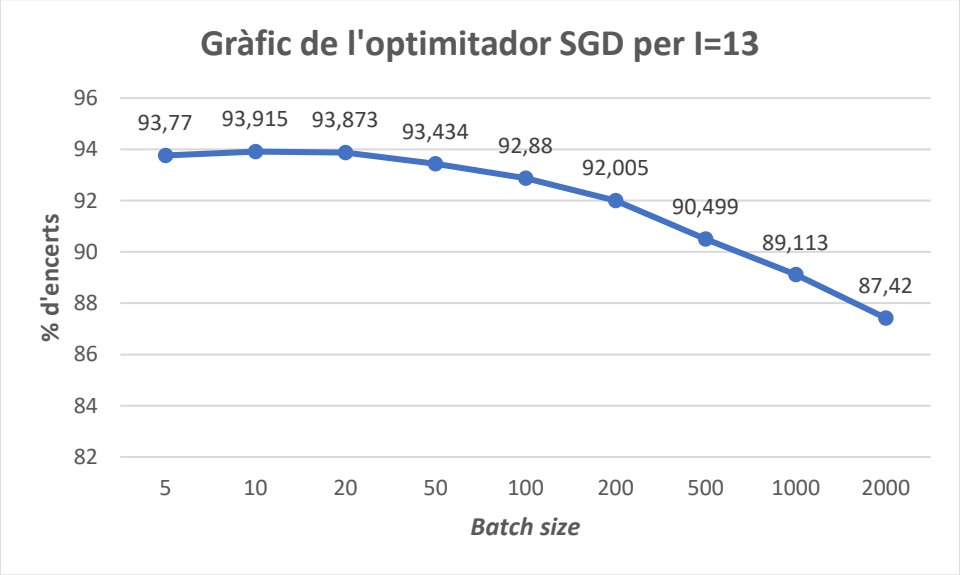
11.3.2.3 SGD

El tercer optimitzador a analitzar serà el del SGD. L'*Stochastic Gradient Descent* és un optimitzador d'aprenentatge iteratiu que reemplaça el gradient calculat a partir de les dades del conjunt per una estimació d'un subconjunt de dades a l'atzar.

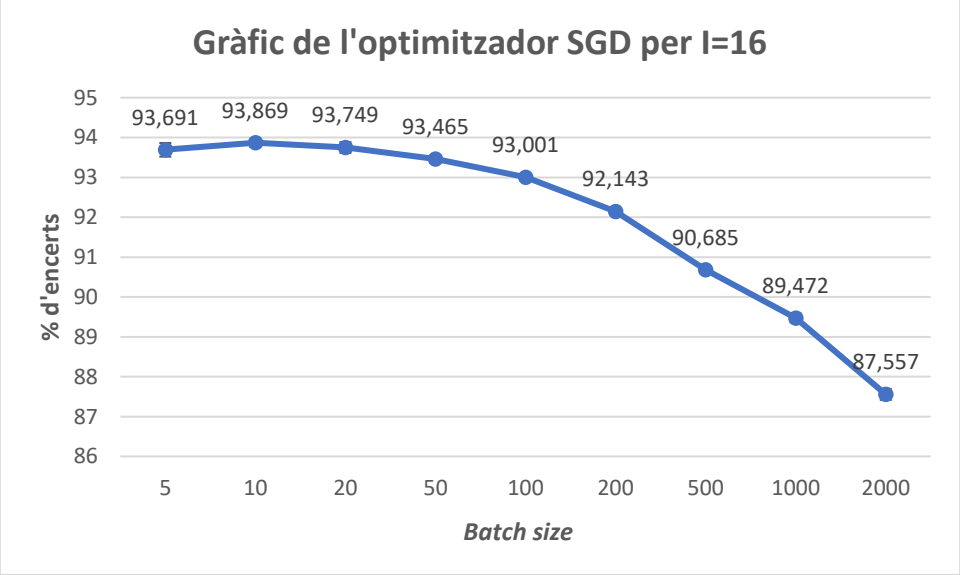
En els *Gràfics 14, 15 i 16* es poden observar el resultats per cadascuna de les 3 *I* diferents.



Gràfic 14: Gràfic de l'optimitzador SGD per I=10



Gràfic 15: Gràfic de l'optimitzador SGD per I=13



Gràfic 16: Gràfic de l'optimitzador SGD per I=16

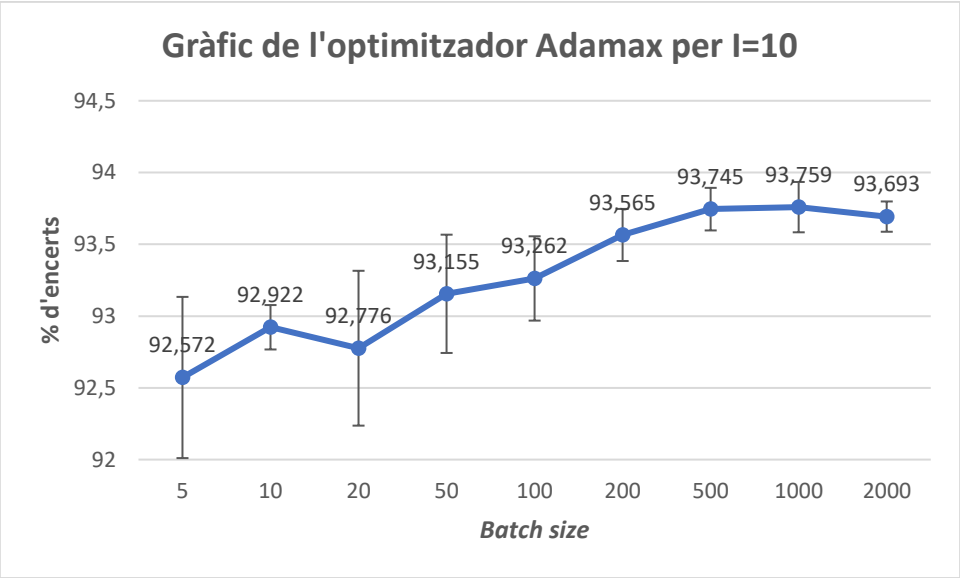
Els resultats de l'otimitzador SGD són gairebé idèntics amb bons resultats amb pocs *batch size*, però l'increment de *batch size* disminueix enormement la precisió en els casos on $I = 13$ i $I = 16$. Aquesta és una dada interessant i es veu el funcionament d'aquest optimitzador ja que és un gradient descendent que actualitza la corba per cada *batch*. Si els *batch* són de mida petita la corba s'actualitza de manera més freqüent i aquesta propietat ajuda a trobar una millor precisió. En canvi amb *batchs* de mida més gran provoquen que la corba creixi massa i s'allunya d'una predicció correcta.

Pel cas on la $I = 10$ no només no es compleix aquesta característica sinó que amb *batchs* més elevats la precisió augmenta. Podria ser un bon cas a estudiar en un altre treball on s'enfoqués el comportament explícit dels optimitzadors.

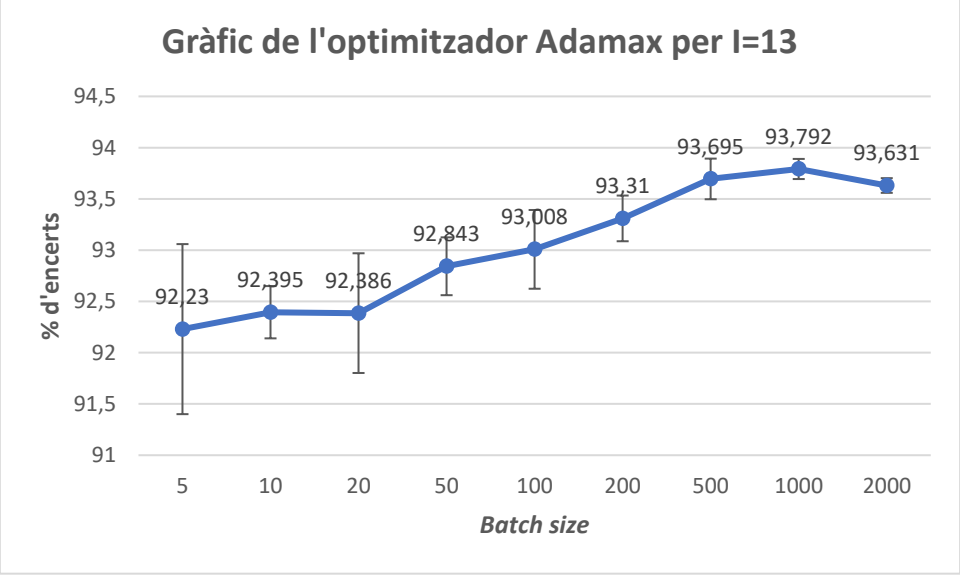
11.3.2.4 Adamax

El quart optimitzador que s'analitzarà serà l'*adamax*. L'*adamax* és un optimitzador d'aprenentatge que actualitza l'optimitzador *Adam* i les seves funcions per, en comptes de tenir en compte unes normes l_1 i l_2 es té una norma l_∞.

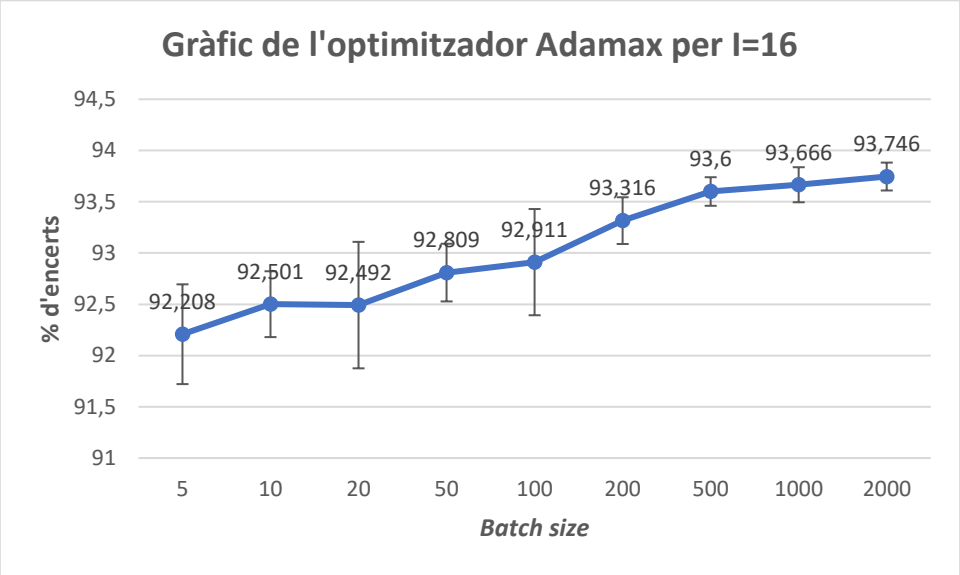
En els *Gràfics 17, 18 i 19* es pot observar els resultats de l'execució de cadascuna de les 3 *I* diferents.



Gràfic 17: Gràfic de l'optimitzador Adamax per I= 10



Gràfic 18: Gràfic de l'optimitzador Adamax per I=13



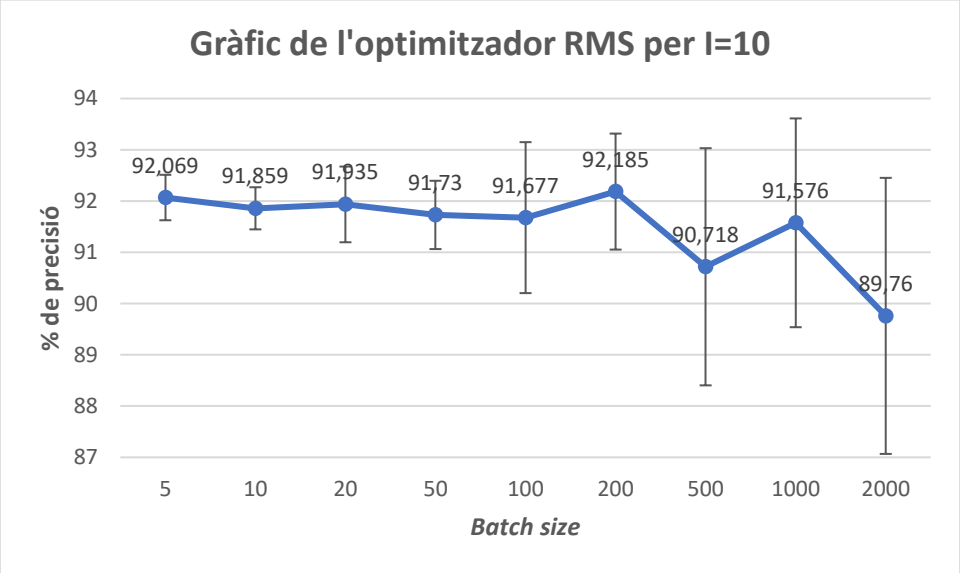
Gràfic 19: Gràfic de l'optimitzador Adamax per I=16

Es pot observar que per l'optimitzador *Adamax* s'aconsegueixen millors resultats amb un nombre de *batch* més elevat aconseguint una precisió al voltant del 93-94% i amb poca desviació estàndard, a diferència dels casos on el *batch* és menor, fet que els resultats siguin més imprecisos.

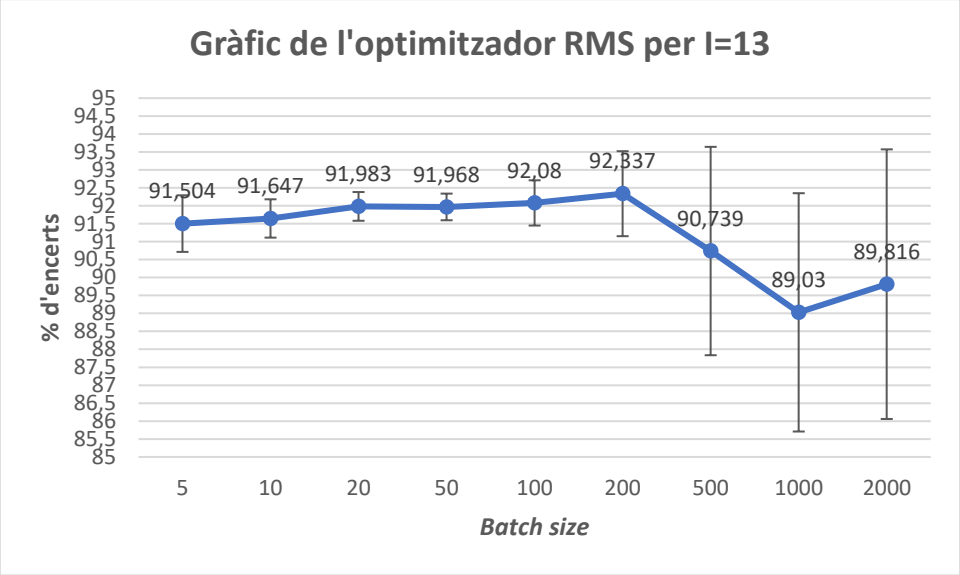
11.3.2.5 RMS

L'últim optimitzador que s'ha utilitzat per fer els experiments és el de RMS. L'*RMS* és un optimitzador d'aprenentatge que divideix la taxa d'aprenentatge del conjunt per mitjanes de les magnituds dels gradients recents.

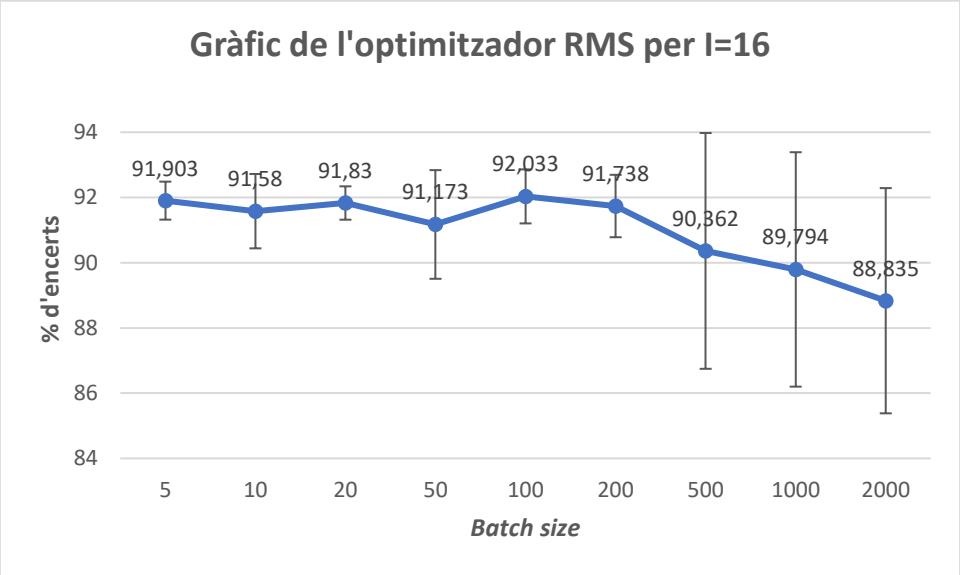
Els resultats es poden veure en els *Gràfics 20, 21 i 22* per cadascuna de les *I.*



Gràfic 20: Gràfic de l'optimitzador RMS per l=10



Gràfic 21: Gràfic de l'optimitzador RMS per l=13



Gràfic 22: Gràfic de l'optimitzador RMS per l=16

Els resultats de l'últim optimitzador mostren valors més dispars al volant del 92% en la majoria de *batch* amb alguns pics màxims aproximant-se al 94%. A més, com més incrementem els nombre de *batch* pitjors resultats s'aconsegueixen, tant a nivell de precisió com de la desviació estàndard, tal com es pot observar per cada execució el valor de la precisió canvia bastant, a diferència dels casos amb *batch* més petits.

Es pot observar que hi ha optimitzadors que amb *batchs* més petits aconsegueixen millors resultats com l'*RMS* i el *SGD* ja que es pot veure que les desviacions estàndards

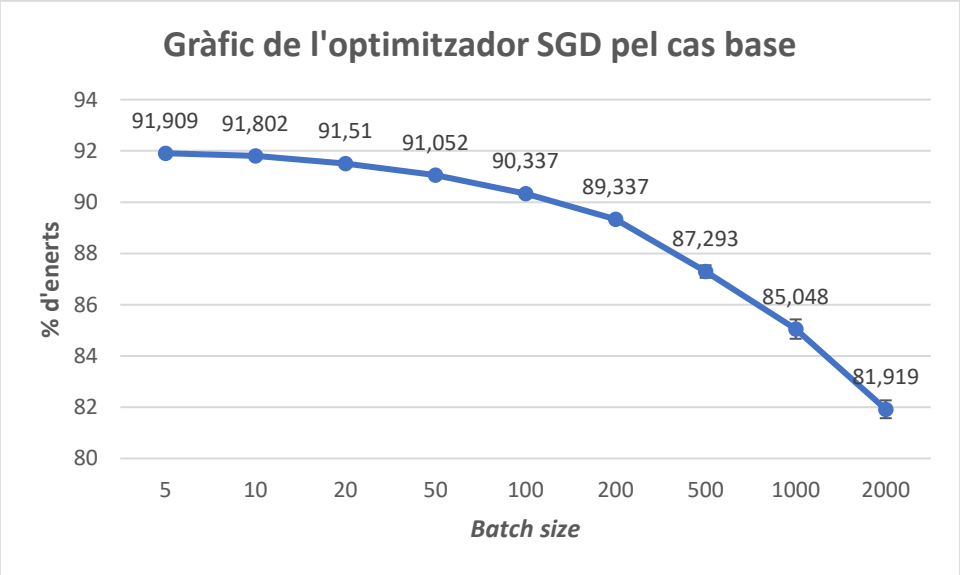
estan molt més ajustats als valors de la mitjana i altres optimitzadors com l'*Adam*, el *N*adam i l'*Adamax* actuen millor amb més quantitat de *batchs*.

Tot i les diferents proves fetes a cadascun dels optimitzadors, cap d'ells aconsegueix donar un resultat que sobresurti als altres, fet que l'elecció de l'optimitzador no és determinant per autòmat cel·lular 110 ja que gairebé tots ells arriben al 93%, però cap sobrepassa el 94%. En alguna execució en particular s'ha pogut observar el 94%, però són petites desviacions donades per l'aleatorietat existent.

Per acabar, l'optimitzador més estable és el *SGD* ja que es pot veure que pot arribar a aquest 93-94% de precisió amb *batchs* petits i, a més, la desviació estàndard en tots els casos és gairebé imperceptible, per tant demostra ser un optimitzador regular i estable per la regla 110 dels autòmats cel·lulars elementals.

Per fer una mostra comparativa, s'utilitzarà l'MNIST sense cap modificació en la xarxa neuronal i l'optimitzador *SGD* per veure si realment la regla 110 és útil a l'hora de reconèixer imatges o no.

Els resultats del cas base es poden veure en el *Gràfic 23*.



Gràfic 23: Gràfic de l'optimitzador SGD pel cas base

Es pot observar que els resultats són lleugerament pitjors. Això significa que els autòmats cel·lulars sí afegeixen informació a la xarxa neuronal que ajuda a millorar la precisió, però no d'una manera excessiva.

Els resultats obren la porta a investigar més en profunditat les possibilitats de la regla 110 i de quina manera o quines maneres es podria portar a terme per incrementar l'efectivitat.

El codi del sistema es pot veure a https://github.com/avanger9/RC-ECA-110/tree/main/implementacio/mnistproblem

On hi ha un fitxer *Readme* amb els detalls de la programació que s'ha portat a terme i com executar-ho si es vol provar.

12 Conclusió

A partir d'aquest punt s'ha pogut investigar, aprendre i visualitzar totes les fases que constitueixen els autòmats cel·lulars elementals i els reservoris de computació.

La primera fase, la senyal d'entrada on s'ha pogut analitzar i veure 2 senyals d'entrada diferents, tant pel problema del 5 bit on calia mapar l'entrada prèviament de manera aleatòria per donar-li informació i el conjunt del MNIST que es separava en 8 capes d'un bit i on cada capa actuava independent una de l'altre.

La segona fase, la del reservori on s'ha pogut veure com els autòmats substitueixen tota la fase de les connexions neuronals per crear un sistema iteratiu on es segueix una regla per definir tota una nova sèrie de dades amb una complexitat computacional baixa.

La tercera i última fase, la senyal de sortida on s'ha pogut veure diferents mètodes d'aprenentatge i com afectava als resultats.

Per un costat s'ha utilitzat la Regressió Lineal per resoldre el problema del 5 bits on s'ha pogut observar uns resultats molt bons, sobretot perquè es considera un problema difícil per les xarxes neuronals recurrents. S'ha aconseguit uns nivells d'encerts del 100% amb *distractors* molt alts, partint de la marca dels 200 com es mostren en l'experiment inicial de l'*Ozgur Yilmaz*.

Per l'altre banda es tenia el problema del conjunt MNIST, la investigació principal d'aquest projecte. S'ha pogut programar un sistema funcional i entrenar un model que, tot i aconseguir uns nivells de precisió considerables, no s'aproxima a altres tècniques amb nivells molt més alts. Tot i així, és un bon material de partida per aprofundir molt més en el tema ja que per aquest treball la intenció ha sigut fer una investigació dels autòmats, la regla 110 i com actuaven en diferents circumstàncies.

En conclusió s'ha aconseguit la intenció d'aquest projecte, s'ha aprés sobre una tècnica que en els últims anys s'ha investigat cada cop més i s'ha programat un sistema que és capaç de transformar un conjunt d'entrades inicials en autòmats i predir amb exactitud el problema del 5 bit. Tot i que al final el problema del MNIST no ha donat tant bons resultats, la intenció d'aquest projecte no era aconseguir bons o dolents resultats, sinó aconseguir veure com l'autòmat cel·lular era capaç d'aconseguir prediccions.

13 Informe de sostenibilitat

Un cop feta l'enquesta i pensat les preguntes detingudament me n'adono que els meus coneixements en aquesta àrea són gairebé nuls i fins ara no li he posat massa atenció ni he tingut en compte els seus efectes.

Un dels punts més importants que me n'he adonat fent l'enquesta es que en cap moment he pensat que el treball que duc a terme pugui tenir un impacte respecte la sostenibilitat, l'únic criteri que s'ha portat a terme fins ara ha sigut el de buscar la manera de que el software del projecte fos el més ràpid possible.

Moltes de les preguntes em fan plantejar si a la pròpia carrera m'han ensenyat prou o potser caldria remarcar més alguns aspectes claus. Si és cert que durant els anys que estudies es fan actes per explicar i valorar l'impacte social, ambiental i econòmic dels projectes que es duen a terme, però tot i així hi ha moltes preguntes que no les he tingut en compte al meu projecte o directament no les sabia per falta de coneixements.

En conclusió, encara em queda un llarg camí per aprendre i tenir en compte tots els possibles impactes que la feina que faig pugui tenir i aprendre a aplicar-ho de manera correcte en els projectes de curt a llarg termini que dugui a terme.

13.1 Dimensions ambientals

13.1.1 Fita inicial

Les dimensions ambientals d'aquest projecte no s'havien tingut en compte fins al moment de fer l'enquesta ja que al ser un projecte on tota la feina es basa en un nou software doncs no s'havia plantejat la idea.

L'impacte ambiental en aquest projecte serà el consum d'electricitat ja que al ser un projecte on el 100% serà treball en un ordinador (o 2 com s'ha calculat a la secció 1.3 recursos materials) només es tindran en compte les hores assignades al projecte per tant en aquest punt no hi ha cap més opció per millorar l'impacte mediambiental.

13.1.2 Fita final

A l'acabar el projecte s'ha utilitzat l'ordinador de sobretaula més hores de les assignades al projecte ja que inicialment es tenia en compte fer 4 hores al dia i algunes hores més al cap de setmana, però la quantitat de temps necessària per calcular els processos ha fet que el cap de setmana l'ordinador es mantingués obert i treballant, per tant al consum energètic se li poden sumar unes 40 hores més. Pel que fa a un ordinador portàtil no s'ha utilitzat, per tant en aquest punt no hi hagut cap cost.

El total del temps calculat inicialment va ser de 450 hores, més les 40 hores que s'afegeixen a l'acabar el projecte. De mitjana un ordenador gasta entre uns 150/200 Watts/hora [52].

$$200 \frac{watts}{hora} * 450\ hores = 9000\ Watts$$

Per tant el cost total pel programador ha sigut de 9000 Watts utilitzant el valor màxim de la mitja.

$$200 \frac{watts}{hora} * 350\ hores = 7000\ hores$$

El director del treball ha utilitzat 350 hores, per tant el cost total ha sigut de 7000 Watts.

El cost ambiental final del projecte s'ha tingut en compte ja que en tot moment l'ordinador de sobretaula ha estat encès per portar a terme el projecte i en els moments que no es treballava perquè duia a terme els càlculs del projecte es mantenia en baix consum.

La vida útil del projecte no es significativa ja que no reduirà ni empitjorarà l'impremta ecològica, però en general es un projecte d'un cost baix: no s'ha utilitzat diàriament moltes hores ni tampoc han participat més persones que el director del projecte i el programador, per tant és un projecte de cost baix.

Per un projecte futur es tindrà ja en compte el consum elèctric ja que aquest treball ha servit com a base per aprendre i entendre que cal mesurar com s'utilitzen els aparells electrònics per millorar la dimensió ambiental.

13.2 Dimensió econòmica

13.2.1 Fita inicial

La dimensió econòmica és un dels punts que més s'ha tingut en compte a l'hora de portar a terme aquest projecte i els costos s'han calculat com si fos un projecte a escala real d'una empresa. S'ha calculat tots els costos tenint en compte els costos de personal, material i les possibles contingències.

Aquest projecte intenta donar un nou punt de vista per donar una altre solució a un problema, per tant si l'experiment dona molt bons resultats amb un cost més baix que altres solucions ajudaria a abaratir els costos en altres projectes que necessitessin aquest software.

13.2.2 Fita final

La previsió econòmica s'ha ajustat als costos calculats a l'apartat *8: Pressupost* i no hi ha cap contingència. A més, s'ha aconseguit reduir el cost ja que l'ordinador portàtil no va caler comprar-lo.

El cost final del projecte era de 11.405,88€ i si se li descompta el cost del portàtil i les contingències queda 9.578,62€.

13.3 Dimensions socials

13.3.1 Fita inicial

La dimensió social serà el punt on més aprendré respecte un projecte de software a nivell real (per comparar el que s'ha vist fins ara en el transcurs del grau d'*Enginyeria Informàtica*), les seves característiques i el desenvolupament. Seran mesos d'aprenentatge en tots els sentits en la definició d'un projecte a gran escala.

Els problemes de classificació fa molts anys que s'investiguen i apareixen noves tècniques que milloren les actuals o aporten un sistema més innovador respecte altres solucions. En aquest cas s'intentarà avaluar millorar les solucions existents, per tant crec que si es necessari aquest projecte i em permetrà aprendre a analitzar amb major rigor les dimensions d'un gran projecte.

13.3.2 Fita final

Durant la realització del projecte s'ha pogut estudiar amb profunditat tot els articles previstos a la planificació i d'aquesta manera s'han assolit els coneixements necessaris per portar a terme tot el software necessari per a què es pogués veure el funcionament del sistema, entendre com funciona i si pot tenir alguna rellevància en futurs projectes on s'aprofundeixi més en el tema.

A nivell professional queda la pregunta a l'aire de si realment pot ser un mètode eficaç i si aprofundint més en el tema es pot trobar solucions que facin més viable el projecte, però pot ser útil per si alguna persona o col·lectiu vol seguir investigant ja té una base d'on pot començar.

14 Referències

[1] "Elementary Cellular Automaton. A Theory On How Simple Structures… | by Jesus Najera | Cantor's Paradise | Medium." https://medium.com/cantors-paradise/elementary-cellular-automaton-e27e3d1008d9 (accessed Nov. 15, 2020).

[2] "The Origins of Cellular Automata." http://psoup.math.wisc.edu/491/CAorigins.htm (accessed Nov. 22, 2020).

[3] "Cellular Automaton -- from Wolfram MathWorld." https://mathworld.wolfram.com/CellularAutomaton.html (accessed Nov. 22, 2020).

[4] P. Page, "Paul's Page of Conway's Life Miscellany." http://www.radicaleye.com/lifepage/ (accessed May 07, 2021).

[5] "Four Classes of Behavior: A New Kind of Science | Online by Stephen Wolfram [Page 231]." https://www.wolframscience.com/nks/p231--four-classes-of-behavior/ (accessed May 07, 2021).

[6] S. Wolfram, "Tables of Cellular Automaton Properties," 1983.

[7] T. Neary and D. Woods, "P-completeness of cellular automaton Rule 110 ⋆." Accessed: Jun. 17, 2021. [Online]. Available: http://www.bcri.ucc.ie/.

[8] "¿Qué es el Reservoir Computing? - YouTube." https://www.youtube.com/watch?v=3gA4ML3TUao&ab_channel=InstitutodeFísicaInterdisciplinarySistemasComplejos%28IFISC%29 (accessed Nov. 22, 2020).

[9] F. Hadaeghi, "Reservoir Computing Models for Patient-Adaptable ECG Monitoring in Wearable Devices," *arXiv*, 2019.

[10] C. Sun, M. Song, S. Hong, and H. Li, "A REVIEW OF DESIGNS AND APPLICATIONS OF ECHO STATE NETWORKS," 2020.

[11] N. : Oladipupo and G. Gbenga, "RESEARCH ON THE CONCEPT OF LIQUID STATE MACHINES."

[12] "Red neuronal de impulsos - Wikipedia, la enciclopedia libre." https://es.wikipedia.org/wiki/Red_neuronal_de_impulsos (accessed May 08, 2021).

[13] "MNIST Benchmark (Image Classification) | Papers With Code." https://paperswithcode.com/sota/image-classification-on-mnist (accessed Nov. 08, 2020).

[14] A. Morán, C. F. Frasser, and J. L. Rosselló, "Reservoir Computing Hardware with Cellular Automata," *arXiv*, Jun. 2018, Accessed: Jan. 27, 2021. [Online]. Available: http://arxiv.org/abs/1806.04932.

[15] A. Moran, C. F. Frasser, M. Roca, and J. L. Rossello, "Energy-Efficient Pattern Recognition Hardware with Elementary Cellular Automata," *IEEE Trans. Comput.*, vol. 69, no. 3, pp. 392–401, 2020, doi: 10.1109/TC.2019.2949300.

[16] O. Yilmaz, "Reservoir Computing using Cellular Automata," pp. 1–9, 2014, [Online]. Available: http://arxiv.org/abs/1410.0162.

[17] J. Fernández González, "Introducción a las metodologías ágiles Otras formas de analizar y desarrollar."

[18] K. Schwaber and J. Sutherland, "The Scrum Guide The Definitive Guide to Scrum: The Rules of the Game," 2020.

[19] "¿Qué es la metodología Agile y por qué está de moda? | Progressa Lean." https://www.progressalean.com/metodologia-agile/ (accessed Jan. 11, 2021).

[20] "Git." https://git-scm.com/ (accessed Jan. 13, 2021).

[21] "GitHub." https://github.com/ (accessed Mar. 18, 2021).

[22] Facultat d'Informàtica de Barcelona, "Normativa Del Treball Final De Grau Del Grau En Enginyeria Informàtica De La Facultat D'Informàtica De Barcelona," 2020, [Online]. Available: https://www.fib.upc.edu/sites/fib/files/documents/estudis/normativa-tfg-mencio-addicional-gei-br.pdf.

[23] E. T. Bye, “Investigation of Elementary Cellular Automata for Reservoir Computing,” no. June, 2016.
[24] M. Margem and O. S. Gedik, “Feed-forward versus recurrent architecture and local versus cellular automata distributed representation in reservoir computing for sequence memory learning,” *Artif. Intell. Rev.*, vol. 53, no. 7, pp. 5083–5112, 2020, doi: 10.1007/s10462-020-09815-8.
[25] H. Jaeger, “Long Short-Term Memory in Echo State Networks: Details of a Simulation Study,” *Report*, no. 27, p. 29+, 2012, [Online]. Available: http://minds.jacobs-university.de/pubs%5Cnhttp://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Long+Short-Term+Memory+in+Echo+State+Net-+works+:+Details+of+a+Sim-+ulation+Study#0.
[26] “Sueldo: Jefe De Proyecto | Glassdoor.” https://www.glassdoor.es/Sueldos/jefe-de-proyecto-sueldo-SRCH_KO0,16.htm (accessed Feb. 03, 2021).
[27] “Salario de un Desarrollador/a de software en España.” https://es.indeed.com/career/desarrollador-de-software/salaries?from=top_sb (accessed Feb. 03, 2021).
[28] “Home - Aurea Coworking - Gala Placidia.” https://www.aureacoworking.com/ca/ (accessed Feb. 07, 2021).
[29] “Aportaciones de esta actualización INFORMACIÓN CIENTÍFICA-TÉCNICA Enfermedad por coronavirus, COVID-19.” Accessed: Feb. 09, 2021. [Online]. Available: https://www.aemps.gob.es/https://www.mscbs.gob.es/profesionales/saludPublica/prevPromocion/vacunaciones/covid19/vacunasCovid19.htm.
[30] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.
[31] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber, “LSTM: A Search Space Odyssey,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017, doi: 10.1109/TNNLS.2016.2582924.
[32] “Python Release Python 3.9.0 | Python.org.” https://www.python.org/downloads/release/python-390/ (accessed May 10, 2021).
[33] “Overview — NumPy v1.20 Manual.” https://numpy.org/doc/stable/ (accessed May 10, 2021).
[34] “scikit-learn: machine learning in Python — scikit-learn 0.24.2 documentation.” https://scikit-learn.org/stable/ (accessed May 10, 2021).
[35] “Python vs C | Benchmark | Sep, 2020 | The Startup.” https://medium.com/swlh/two-programming-languages-same-algorithm-c65992801af1 (accessed May 10, 2021).
[36] S. Nichele and M. S. Gundersen, “Reservoir Computing Using Non-Uniform Binary Cellular Automata.”
[37] “sklearn.linear_model.LinearRegression — scikit-learn 0.24.2 documentation.” https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html (accessed May 11, 2021).
[38] S. Nichele and A. Molund, “Deep reservoir computing using cellular automata,” *arXiv*, 2017.
[39] S. Nichele and M. S. Gundersen, “Reservoir computing using nonuniform binary cellular automata,” *Complex Syst.*, vol. 26, no. 3, pp. 225–246, 2017, doi: 10.25088/ComplexSystems.26.3.225.
[40] “Matplotlib: Python plotting — Matplotlib 3.4.2 documentation.” https://matplotlib.org/ (accessed May 22, 2021).
[41] “Compuertas lógicas y sus tablas de verdad - Ingeniería Mecafenix.” https://www.ingmecafenix.com/electronica/compuertas-logicas/ (accessed May 14, 2021).
[42] “Max-pooling / Pooling - Computer Science Wiki.”

https://computersciencewiki.org/index.php/Max-pooling_/_Pooling (accessed May 14, 2021).
[43] “Softmax function - Wikipedia.” https://en.wikipedia.org/wiki/Softmax_function (accessed May 22, 2021).
[44] “Why One-Hot Encode Data in Machine Learning?” https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/ (accessed May 22, 2021).
[45] “Keras: the Python deep learning API.” https://keras.io/ (accessed May 22, 2021).
[46] “The Sequential model.” https://keras.io/guides/sequential_model/ (accessed May 29, 2021).
[47] “Dense layer.” https://keras.io/api/layers/core_layers/dense/ (accessed May 29, 2021).
[48] “Norma matricial - Wikipedia, la enciclopedia libre.” https://es.wikipedia.org/wiki/Norma_matricial (accessed Jun. 19, 2021).
[49] “Optimizers.” https://keras.io/api/optimizers/ (accessed May 27, 2021).
[50] “MNIST CNN optimizer comparison with tensorflow.keras | by Jay | Medium.” https://onlytojay.medium.com/mnist-cnn-optimizer-comparison-with-tensorflow-keras-163735862ecd (accessed May 29, 2021).
[51] “Epoch vs Batch Size vs Iterations | by SAGAR SHARMA | Towards Data Science.” https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9 (accessed May 29, 2021).
[52] “How much power does a computer use? And how much CO2 does that represent? – Energuide.” https://www.energuide.be/en/questions-answers/how-much-power-does-a-computer-use-and-how-much-co2-does-that-represent/54/ (accessed Jun. 10, 2021).