Universitat de Girona
Escola Politècnica Superior

Treball final de grau

Estudi: Grau en Disseny i Desenvolupament de Videojocs

Títol: Tècniques de *Deep Learning* per la síntesi d'imatges. Aplicació a la restauració d'imatges.

Document: Memòria

Alumne: Bryan Miguel Perez Ruchat

Tutor: Dr. LLADO BARDERA, XAVIER

Departament: ARQUITECTURA I TECNOLOGIA DE COMPUTADORS

Àrea: ARQUITECTURA I TECNOLOGIA DE COMPUTADORS

Convocatòria (mes/any) Juny 2022

Índex

Índex de Figures

Índex de taules

1 Introducció

La transferència d'estils i la millora de qualitat d'imatges són tècniques que s'estan començant a utilitzar en l'àmbit dels videojocs, principalment perquè s'està arribant a un punt en què certs mètodes es poden implementar en temps real i, per tant, es poden utilitzar en un videojoc. També cal remarcar que els motors de jocs més grans ja estan incloent eines per poder utilitzar xarxes neuronals (tècniques de *Deep Learning*) com a post processat en el joc. Aquestes tècniques estan apareixent en els videojocs al mateix temps que certes companyies estan fent que es pugui jugar a jocs que estiguin corrent al servidor de la companyia i l'usuari només necessita una pantalla i una manera de registrar els inputs.

Aquests sistemes com el *Stadia* [1] o el *GeForce Now* [2] requereixen un gran ample de banda per enviar la imatge als jugadors i molt baixa latència per tal d'evitar problemes de *lag*[1]. La temàtica que s'estudiarà en aquest TFG, podria emprar-se en aquests casos per reduir la quantitat d'informació a enviar a cadascun dels jugadors, fent el sistema més accessible i més estable. La Figura 1 representa el funcionament d'aquests sistemes, la part encerclada és la part que es pot relacionar amb aquest projecte, com es veurà més endavant.

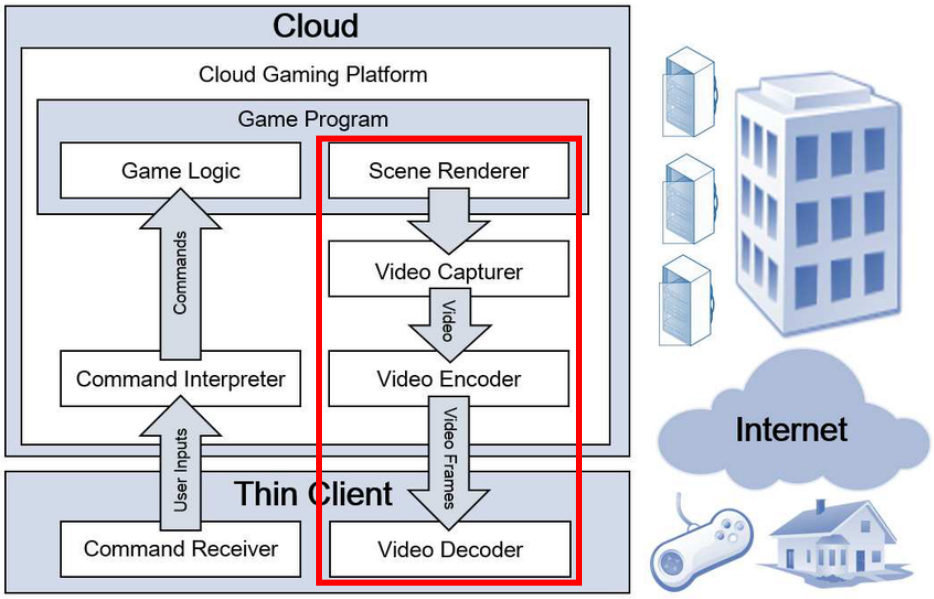


Figura 1. Diagrama cloud gaming Font: https://mobillegends.net/online-game-cloud-network-optimization-scheme-develop-paper

[1] Retard excessiu de quan s'envia la informació fins que arriba

Les tècniques de *Deep Learning* estan millorant cada any, creant noves possibilitats sobretot en la síntesi d'imatges. Una de les aplicacions que en els darrers anys s'han desenvolupat bons mètodes seria el de restaurar imatges antigues de forma automatitzada. Aquestes tècniques poden solucionar diferents situacions de les fotografies antigues, entre altres, eliminació de soroll, pintar-les o augment de resolució. Per aquest treball, ens centrarem en l'eliminació de soroll i posar color a les imatges en blanc i negre usant tècniques de *Deep Learning.* Amb el coneixement obtingut solucionant aquest problema acotat, es podran desenvolupar mètodes pels videojocs per solucionar problemes similars.

1.1 Síntesis d'imatges

La síntesi d'imatges consisteix en generar noves imatges donat unes dades d'entrada. En el nostre cas es realitzaran transformacions directament a les imatges. Aquest conjunt de tècniques tenen nombroses aplicacions com poden ser la transferència d'estils o la restauració d'imatges. La que s'ha estudiat en aquest TFG són les basades en la restauració d'imatges i en la generació sintètica d'informació de color , veure Figura 2.

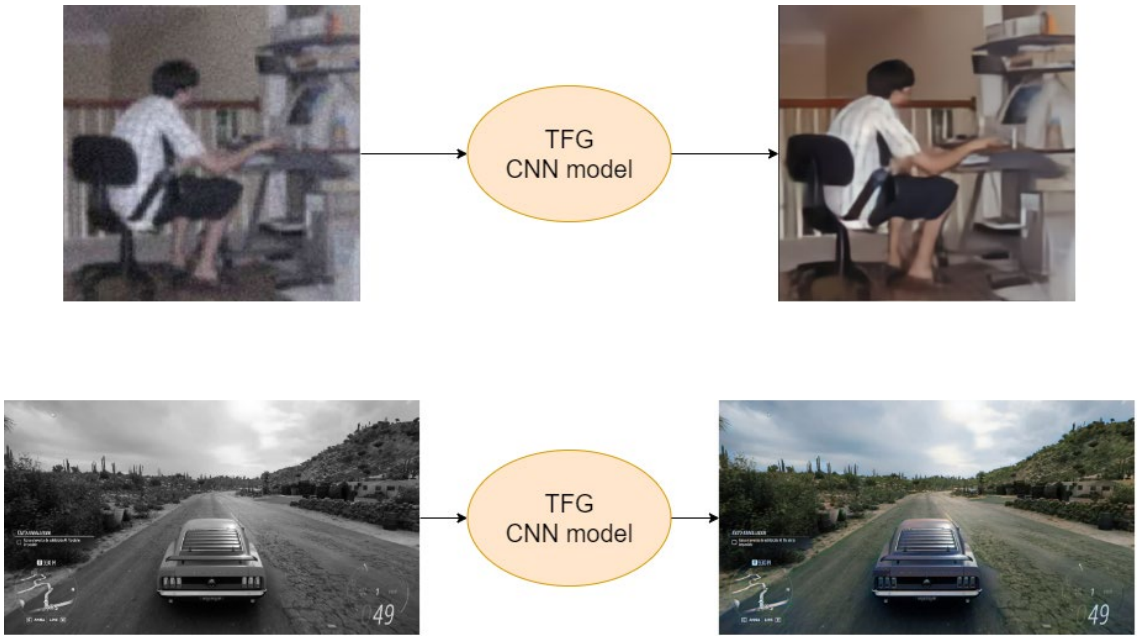


Figura 2. Representació de l'objectiu del TFG. A) Millora de la qualitat d'imatges B) Posar colors a les imatges

La restauració artística consisteix en reparar o conservar obres d'art amb l'objectiu de preservar la cultura de l'època que s'han creat. El principal objectiu és retornar-li la seva antiga funció o significat que, en el pas del temps hagi pogut perdre. La part de restauració de pintures és la més similar al que es fa en aquest TFG. En la Figura 3 es mostra un cas de restauració de pintura.

En l'actualitat, ha aparegut un altre corrent que treballa de manera similar en fotografies antigues que es digitalitzen per restaurar i, a la vegada poder mantenir la informació original. Aquest TFG es centrarà en aquest repte de restaurar imatges digitalitzades utilitzant tècniques novedoses d'intel·ligència artificial basades en *Deep Learning*, que com es veurà, es podrien emprar també en el camp dels videojocs.

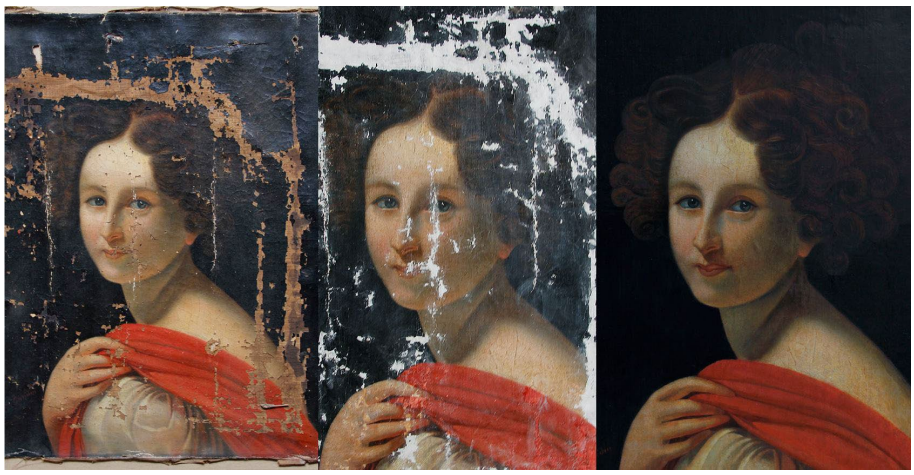


Figura 3. Restauració de pintura. A) Pintura amb danys B) Tractament previ C) Pintura restaurada. Font: https://arte-restauracion.es/restauracion-pintura-retrato-de-mujer/

Figura 4. Restauració fotografia. A) Fotografia antiga amb dants B) Tractament de danys estructurals en blanc i negre C) Resultat final amb colors. Font: https://www.paredro.com/restauracion-digital-de-fotografias-su-importancia-y-ejemplos/

La restauració d'imatges antigues es diferencia de la restauració de pintures no només per la tècnica que s'utilitza sinó que es tracten problemes diferents, veure les figures Figura 3 i Figura 4 on es mostra un exemple de cada cas. La principal diferència és que en la restauració de pintures, es tracta el pas del temps mentre que en la restauració de fotografies es tracta la baixa qualitat fotogràfica que es tenia al moment de fer-la o ratllades o doblegs que s'hagin pogut fer posteriorment, és a dir, un intenta tornar a portar l'obra al punt de creació i l'altra intenta portar-ho al moment actual.

Els problemes que s'han de tractar al restaurar les imatges es poden diferenciar entre estructurals i no estructurals [3].

- Els danys estructurals seran tots aquells que no depenen de la càmera sinó de com s'ha guardat la imatge i si ha patit molt de desgast respecte a l'estructura general de la fotografia, els exemples més clars és si es tenia la fotografia plegada s'hagi creat una línia blanca i, per tant, s'hagi perdut informació d'aquella zona. També serien danys estructurals si falta una part de la imatge per un forat o faltin puntes o similars. Veure Figura 5 A per veure un exemple d'imatge amb danys estructurals.

- Els danys no estructurals són els danys que no es podien solucionar al moment de la creació i que, el pas del temps i com s'han guardat no afecta o quasi no afecta en aquesta part. Els exemples clars serien el soroll que la càmera hagi pogut introduir al fer la fotografia, la profunditat de color que tenia la càmera i, per tant, la fotografia té menys colors o colors menys vius. Veure Figura 5 B per veure un exemple d'imatge amb danys no estructurals.

Figura 5. Diferència entre danys estructurals i danys no estructurals. A) Fotografia amb danys estructurals B) Fotografia amb danys no estructurals. Font: http://www.prweb.com/releases/2004/08/prweb149074.htm i https://themindcircle.com/creepy-vintage-photos/2/

Per aquest treball, ens centrarem en tractar els problemes no estructurals. Per fer-ho partirem

de dues estratègies. La primera és la restauració eliminant el soroll de les imatges, amb una arquitectura de *Deep Learning* i l'altra part es basa en passar les imatges en blanc i negre a color per tenir imatges més vives i més actuals, també amb una tècnica de *Deep Learning*.

Per poder dur a terme el treball es necessitarà una gran quantitat de dades, en concret d'imatges antigues i imatges actuals per poder dur a terme l'entrenament. D'imatges antigues en necessitarem de dos tipus també, a color i en blanc i negre. Per aconseguir les dades es combinaran diferents estratègies: per a les imatges antigues en blanc i negre es buscaran bases de dades públiques, i per a les imatges a color es buscaran pel·lícules antigues, ja que no s'han trobat bases de dades de fotografies antigues a color. Per les imatges actuals s'ha buscat un data set de classificació que conté moltes imatges de diferents objectes i sers vius per tal de tenir un bon set de dades.

Finalment per treballar en la part més similar en l'àmbit dels videojocs, es tractaran vídeos a nivell de gris i es passaran a color. L'objectiu seria desenvolupar el que es veu en la Figura 2 B, tenint un sistema de *Deep Learning* per tal de poder obtenir imatges a color donades imatges en blanc i negre, revisant problemes que poden sorgir al tractar imatges com la coherència temporal entre els diferents frames. Aquestes proves finals, ens permetran apropar a l'ús que es podria dur a terme per un sistema com el *Stadia*.

1.2 Motivacions

Amb l'augment de la capacitat computacional, s'estan podent crear ordinadors potents que ens permeten entrenar xarxes neuronals. La base teòrica ve dels anys 80 i ara es comencen a fer avenços cada pocs anys. Un dels quals el *paper Bringing Old Photos Back To Life* [3] d'uns anys enrere que mostra resultats fascinants pel problema que es vol tractar en aquest TFG. Ara mateix ja es poden simular entorns usant IA, generar animacions per diferents personatges. En un futur que es podrà arribar a fer? Amb aquesta pregunta en ment s'ha volgut aprendre com es desenvolupava un projecte de IA, concretament la col·lecta de dades, les diferents arquitectures i les seves funcions i l'anàlisi dels resultats.

Fins fa pocs anys les tècniques de *Deep Learning* eren molt costoses, computacionalment parlant, i no es podien realitzar en un projecte com un videojoc que necessita funcionar en temps real. Per aquesta raó s'ha decidit investigar aquestes tècniques, ja que, en un futur no molt llunyà es podran usar als videojocs i crear noves modalitats.

1.3 Propòsits i objectius

L'objectiu d'aquest TFG és entendre el funcionament de les xarxes neuronals i com funcionen els algoritmes de *Deep Learning* per al processament d'imatges. Un dels problemes que es tractaran és la restauració d'imatges antigues, eliminant el soroll que puguin tenir. També es vol analitzar, amb la síntesi d'imatges, la posada de color a imatges en blanc i negre. Amb aquests models es veurà com es poden integrar dins dels videojocs per tenir entorns més realistes, generant imatges i textures, que en el cas de les plataformes de *cloud computing* poden ajudar en el procés de vídeo *encoding* i *decoding*.

Els subobjectius del treball serien, per tant:

- La col·lecta de dades, és la primera part i molt important, s'han de buscar diferents tipus d'imatges per tal de poder entrenar les xarxes neuronals. Per aquest treball, necessitarem imatges actuals en color i imatges antigues tant a color com en blanc i negre. També es necessitarà recollir fotogrames de videojocs, per fer-ho es gravaran escenes dins del joc.

- Dissenyar i implementar una estratègia de *Deep Learning* per corregir soroll, cercant un marc de treball per tal de desenvolupar el projecte, entendre el seu funcionament per tal de poder-hi realitzar proves restaurant les imatges antigues.

- Dissenyar i implementar una estratègia *Deep Learning* per pintar imatges, cercar un mètode que permeti posar-hi color a les imatges en blanc i negre. Permetria pintar les imatges en blanc i negre que s'han obtingut després d'eliminar el soroll amb la primera xarxa. També serà utilitzat en la *demo* per tal de demostrar el seu funcionament en temps real.

- Realització del conjunt d'experiments a les dues xarxes, canviant els paràmetres per tal de veure si s'obtenen millors resultats. En combinació amb l'anàlisi de resultats permetrà establir quins han sigut els millors paràmetres del conjunt de proves que s'hagin desenvolupat i del problema en concret que s'estigui tractant.

- Treballar el pintat d'imatges en vídeos, investigar com utilitzar la xarxa que s'ha emprat per donar color a les imatges per emprar-ho en vídeos, comprovar si s'han de realitzar canvis.

- Realització de la *demo*, utilitzant un motor de jocs, ajuntar els diferents elements per tal de demostrar el seu ús en el camp dels videojocs. Per fer-ho, s'utilitzarà Unity i es dissenyarà un joc en que es pugui transferir els colors d'un joc ja existent.

- Anàlisi de resultats, amb avaluacions quantitatives i qualitatives, per tal de poder comparar els mètodes i observar que ha aportat millors resultats.

- Redacció de la memòria, reunir tota la informació així com il·lustrar els resultats obtinguts a un document.

1.4 Planificació

Aquest treball té com a principal objectiu aprendre el funcionament de xarxes neuronals. Per aquesta raó està dividit en tres parts, la recerca d'informació, tant de dades com de tècniques, de la d'implementació i/o modificació d'aquests algoritmes i, finalment l'anàlisi dels resultats obtinguts.

Primerament, es va realitzar la recerca de tema, parlant amb el tutor d'aquest TFG el Dr Xavier Lladó Bardera per tal de tenir una guia de quin tema seria una bona idea per aprendre sobre aquest tema. Aquesta tasca inicial va ser del dia 21 de maig 2021 fins al 13 Agost del mateix any.

Un cop es tenia el tema triat s'havien de buscar dades, per aquesta tasca s'havia posat com a data màxima el mes de setembre. L'objectiu era tenir aquesta part abans del començament de classes. En paral·lel es va dur a terme una recerca d'entorns i possibles eines per dur a terme la tasca final de restaurar les imatges.

Durant el primer semestre l'objectiu era tenir l'entorn funcionant i dur a terme un conjunt de proves per saber si es tenia la quantitat de dades suficients i quins problemes podrien sorgir. El que quedava de Gener i Febrer bàsicament van estar dedicats a solucionar aquests problemes i fer els entrenaments de les diferents xarxes.

A partir de meitats de Febrer es tindria almenys un dels dos entorns en funcionament per tal de poder començar alguns tests. Tenint en compte els potencials problemes que podrien sorgir, es va planificar de començar aquesta tasca al Febrer fins al Març per tal de poder començar l'anàlisi de resultats.

Al mes de març es començarà a realitzar l'anàlisi de resultats com la redacció de la memòria per tal de no tenir-ho tot junt al final. L'objectiu és tenir l'anàlisi dels resultats acabats a finals d'abril així com tenir una *demo* funcional a finals d'abril.

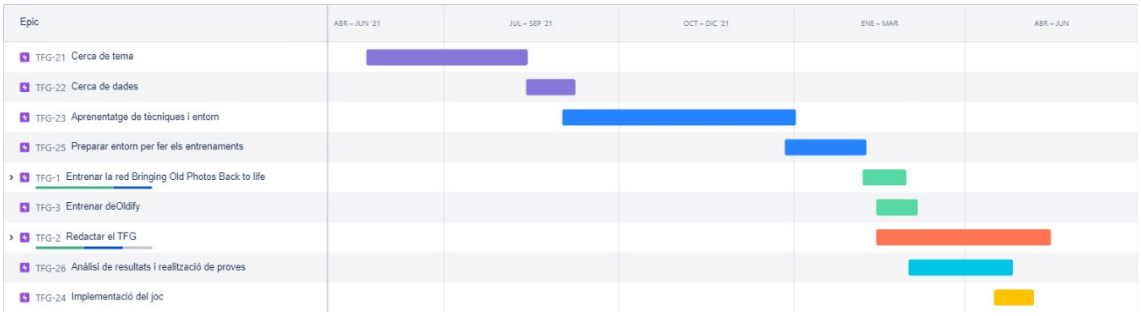


Figura 6. Planificació, full de ruta del Jira

Figura 7. Llegenda planificació

Amb aquesta planificació quedaria un mes per tal de preparar els vídeos mostrant els resultats obtinguts així com finalitzar tasques que puguin quedar pendents. Per tal de tenir-ho tot organitzat s'ha emprat el Jira[2]. Com s'ha emprat el Jira més especificat queda explicat en el següent apartat

[2] https://www.atlassian.com/software/jira

1.5 Continguts de la memòria

Aquest treball està dividit en els següents apartats:

- **Estudi de viabilitat**: S'exposarà la viabilitat del treball, explicant on es pot emprar i perquè és viable fer-ho ara mateix.
- **Desenvolupament**: En l'apartat de desenvolupament s'explicaran els pilars per poder entendre el treball i així com els requisits tecnològics que hi ha.
- **Implementació**: En la implementació s'exposarà com estan organitzades les arquitectures que s'han usat per fer el treball així com els altres programes o modificacions que s'hagin fet.
- **Proves i resultats**: En aquest apartat s'explicaran les proves realitzades i quins resultats s'han obtingut en cada prova per poder generar conclusions.
- **Conclusions**: Les conclusions serien un resum de què s'ha aconseguit segons les proves i resultats obtinguts.
- **Treball futur**: En el treball futur s'explicarà, segons els resultats obtinguts, millores que es podrien fer en el projecte.
- **Bibliografia**: La bibliografia contindrà el recull de fonts d'informació que s'han utilitzat per fer el treball

1.6 Marc de treball i conceptes previs

Abans de parlar de les decisions que s'han pres al treball caldria posar certs punts en context. Aquest apartat està destinat a fer aquesta posada en context explicant els termes que aniran sortint al llarg del treball.

Per començar caldria explicar que és la IA[3] en conceptes més generals, una IA intenta dotar d'intel·lecte a una computadora de manera que rebent inputs del seu entorn pugui prendre una decisió que l'apropi al seu objectiu. Això permet obtenir una manera de solucionar certs problemes de forma més eficient del que es pot fer amb els algoritmes actuals. Amb aquestes

[3] Intel·ligència artificial

tècniques, els problemes *NP Hard*[4] es poden passar per un dels molts algoritmes de IA per tal de trobar-hi una solució.

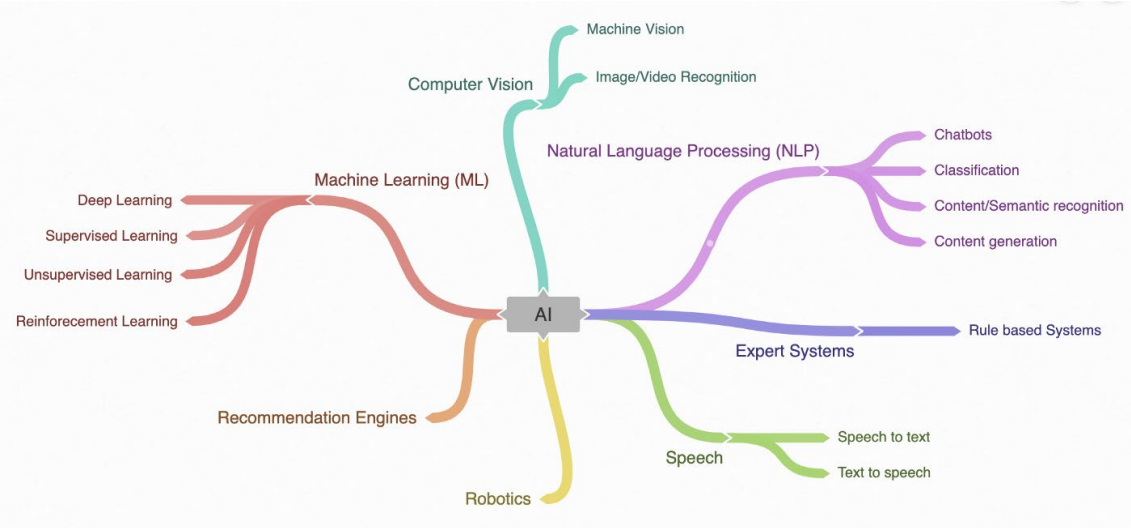


Figura 8. Branques de la IA. Font: https://medium.com/@bharathkumar.h/artificial-intelligence-101-tech-terms-explained-for-non-tech-folks-3190d7e711cd

Com podeu veure hi ha moltes branques dins de la intel·ligència artificial, la que en aquest treball ens interessa és la de *Machine Learning* i, més concretament la part de *Deep Learning*.

1.6.1 Què és *Machine Learning* i perquè no és *Computer Vision ?*

La part de *Machine Learning* es centra en crear models i algoritmes que, després d'haver passat un procés d'entrenament en el que han rebut una sèrie de dades de mostra perquè es pugui crear un model específic per a poder fer una predicció o prendre decisions. Aquesta branca es diferencia de la de visió per computador, ja que la segona, no es basa en tenir un aprenentatge sinó que després d'analitzar un problema concret s'apliquen una sèrie d'operacions matemàtiques a la imatge per tal d'extreure informació o per modificar aquesta imatge. Tot i aquesta diferència, poden haver-hi casos en què problemes típics de la visió per computador s'acabin solucionant en altres àmbits de la IA com és el cas de la restauració d'imatges que la visió per computador empraria filtres per solucionar aquest problema mentre que la branca de M*achine Learning* utilitzaria xarxes neuronals per dur a terme aquesta tasca.

1.6.2 *Deep Learning*

El *Deep Learning* és una branca del *Machine Learning* en què s'utilitzen xarxes neuronals per solucionar el problema en concret. Les xarxes neuronals són sistemes computacionals inspirats

[4] https://mathworld.wolfram.com/NP-HardProblem.html

en el funcionament biològic d'un cervell animal. Per aquesta raó està constituït de les mateixes parts: neurones, axons i dendrites.

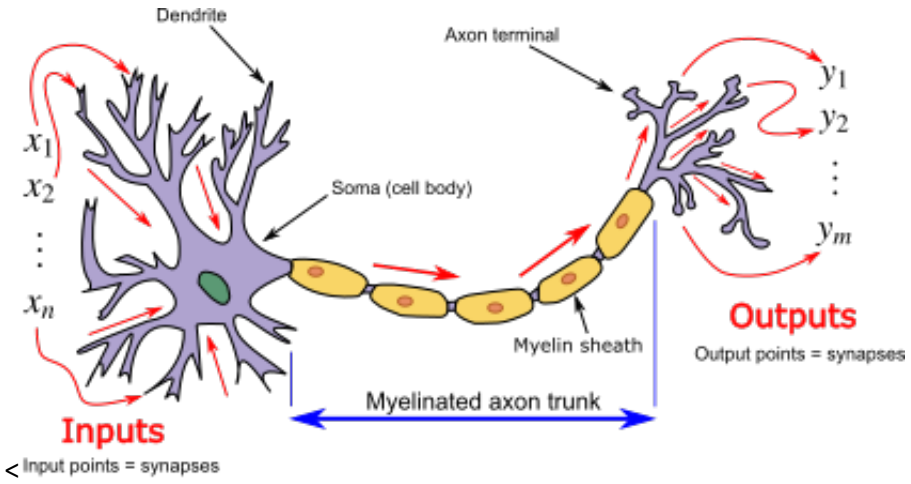


Figura 9. Representació d'una neurona. Font: https://en.wikipedia.org/wiki/Artificial_neuron

Per entendre com funciona una xarxa neuronal primer haurem d'entendre com funciona el seu component principal, una neurona. En la imatge es poden veure les diferents parts nomenades anteriorment. La primera seria la de les dendrites de la neurona on es concentren els seus inputs que anomenarem X, la neurona és el centre de processament on s'opera amb els inputs, per tant, podem nomenar a la neurona com a F de funció i, finalment tenim l'output o outputs de la neurona que definirem com a Y i contactarien en altres dendrites o, en el cas de ser la última neurona seria l'output final. Per tant, es pot definir una neurona com a Y= F(X), és a dir, una neurona és com una funció qualsevol. Amb el que es té fins ara l'aprenentatge de la xarxa només podria modificar la funció F ja que la resta són inputs i outpus. En els *fully-connected layers* [4], les variables que es té són el biaix de la neurona i, per cadascun dels inputs se li atribueix un pes de manera que la funció finalment seria Y = b + W*X en què W és un vector amb els pesos, X és un vector d'inputs, b és el biaix de la neurona i Y és l'output de la neurona. En la natura no sempre s'envia senyal a la següent neurona i, en aquestes neurones tampoc, el que es fa és passar-ho per una funció d'activació que decidirà si s'envia el senyal o no a la següent neurona i la seva magnitud.

1.6.3 Funcions d'activació

Les funcions d'activació són unes funcions que reben l'output d'una neurona per tal d'indicar que la neurona s'ha activat i el valor que retornarà finalment. Les funcions més comunes són les *ReLU*[5] i les sigmoide. [5]

[5] *Rectified linear unit*, rectificador lineal

La *ReLU* simplement permet que només s'activi en valors positius. Per fer-ho simplement agafa el màxim entre el valor d'entrada i 0.

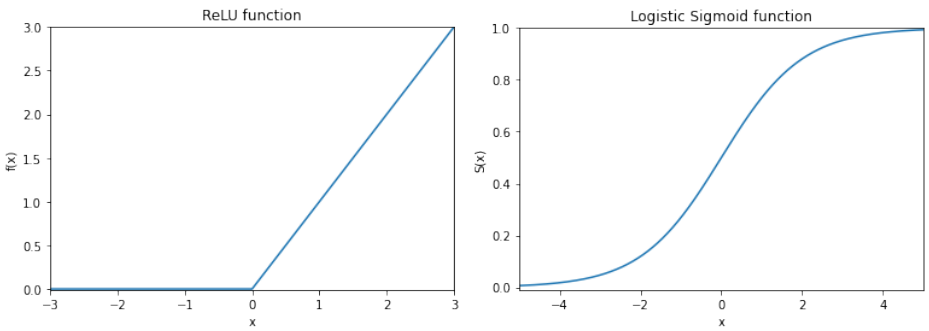


Figura 10. Funció ReLU i funció sigmoide. Font: https://deepai.org/machine-learning-glossary-and-terms/sigmoid-function

La sigmoide simplement aplica una funció logística sigmoide que permet centrar tots els outputs entre 0 i 1. La funció està centrada a 0 de manera que en aquesta els valors negatius queden entre 0 i 0.5 i els positius de 0.5 a 1.

1.6.4 Xarxes neuronals

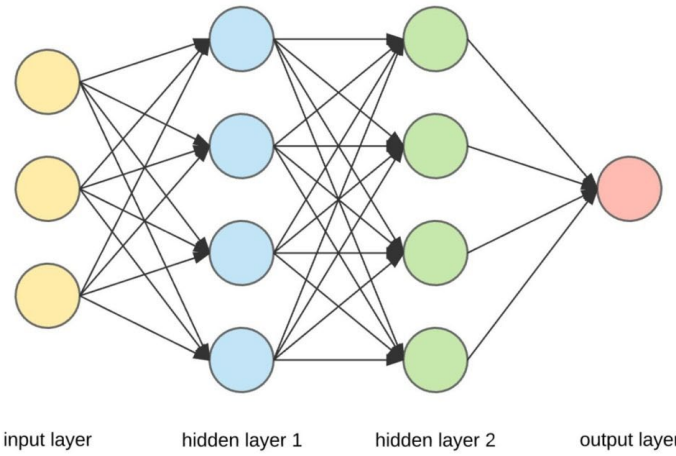


Figura 11. Xarxa neuronal. Font: https://orgs.mines.edu/daa/blog/2019/08/05/neural-networks-mva/

Per la part de xarxes neuronals, el que es fa és combinar moltes neurones, normalment es combinen per capes, partint de la d'entrada que és la que rep les dades d'entrada passant per una o més capes ocultes i finalment arribant a la capa de sortida. Les xarxes neuronals que tenen moltes capes ocultes s'anomenen *deep neural network* per diferenciar-les de les *artificial neural networks*. Cada combinació de com es connecten les neurones es diu arquitectura i, un cop es té l'arquitectura dissenyada es pot passar a realitzar l'entrenament de la xarxa neuronal. L'aprenentatge es basa en modificar el valor del biaix de la neurona i els

pesos de cadascuna de les neurones, per tant, en la xarxa completa serà modificar els pesos de cadascuna de les neurones fins arribar a un punt en què la xarxa doni bons resultats sense tenir un problema d'*overfitting*[6]. Les xarxes neuronals es poden entrenar de dues maneres, de manera supervisada o no supervisada.

L'aprenentatge supervisat es tracta de donar-li a la xarxa neuronal parelles de dades d'entrada i la sortida corresponent de manera que l'objectiu de la xarxa és que vagi fent el *mapping*[7] entre els dos, es diu supervisat, per tant, perquè es té el resultat objectiu i, per tant, es pot mesurar fàcilment el bé que ho està fent. En l'aprenentatge no supervisat, en canvi, només se li dona *l'input*[8] a la xarxa neuronal i es genera *l'output*[9] en què la xarxa intenta imitar les dades per tal d'obtenir el mateix resultat. Per aquest treball tindrem aprenentatge supervisat, ja que, donat que s'utilitzen codificadors i descodificadors *l'output* del descodificador hauria de ser igual a *l'input* del codificador.

1.6.5 Què és una convolució?

En les xarxes neuronals que tracten imatges solen tenir capes on es fa l'operació de convolució i algunes també tenen convolució transposada que seria l'operació contrària. L'operació de convolució tenen com a objectiu reduir la quantitat de dades a tractar, és a dir, comprimir la informació. Per fer-ho el que es fa és definir un *kernel*, que és com una matriu quadrada amb diferents valors segons quins sigui el nostre objectiu, per exemple en visió per computador es poden usar per detectar els contorn. L'objectiu és superposar el *kernel* sobre cadascun dels píxels de la imatge menys els contorns en el nostre cas i sumar cadascuna de les multiplicacions de píxel de *kernel* que estigui superposat a la imatge. En visió per computador es pot suposar que els que no es poden accedir són 0 o 1 depenen del *kernel* per no tenir imatges de diferents dimensions. Com a resultat s'obté una imatge, en el nostre cas més petita i, la reducció resultant depèn de la mida del *kernel* emprat.

[6] Quan el model estadístic s'assembla massa a les dades d'entrada i no podrà funcionar bé en dades que no hagi vist. https://www.ibm.com/cloud/learn/overfitting
[7] Mapeig
[8] Entrada
[9] Sortida

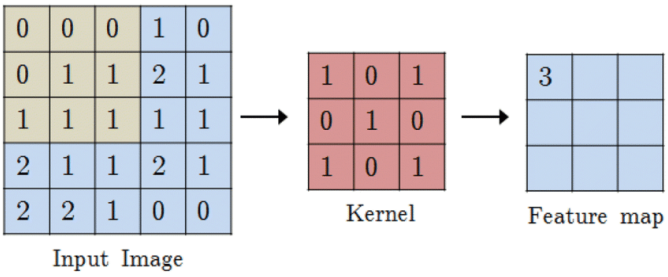


Figura 12. Convolució en forma de kernel. Font: https://www.researchgate.net/figure/Graphical-representation-of-convolution-operation-the-center-pixel-of-kernel-is-placed_fig3_340255272

Per l'operació contrària que passes d'una imatge petita a una imatge més gran passant-la per un *kernel,* s'ha de redefinir una mica com es pot dur a terme la convolució. El *kernel* es pot transformar en una matriu de convolució i la imatge d'entrada s'ha de transformar en un vector per poder-ho fer utilitzant la multiplicació de matrius. La matriu de convolució s'obté reorganitzant el *kernel* en una matriu i aplicant *zero padding*[10] en diferents punts. Aquesta operació segueix donant el mateix resultat simplement s'han reorganitzat les dades. Per poder fer el que seria l'*upscaling*[11] simplement s'ha de transposar la matriu de convolució que s'ha obtingut i passar-li la imatge petita en forma de vector.

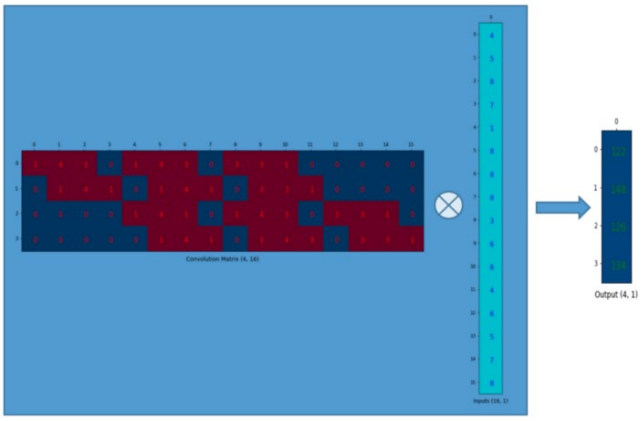


Figura 13. Convolució en forma de matriu. Font: https://naokishibuya.medium.com/up-sampling-with-transposed-convolution-9ae4f2df52d0

[10] Afegir zeros
[11] Escalar la imatge, augmentat les dimensions

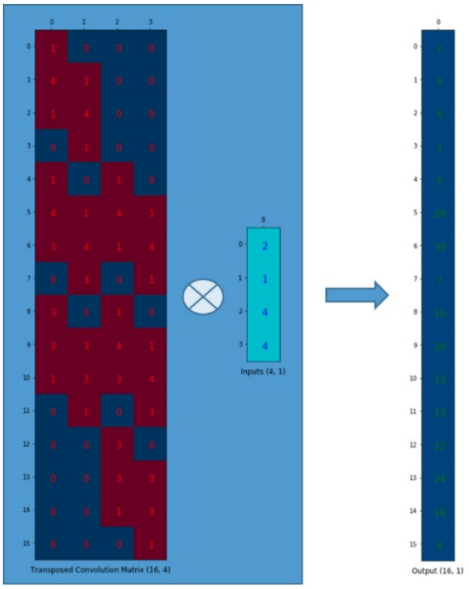


Figura 14. Convolució en forma de matriu transposada. Font: https://naokishibuya.medium.com/up-sampling-with-transposed-convolution-9ae4f2df52d0

1.6.6 Què és el *pooling*?

El *pooling* és una operació similar en la convolució, ja que, té el mateix objectiu, reduir o concentrar la informació. Per fer-ho es crea una finestra (com el *kernel* en la convolució) i es mira el màxim valor d'aquesta finestra per fer un *max pooling* o el valor mitjà per fer un *average pooling*. [6]

1.6.7 Què és la funció *softmax*?

La funció *softmax* permet que donat un vector de valors reals amb tot tipus de valors tant, negatius, positius o majors que 1 es transformi en un vector en què tots estiguin entre 0 i 1. Aquesta part ja la podria fer una funció sigmoide. La diferència és que la funció *softmax* també fa que el vector resultant sumi 1 en total.

Ara que s'han vist els blocs bàsics per construir les arquitectures que s'han utilitzat per a aquest treball, descriurem les xarxes bàsiques emprades per realitzar aquest treball. A l'apartat d'implementació s'explicarà com aquestes arquitectures han estat usades per obtenir els resultats que s'obtenen.

1.6.8 Què és una CNN?

Una xarxa neuronal convolucional és un tipus de *Deep Neural Network* que utilitza un conjunt de capes, les capes convolucionals, *pooling layers* i finalment un *fully-onnected layer*. L'objectiu és que la convolució concentri la informació de la imatge en, per exemple contorns o formes. A més capes de convolució més es concentra aquesta informació, fent que en les

últimes capes certs objectes de les imatges puguin activar només certes neurones. Aquestes neurones que s'hauran activat, per tant, poden acabar diferenciant objectes els uns amb els altres [7].

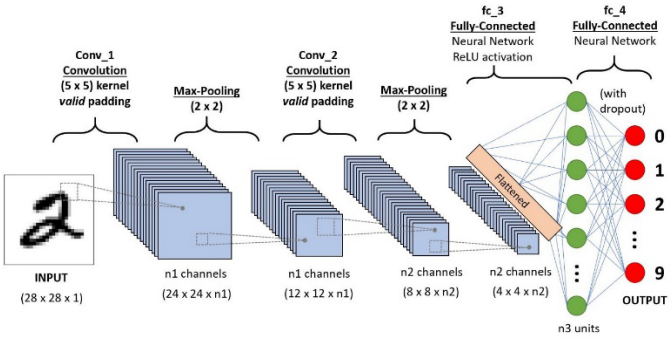


Figura 15. Arquitectura CNN[12]*. Font: https://www.analyticsvidhya.com/blog/2020/10/what-is-the-convolutional-neural-network-architecture/*

2 Estudi de viabilitat

En un projecte d'intel·ligència artificial una part molt important és la col·lecta de dades, ja que sense aquestes no es pot desenvolupar un projecte de forma satisfactòria. Per poder fer aquesta part, per tant, es necessita temps principalment per trobar diverses fonts i comprovar que sigui el que es vol o es necessita.

Un cop es té la part de la intel·ligència artificial ben preparada i entrenada s'ha de buscar com es pot importar a un motor de jocs per tal de realitzar un videojoc. Per poder dur a terme correctament la recerca en intel·ligència artificial i el videojoc posterior es necessitaran molts recursos entre personal, *software*, *hardware* i altres eines que poden ajudar en el desenvolupament, les quals es concreten en aquest apartat.

2.1 Software

A continuació s'exposen els programes que s'han utilitzat per dur a terme el desenvolupament del projecte.

[12] Representa l'exemple del MINST Dataset(https://deepai.org/dataset/mnist)

2.1.1 Unity

Figura 16. Logo de Unity

Unity és un dels motors de jocs més coneguts i ofereix una llicència lliure per empreses que facturen menys de 100.000€ anuals. Amb aquest motor es pot desenvolupar la *demo* del projecte per tal de mostrar com és el seu funcionament.

2.1.2 Visual Studio Code

Figura 17. Logo Visal Studio Code

Editor de codi bastant lleuger i multiplataforma, ha estat útil per fer petites modificacions i en resum poder analitzar el codi en cada sistema operatiu.

2.1.3 PyCharm

Figura 18. Logo PyCharm

Editor de codi molt usat en projectes de *python*, té moltes funcionalitats i és multiplataforma. S'ha usat directament en *Ubuntu* per tal de no haver-lo d'instal·lar dos cops. L'editor conté moltes eines que han sigut d'utilitat per tal de poder analitzar el codi.

2.1.4 Jupyter lab

Figura 19. Logo Jupyter Lab

Jupyter lab és un programa que permet la computació interactiva i ha sigut clau per fer algunes proves de forma ràpida i és *open source*. Ha estat molt útil al llarg del treball per poder executar codi i visualitzar els resultats directament en el mateix lloc.

2.1.5 Word

Figura 20. Logo Word

L'editor de text més conegut ha sigut clau per tal de redactar aquesta memòria. Les seves funcionalitats per gestionar la bibliografia i tot tipus de referències a més de ser el que més es coneixia.

2.1.5.1 Cost del software

Nom	Cost
Unity	0€ en llicència personal[13]
Visual Studio Code	0€ i *open source*[14]
PyCharm	0€ en llicència d'estudiant[15]
Jupyter Lab	0€ i *open source*[16]
Word	0€ en llicència d'estudiant[17]

Taula 1. Costos de software

[13] Llicència Unity: https://unity3d.com/unity/activation/personal
[14] Llicència Visual Studio Code: https://code.visualstudio.com/license
[15] Llicència PyCharm: https://www.jetbrains.com/community/education/#students
[16] Llicència JupyterLab: https://github.com/karmanandan/JupyterLab/blob/main/LICENSE
[17] Llicència Word: https://www.microsoft.com/es-es/education/products/office

2.2 Recursos

2.2.1 Hadware

Per al hardware es té una torre de les següents característiques:

- **Processador**: Intel Core i9-10900KF @ 3.7Ghz x 20
- **Targeta gràfica**: NVIDIA GeForce RTX 3080Ti
- **RAM**: 32GB
- **Disc dur**: 500GB SSD, 500GB HDD i 1TB HDD. 5TB HDD extern

Mentre es tenia una xarxa entrenant-se a la torre s'ha aprofitat el portàtil per preparar la memòria o la *demo*. El portàtil té les següents característiques:

- **Processador**: Intel Core i7-9750H @ 2.6Ghz x 12
- **Targeta gràfica**: NVIDIA GeForce RTX 2060
- **RAM**: 16GB
- **Disc dur**: 1TB SSD
-

2.2.1.1 Cost del hardware

Nom	Cost
Ordinador portàtil	1300€
Ordinador sobretaula	2500€
Total	3800€, cost assumit per aquest projecte 0€

Taula 2. Costos de hardware

El sistema operatiu utilitzat per les xarxes neuronals és *Ubuntu* mentre per la *demo* en *Unity* i la redacció de la memòria s'ha utilitzat Windows 10.

2.2.2 Humans

Per la part de recursos humans s'han tingut en compte les següents consideracions, primerament que la durada del projecte seria de 3 mesos i una setmana, donant un total de 720 hores. Pel que fa al sou per tal de poder estimar un cost monetari, s'ha estimat que seria d'uns 13,5€/h, equivalent a tres vegades el sou mínim dels estudiants en estades a l'entorn laboral de l'EPS[18]. Segons aquests criteris el resultat que s'obté és el següent:

[18] A data d'avui, 10/4/2022

Tasca	Hores	Cost (€)
Recerca de tècniques	30	405
Investigació viabilitat	25	337,5
Aprenentatge llibreries	35	472,5
Implementació de les tècniques	45	607,5
Testeig i correcció de les tècniques	160	2160
Implementació del joc	25	337,5
Implementació del sistema de joc a distancia (tipus *Stadia*)	60	810
Documentació del projecte	95	1282,5
Total	475	6412,5

Taula 3. Estimació recursos humans

Segons la taula, per tant, els costos humans s'eleven a un total de 6412 €. Aquest projecte s'ha fet sense ànim de lucre i, per tant, aquest cost és una estimació. Les estimacions quadren amb un temps equivalent a un treball a jornada completa de manera que és viable tant per la part econòmica com per la part temporal.

2.3 Viabilitat

2.3.1 Estudi de mercat

Abans de desenvolupar el treball s'han d'investigar quines tecnologies i quins jocs en els que podrien aplicar les xarxes neuronals que s'estan desenvolupant. Així, la primera part, es centrarà en mirar quines tecnologies es troben al mercat i la segona en mirar quins jocs podrien ser un bon objectiu per poder aplicar-los la restauració de color mitjançant les esmentades tecnologies.

Stadia i *GeForceNow* són plataformes de *cloud gaming* que possibiliten que el joc estigui dins del servidor i, el jugador només rep imatges i envia els inputs al servidor per tal que els pugui processar.

2.3.1.1 Stadia

Stadia [1] aprofita el navegador Chrome (creat i mantingut per l'empresa mare, *Google*) per tal de poder utilitzar el seu sistema de *gaming*. Recomanen una connexió de 10 Mbps i, en cas de

voler utilitzar en resolució a 4K recomanen 35 Mbps. L'objectiu és que no compris jocs i facis servir la seva subscripció que conté molts jocs gratuïts per 9,9€ al mes[19].

Figura 21. Logo Stadia

2.3.1.2 GeForceNow

Nvidia GeForce Now [2] demana que et descarreguis el seu software, pels requisits de connexió són similars als de *Stadia*, 15 Mbps per 720p, 25 Mbps per 1080p i 35 Mbps per 4K. Especificant que permetria uns 60 *fps*[20], també recomanen tenir una latència inferior a 80 ms, recomanant que sigui menor a 40 ms.

Figura 22. Logo Nvidia GeForce Now

2.3.1.3 Forza Horizon 5

Aquest és el joc que s'ha agafat com a referència per tal de desenvolupar la part de restauració de color. És un joc de carreres en món obert on es poden veure una gran varietat d'escenaris i conté un ampli ventall de vehicles.

L'objectiu ha estat cercar un joc que tingués una varietat d'escenari, colors i objectes per tal de tenir un repte per la intel·ligència artificial. Tot i tenir una bona combinació d'aquests tres factors, es creu que aportarà bons resultats perquè tot i tenir un escenari canviant, les zones tenen certa similitud entre un *frame* i el següent.

[19] A data de 19/4/2022
[20] Frames per second, fotogrames per segon.

Figura 23. Logo Forza Horizon 5

2.3.2 Públic objectiu i perfil de jugador

Aquesta recerca està dissenyada per tal d'utilitzar-se en sistemes com *Stadia* i, per tant, qualsevol jugador disposat i amb els requisits que es necessiten per tenir aquests sistemes està dins del públic objectiu. La tipologia més gran dels jugadors que empren aquestes tecnologies són jugadors que no volen que es facin trampes, per tant, es podria dir que són jugadors als que els agraden els jocs competitius. També té l'avantatge que els sistemes multijugador competitius podrien anar més suau si el joc està corrent en un mateix servidor de manera que l'únic problema de latència que hi pogués haver seria del servidor a cadascun dels jugadors.

2.3.3 Estat de l'art

En aquest apartat veurem quines investigacions s'han dut a terme en aquest àmbit, com s'aplicaran i la importància que tindran al llarg del treball. Principalment, es parlarà de dues tècniques, la de *Bringing Old Photos Back To Life* [3] i *DeOldify* [8].

El primer és un *paper* en què s'explica com dur a terme la restauració d'imatges antigues pel que fa al desgast i danys que es produeixen amb el pas del temps. Els problemes que tracta són els danys estructurals i els no estructurals. Per tant, cobreix molta part del que es vol tractar durant aquest treball.

El segon és una eina que serveix per donar color a les imatges en blanc i negre. D'aquesta manera, es podrien passar les imatges que s'hagin restaurat, però que no tenien color per tal d'acabar de portar-les a l'actualitat. Per tal de solucionar aquesta part, utilitzant aquesta eina, també ens servirà per poder desenvolupar un joc de *demo* per exemplificar el seu ús pràctic en un sistema com el *Stadia*.

Per tant, el que s'estudia en aquest treball es podria aprofitar per a aquests sistemes tipus *Stadia* per tal de reduir la quantitat d'amplada de banda necessària per a fer funcionar correctament el sistema. Aquesta part en resum té com a objectiu reduir la quantitat de dades a enviar, però això, almenys de forma directa no hauria d'afectar a la latència.

2.3.4 Obtenció de dades

Com s'ha comentat anteriorment, les dades són el cor dels projectes d'intel·ligència artificial, per tant, una part important a tenir en compte és si es poden obtenir dades. Per aquest treball principalment es necessitaven quatre tipus d'imatges:

- Imatges antigues, amb desgast
- Imatges actuals i/o imatges restaurades
- Imatges en blanc i negre
- Imatges a color

Per la part de restauració es necessiten imatges antigues amb desgast i, seria ideal tenir la versió restaurada. Tenir la imatge restaurada és un procés molt costós i, per tant, es tindria un set molt petit o seria molt car d'obtenir, per tant, per tal de poder restaurar-les s'haurà de passar per imatges generades. Al tractar amb imatges amb soroll (danys no estructurals) es pot simular bastant fàcilment el tipus de dany que es vol, incorrent en el potencial problema que les imatges obtingudes no representin correctament el dany real. La segona part del treball que consisteix en la coloració d'imatges. És molt més fàcil obtenir dades, ja que es poden agafar les imatges a color i passar-les a blanc i negre. Per tant, s'han hagut de buscar data sets d'imatges actuals a color i imatges antigues amb danys.

3 Desenvolupament

3.1 Requisits del sistema

Per poder executar els diferents projectes es tenen les següents limitacions:

- Sistema operatiu: *Ubuntu*
- 12 GB RAM a la GPU
- 32 GB RAM al PC

L'entrenament és molt costós, raó d'aquestes limitacions. En el primer *paper* aquestes són lleugerament menors per dur a terme l'entrenament, el que sobretot requereix molta capacitat computacional és el *deoldify* per intentar dur a terme una tasca molt més complexa.

3.2 Metodologia de treball

Per tal de desenvolupar aquest projecte s'ha utilitzat la metodologia *Scrum* [9] i s'han desenvolupat *sprints*. La metodologia de *Scrum* permet treballar en petites parts i obtenir feedback de la feina feta de manera que si s'ha de canviar o descartar alguna cosa, es fa en poques setmanes i no es perd molt de temps. Els *sprints* és la representació de la metodologia, es dissenyen un set de tasques i, com a *deadline*[21] s'estableix el pròxim *sprint*. Quan s'arriba a la data d'entrega es fa una retrospectiva i es defineixen noves tasques.

L'objectiu és que el tutor estigui al corrent de tot el que es desenvolupava i quin és l'estat del projecte en tot moment. Per comunicar-li l'estat s'ha utilitzat el correu electrònic on s'ha especificat l'estat del projecte respecte a l'última reunió. D'aquesta manera es podia aprofitar els avenços per saber si s'havia de crear un nou *sprint*. Amb les reunions que s'anaven realitzant, s'anaven perfilant un seguit de tasques generals que s'havien de fer en les pròximes setmanes. Després de la reunió s'estructuraven les tasques per poder aprofitar al màxim el temps i els recursos disponibles. Els *sprints*, no estaven programats en antelació sinó que a mesura que s'anaven completant les tasques i començaven a sorgir dubtes o problemes es feia la reunió llavors, per tant, els *sprints* no eren d'una durada fixa com serien de forma ideal.

[21] Data límit

4 Estudis i decisions

Per tal de dur a terme aquest TFG, s'han pres un conjunt de decisions per tal d'emmarcar el treball en un àmbit concret. Les decisions que s'han hagut de prendre inclouen els papers en què s'hagi basat principalment aquest treball com les tecnologies per tal de desenvolupar-lo.

Primerament, els estudis que han servit com a base per realitzar el treball és el de *Bringing Old Photos Back To Life* i el *deOldify*. Aquests dos contenien la informació necessària per poder desenvolupar el projecte.

Al tenir una base sobre la qual treballar, les llibreries utilitzades depenien principalment de les que s'empraven en aquests dos estudis. Els dos es basen en *python* i utilitzen la targeta gràfica per a dur a terme els entrenaments. El primer es basa principalment en *pytorch* mentre que el segon utilitza *pytorch*, *fastai* i *jupyterlab*.

5 Disseny del sistema i implementació

En aquest apartat s'explicarà el marc de treball i quins paràmetres es poden modificar de les xarxes per tal de dur a terme les proves posteriorment.

5.1 Marc de treball

5.1.1 Que és VGG?

VGG [10] és una xarxa CNN especialitzada per dur a terme una tasca de classificació. Aquesta xarxa té com a objectiu obtenir un vector de característiques(*features*) d'una imatge per tal de poder saber que conté.

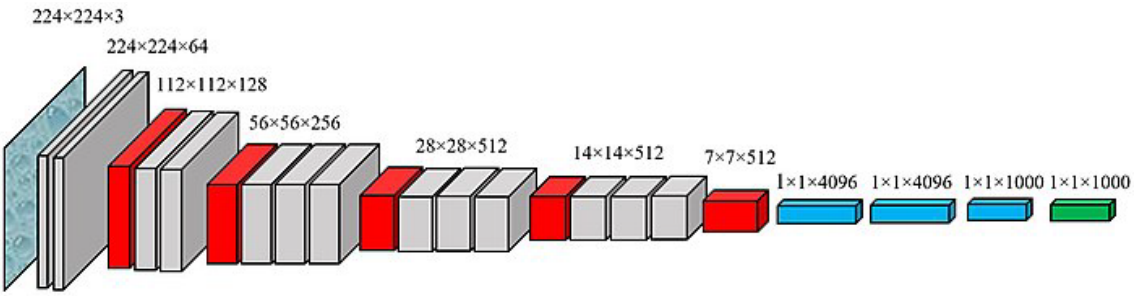


Figura 24. Estructura d' una VGG(VGG-16). Font: https://en.everybodywiki.com/VGG_Net

Les capes grises representen capes de convolució i *ReLU*, les capes vermelles representen l'operació de *max pooling*, les blaves són *fully-connected layers* i finalment al final trobem un *softmax*. Aquesta combinació de capes permet concentrar la informació d'una imatge en un vector de característiques que ens descriu el contingut de la imatge.

5.1.2 Què és *ResNet*? Què és un *ResBlock*?

ResBlock [11] és un bloc que permet reduir la difusió de gradient que, simplement és que a força de dur a terme les operacions de multiplicació es va perdent precisió. Per reduir la reducció de precisió es defineix l'output com l'input més l'input passat per la funció, com un tipus de normalització.

La *ResNet* correspon a l'arquitectura completa, on a part de les convolucions s'han inclòs els *ResBlock* per transmetre informació a través de diverses capes per tal d'evitar la pèrdua d'informació. En la imatge es veu representat el model *plain*[22] i el residual, el residual és com el *plain,* però conté el *ResBlock* i es transmet la informació a través de les fletxes en negreta. Les que estan en línia discontinua representen que s'ha de tractar un increment de dimensió. [12]

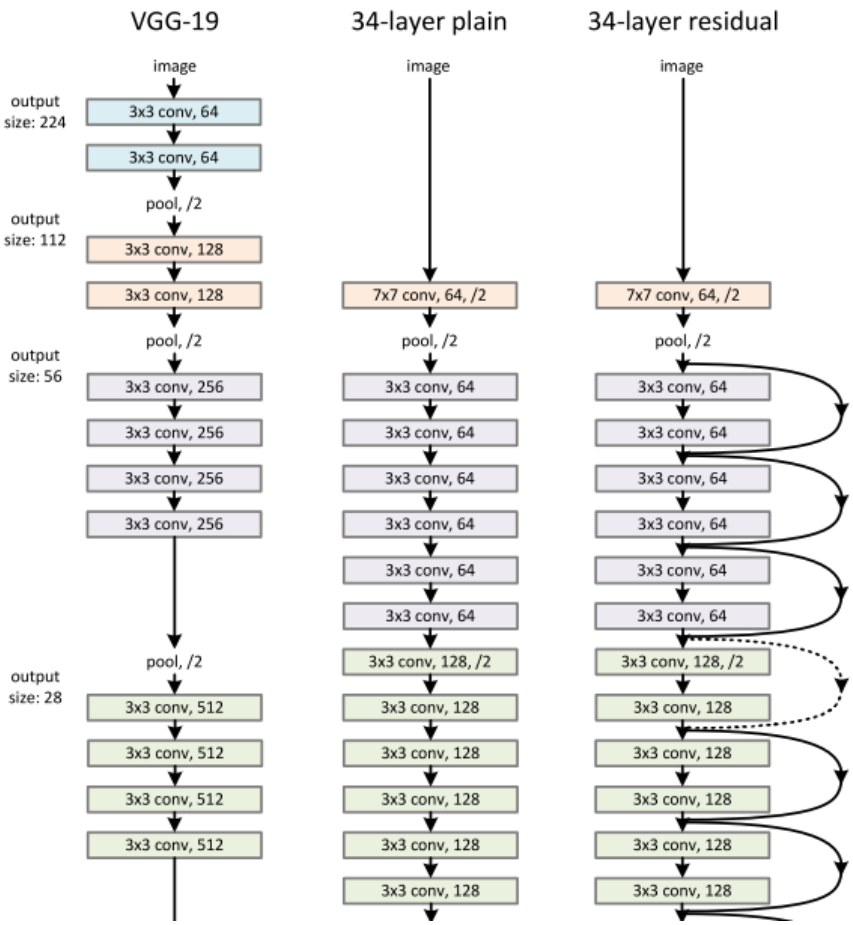


Figura 25. Comparativa d'arquitectures VGG, plain i residual. Font: https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035

[22] Pla, fent referència a que la xarxa és completament lineal sense salts.

5.1.3 Latent space

És un espai on s'ha extret les característiques més significatives de la imatge segons la xarxa neuronal. En algunes arquitectures on es troba un codificador i després un descodificador, com les *VAEs*, el *latent space* seria *l'output* del codificador i *l'input* del descodificador. Normalment, és un vector que conté una sèrie de valors, cada combinació de valors significaria una imatge resultant diferent.

5.1.4 Partial nonlocal block

Part utilitzada per omplir els forats en *latent space*. Utilitza la informació adjacent per a omplir la informació mancant. L'objectiu és aprofitar la informació del voltant. En aquest cas, s'utilitza la informació al voltant de una zona on falta informació, com un forat a una foto per generar informació que concordi amb el seu entorn. [3]

5.1.5 Backpropagation

Backpropagation [13] és el terme usat per la propagació endarrere dels errors. S'utilitza en els mètodes d'entrenaments supervisats, ja que, amb una funció de pèrdues que utilitza l'objectiu al que es vol arribar i el resultat que s'ha obtingut per tal d'obtenir un gradient d'aquesta funció de pèrdues. Amb aquest gradient podem actualitzar cada pes de la xarxa neuronal anant de l'última capa fins a la primera.

5.1.6 Què es una GAN?

General adversarial network. L'estructura d'una *GAN* es basa en un generador i un discriminador. El generador crea imatges a base d'unes dades comprimides i el discriminador ha de diferenciar les imatges generades de les imatges reals. S'entén que una *GAN* està entrenada quan el discriminador ja no és capaç de diferenciar les imatges generades de les reals. Segons unes instàncies X i unes etiquetes Y, el generador busca obtenir la probabilitat conjunta p(X,Y) o p(X) si no hi ha etiquetes, mentre que el discriminador modela la probabilitat condicional p(X|Y). [14]

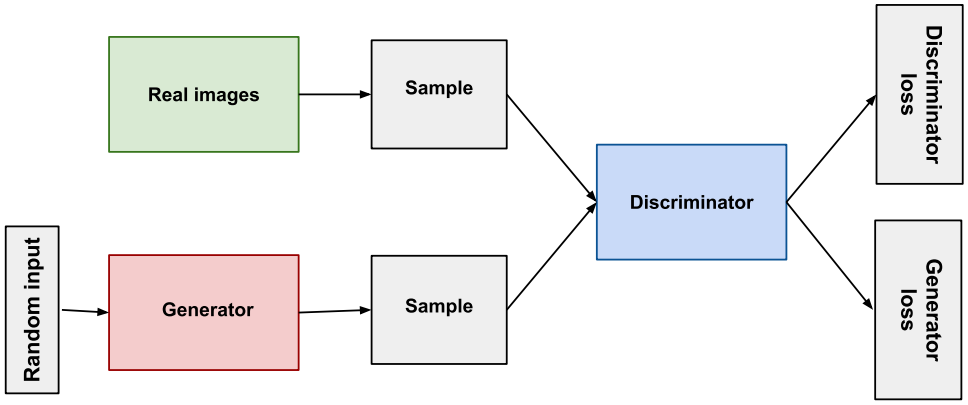


Figura 26. Estructura GAN. Font: https://developers.google.com/machine-learning/gan/gan_structure

5.1.6.1 CGAN

És un tipus de *GAN* concret en què es té més dades *d'input*, això permet que un generador que normalment generaria solament basant-se en les dades codificades puguem afectar a aquesta generació indicant-li el que volem. Aquesta informació també es pot passar al discriminador. El cas més clar és si es té un generador de nombres, que, en canvi, de generar nombres aleatoris generi el nombre que nosaltres volem. Les xarxes *CGAN* modelen la probabilitat condicional P(X|Y) en canvi. de la probabilitat conjunta.

5.1.7 Què és una U-net?

La *Unet* és una arquitectura que està dissenyada per tal de solucionar el problema de la *semantic segmentation*[23] en el mínim de dades d'entrenament possible. Es basa en un sistema de codificació i descodificació. La part de codificació redueix per la meitat la dimensió espacial i dobla en nombre de filtres, el descodificador, d'altra banda, fa l'operació contrària.

[23] Segmentació de la imatge, fent que cada píxel de la imatge pertanyi a una classe en concret

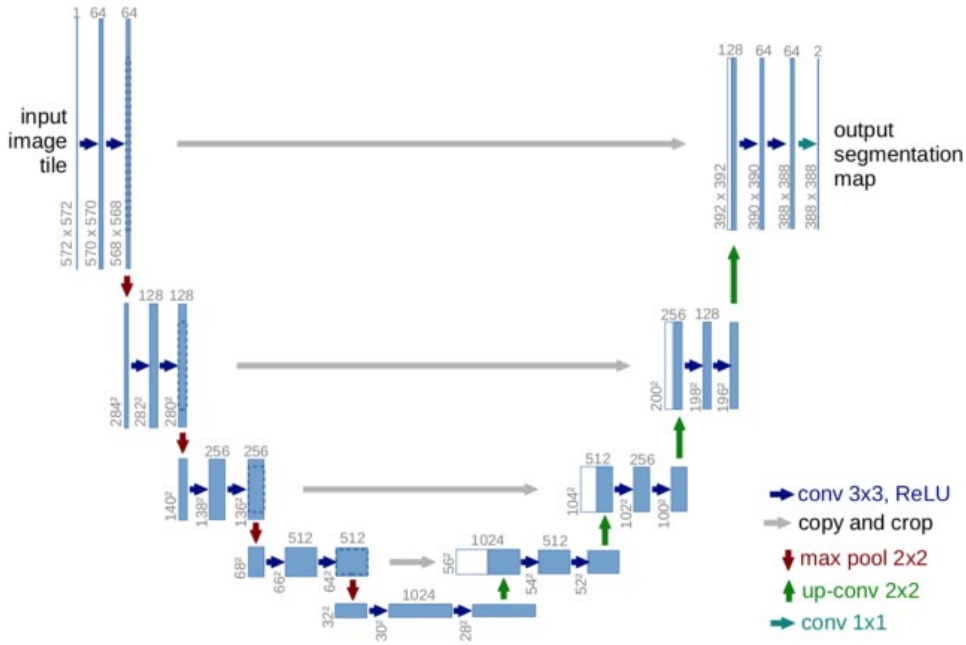


Figura 27. Arquitectura U-Net. Font: https://towardsdatascience.com/unet-line-by-line-explanation-9b191c76baf5

La codificació es duu a terme utilitzant convolucions i un *ReLU* en combinació a un *max pooling*. L'objectiu és que al final del codificador es tingui una descripció de què conté la imatge.

Aquesta xarxa a part de tenir un codificador i descodificar inclou dos termes que l'ajuden a complir la seva tasca. La primera són les anomenades *skip connections* que són utilitzades per donar-li al generador més informació per tal de generar la imatge, d'aquesta manera també es pot millorar l'aprenentatge amb *backpropagation*, per tal de fer-ho es passen les dades des del codificador al descodificador. L'altre terme és el *bridge* que és la part que connecta el codificador i el descodificador, també està format de convolucions i *ReLU*.

La descodificació es realitza fent la convolució transposada, s'ajunta amb la *skip connection* adient, es fan unes convolucions seguides d'una *ReLU* i es torna a repetir el procés. D'aquesta manera s'obté la imatge segmentada que es passa per una convolució i per la funció d'activació de sigmoide.

5.1.8 Pix2Pix

Pix2Pix [15] és una xarxa basada en una *CGAN* on el generador és, més concretament un *Unet* modificada. Aquesta xarxa té com a objectiu aprendre a fer el mapeig entre l'input i l'output.

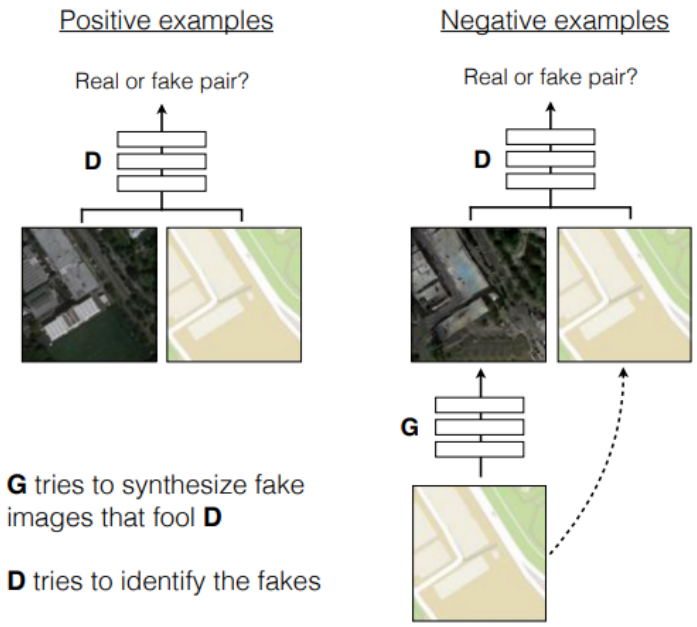


Figura 28. cGAN de Pix2Pix. Font: https://yusuke-ujitoko.hatenablog.com/entry/2017/06/25/165008

L'estructura del generador és així:

- Cada bloc del codificador: Convolució -> Normalització per lots -> *Leaky ReLU*[24]
- Cada bloc del descodificador: Convolució transposada -> Normalització per lots -> Abandonament (als 3 primers blocs) -> ReLU
- Hi ha connexions entre el codificador i el descodificador, com és el cas de la Unet.

5.1.9 SAGAN

Les *SAGAN* [16] són *self attention GAN*. És un tipus de *GAN* que conté *layers* de *self-attention*. Aquestes capes busquen relacionar els *inputs* entre ells i, *l'output* que s'obté és un vector de la mateixa dimensió que els *inputs* que conté els valors que representen aquesta relació. La idea és que cada *input* sàpiga quina atenció o importància hauria de donar a cadascun dels altres *inputs*.

[24] ReLU però en petita inclinació per valors negatius, en canvi de valor 0. https://paperswithcode.com/method/leaky-relu

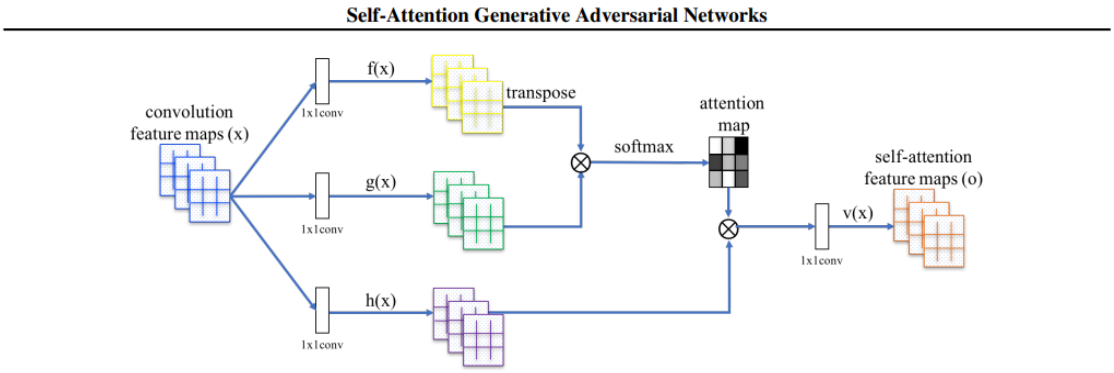


Figura 29. Estructura capa de self-attention. Font: https://arxiv.org/pdf/1805.08318.pdf

Aquesta estructura permet tenir en compte la informació propera per tal de tenir més informació per modificar *l'input* en concret. Els sistemes de traducció utilitzen aquesta arquitectura per tal de donar més importància a certes paraules de la oració més que d'altres. En imatges, si es té un forat per exemple, es pot utilitzar la informació del voltant per omplir el forat.

5.1.10 VAE

Variational Autoencoder, es tracta d'una estructura de xarxa neuronal basada en un codificador i un descodificador. El codificador fa una mena d'anàlisi de components principals per tal d'obtenir les dades més importants, en el nostre cas d'una imatge. El descodificador ha de ser capaç de tornar a generar la imatge donada les dades codificades.

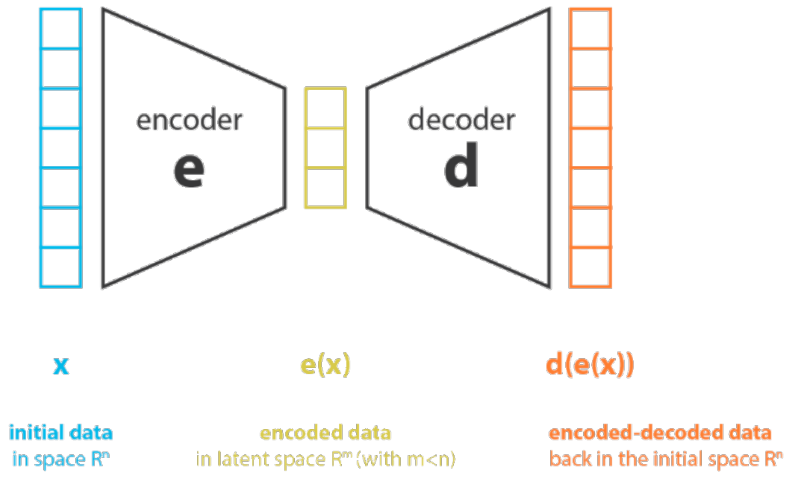


Figura 30. Estructura VAE. Font: https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

5.2 Bringing Back Old Photos Back To Life

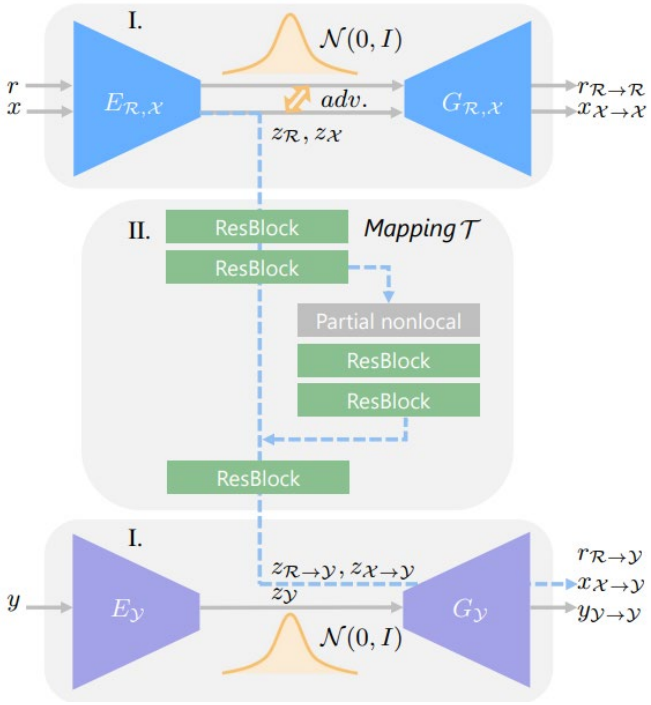


Figura 31. Arquitectura de la xarxa de restauració d'imatges. Font: https://github.com/microsoft/Bringing-Old-Photos-Back-to-Life

Es basa en dos *VAEs* on la primera treballa en imatges codifica i descodifica les imatges antigues i les imatges sintètiques, de manera que quan es té codificat s'assemblin molt les sintètiques i les antigues reals. La segona *VAE* realitza el procés de codificació i descodificació de les imatges actuals. Amb aquestes dos *VAEs* es pot muntar el sistema final, que es tracta d'un sistema de traducció entre les dades codificades de les imatges antigues i les dades codificades de les imatges reals, veure Figura 31. Amb aquesta traducció l'execució que es durà a terme per tal d'obtenir una imatge restaurada seria, es codifica la imatge antiga utilitzant la primera *VAE*, es tradueix per a la segona *VAE* i aquesta segona, ha de generar la imatge restaurada a partir de la informació codificada.

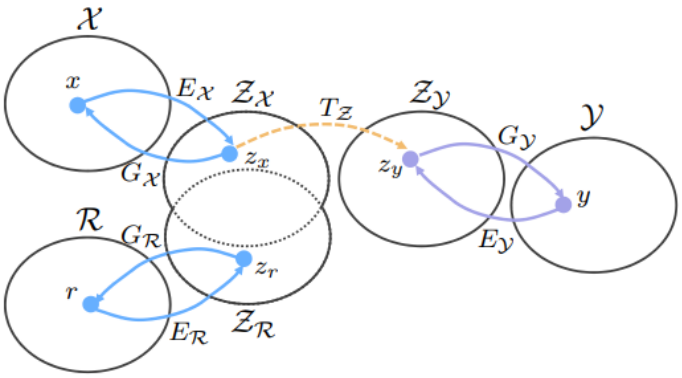


Figura 32. Representació de els traduccions. Font: https://github.com/microsoft/Bringing-Old-Photos-Back-to-Life

En resum, el que s'està realitzant és una traducció en diferents dominis. X i R serien les imatges sintètiques i reals, quan es troben codificades l'objectiu és que comparteixen una part del domini. Aquesta part és important, ja que per passar d'aquest domini codificat al domini real codificat es necessita poder dur a terme el *mapping* entre les imatges sintètiques en la seva versió actual sense danys. Quan s'entrena la part de traducció codificada només en imatges sintètiques si el domini d'imatges sintètiques no es superposa bé amb el de les imatges reals no es podrà dur a terme un bon entrenament. La part final seria utilitzar el generador de les imatges reals per a generar la imatge final amb la informació que es té codificada.

5.3 Deoldify

Aquest codi empra la xarxa U-net per al generador i, usant el mecanisme de self-attention mencionat anteriorment. A més, utilitza un mètode d'entrenament anomenat *No-GAN* que permet facilitar l'entrenament, realitzant l'entrenament del generador i del discriminador per separat d'una manera més directa. Per tal de solucionar els diferents problemes que es poden tenir al voler donar color a una imatge en blanc i negre, utilitzant arquitectures lleugerament diferents s'obtenen tres models, l'artístic, l'estable i el de vídeo.

5.3.1 Model artístic

Aquest model ens dona imatges més colorades, però no té prou robustesa per tasques més normals com pintar una escena natural o en els retrats. Per tal de dur a terme l'extracció de característiques s'utilitza una *ResNet34*.

Figura 33. Resultat de l'execució de Deoldify amb el model artístic

5.3.2 Model estable

El model estable, en comparació al model artístic va millor per escenes naturals o els retrats. Per tant, sol donar imatges pintades menys estranyes que l'artístic, però això també fa que siguin menys acolorides. El canvi que s'ha realitzat per què funcioni millor amb aquestes escenes ha sigut canviar la xarxa per la *Resnet101*.

Figura 34. Resultat de l'execució de Deoldify amb el model estable

Veure la Figura 33 i Figura 34 per veure la diferència de color entre el model artístic i el model estable.

5.3.3 Model de vídeo

Aquest, en realitat és una versió del model estable en un entrenament diferent per tal que sigui més robust i consistent. Com a contrapartida és el menys colorit de tots.

5.4 Paràmetres

Les xarxes tenen molts paràmetres que es poden modificar perquè el resultat de l'entrenament sigui més satisfactori. En aquesta part s'explicaran els paràmetres que s'han modificat per dur a terme les proves.

5.4.1 Nombre de dades per entrenar

Tot i no ser un paràmetre intrínsec, és un factor a tenir en compte en entrenar una xarxa neuronal. Afecta el temps d'entrenament i en el resultat però, arriba a un punt que per millorar la xarxa es necessiten moltes dades per millorar-la lleugerament. S'ha d'anar amb compte en les dades que s'utilitzen per entrenar, si s'utilitzen dades massa similars, es pot donar un cas d'*overfitting*, en què la xarxa treballa molt bé en aquestes dades però, no es capaç d'interpretar noves dades.

5.4.2 Dimensions de les imatges a tractar i batch size

Al moment d'entrenar la xarxa, s'escala o es retalla la imatge per tal de poder-la entrenar per limitacions tècniques tant de memòria com de capacitat computacional. Aquest és un factor que també s'ha de tenir en compte, ja que, pot permetre que es puguin posar més imatges en la memòria, com més petites siguin les imatges en més imatges podrem treballar a la vegada. Per tant, es pot buscar un balanç entre el nombre d'imatges que s'entrenen a la vegada (*batch size*) i les dimensions d'aquestes. Quan es treballen en imatges, per pintar-les, per exemple, per tal que doni bons resultats hauria de ser superior a 4, passant-li menys donaria resultats molt dolents.

5.4.3 CGAN

La xarxa té un paràmetre que permet activar si la *GAN* que s'utilitza és condicional o no. Per tant, també pot aportar canvis, més concretament pot canviar el comportament del discriminador al passar-li les dades d'*input*.

```
if self.opt.no_cgan:
    # Fake Detection and Loss
    pred_fake_pool = self.discriminate(None, fake_image, use_pool=True)
    loss_D_fake = self.criterionGAN(pred_fake_pool, False)

    # Real Detection and Loss
    pred_real = self.discriminate(None, real_image)
    loss_D_real = self.criterionGAN(pred_real, True)

    # GAN Loss (Fake Passability Loss)
    pred_fake = self.netD.forward(fake_image)
    loss_G_GAN = self.criterionGAN(pred_fake, True)
else:
    # Fake Detection and Loss
    pred_fake_pool = self.discriminate(input_label, fake_image, use_pool=True)
    loss_D_fake = self.criterionGAN(pred_fake_pool, False)

    # Real Detection and Loss
    pred_real = self.discriminate(input_label, real_image)
    loss_D_real = self.criterionGAN(pred_real, True)

    # GAN Loss (Fake Passability Loss)
    pred_fake = self.netD.forward(torch.cat((input_label, fake_image), dim=1))
    loss_G_GAN = self.criterionGAN(pred_fake, True)
```

Figura 35. Ús de CGAN a l'entrenament

5.5 Realització del joc

Per tal de dur a terme la *demo* del joc s'ha emprat el cotxe dels *Unity assets* i s'ha posat dins d'un circuit de carreres. La combinació d'aquests *assets* ha permès centrar els esforços a mirar com s'ha d'incloure la xarxa neuronal a *Unity* i com s'utilitza.

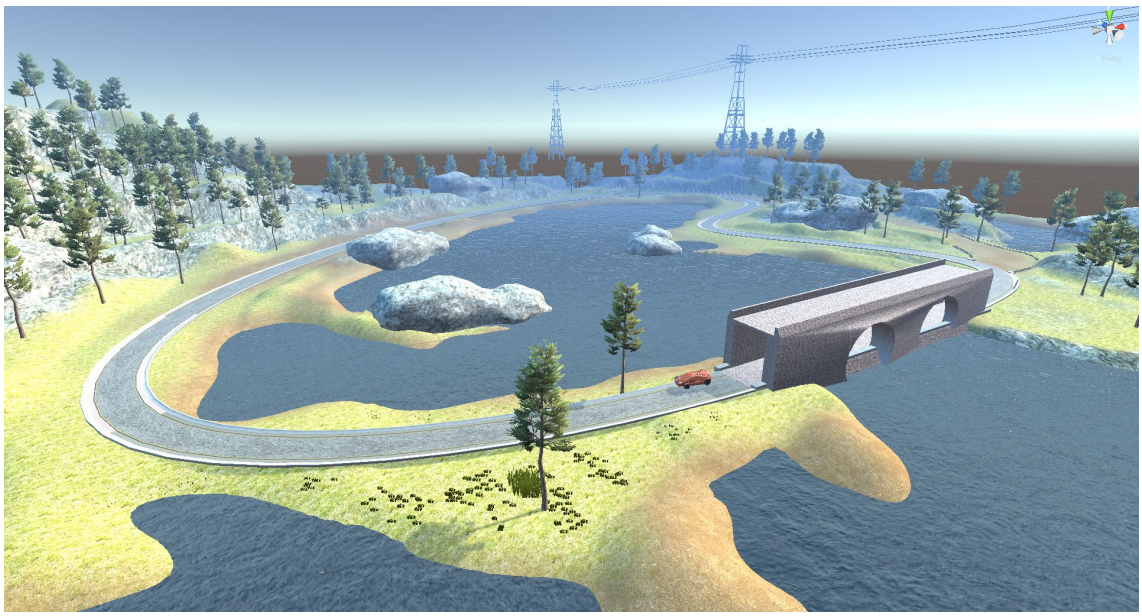


Figura 36. Escenari de la demo.

La xarxa que es vol utilitzar és la de *deOldify* per provar com seria utilitzar una xarxa en un motor de jocs. Per poder veure els resultats primer s'ha de poder passar la imatge en blanc i negre. Per fer-ho s'ha creat un *shader*:

```
fixed4 frag (v2f i) : SV_Target
{
    fixed4 col = tex2D(_MainTex, i.uv);
    fixed bw = ( col.r + col.g + col.b ) / 3;
    return fixed4(bw,bw,bw, col.a);
}
```

Figura 37. Fragment shader per passar imatge a blanc i negre

Simplement per cada *frame*, es passa cadascun dels píxels pel *fragment shader*. Aquest, agafa els valors de roig verd i blau i fa la mitjana, es posa el valor mitjà a cadascun dels valors d'*output*.

Per tal d'aplicar un efecte a través d'una xarxa neuronal s'ha d'instal·lar el *plugin*[25] de barracuda[26] a *Unity*. A més s'ha d'exportar la xarxa en el format *ONNX*[27] que és un format que conté tota la informació de la xarxa i permet treballar en diferents programes.

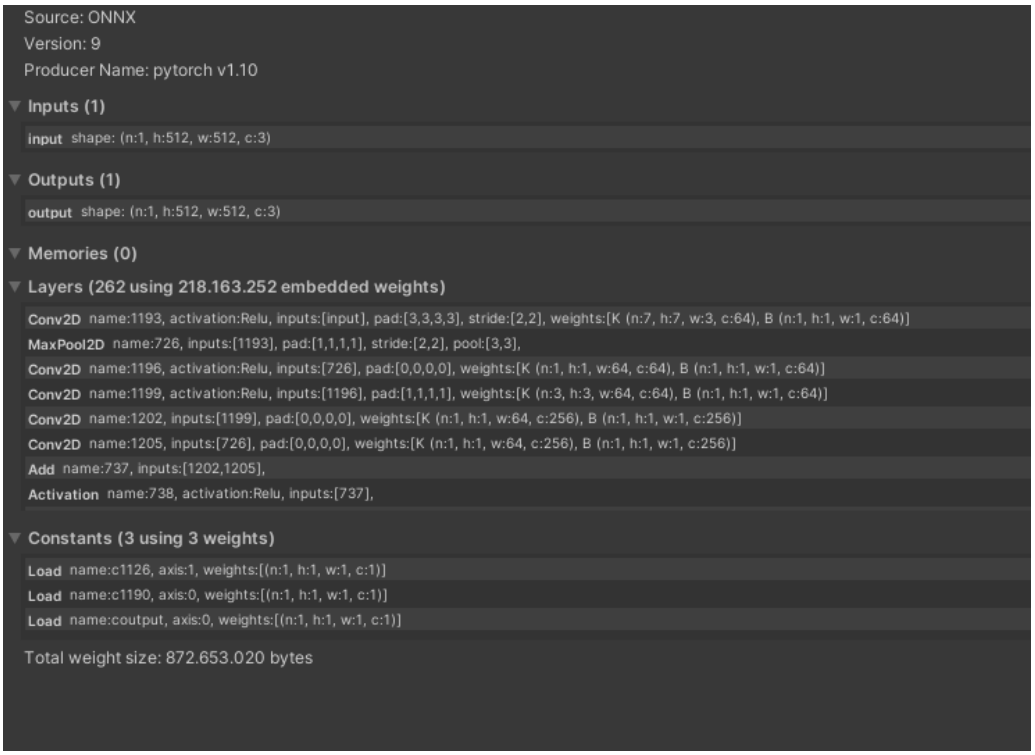


Figura 38. Estructura de la xarxa importada a Unity

[25] Mòdul que proporciona un servei concret a un programa més gran
[26] https://docs.unity3d.com/Packages/com.unity.barracuda@3.0/manual/index.html
[27] https://onnx.ai/

6 Proves i resultats

Per tal d'avaluar els diferents resultats, es realitzaran diferents proves i diferents formes de mesurar els resultats segons la xarxa que s'estigui analitzant.

6.1 Dades usades i tractament de dades

Per tal de poder entrenar les diferents xarxes es necessitaven grans conjunts de dades. Les imatges s'han extret de:

- *DIV2K* (800 fotos en color)
- *PASCAL_VOC* (17126 fotos en color, s'han usat 4200 per realitzar el treball)
- *Geelong historical images*: Fotos antigues de la ciutat de Geelong d'australia (87 fotos antigues tant en lleugers tons de color com en blanc i negre)
- *Tweede wereldoorlog nationaal archief* : Arxius nacionals de la Segona Guerra Mundial de Netherlands. (7860 fotos antigues en blanc i negre, s'han usat 3913 fotos),a partir d'ara abreviat a 2WW.
- *Brave New World* (pel·lícula 1h 59 min de vídeo antic, extretes 7048 fotos a 5 *frames* per segon. Filtrades 3998 fotos per utilitzar), a partir d'ara abreviat a BNW.
- *Remix* de clips de *Pierrot Le Fou*, *Weekend* i Le *Grand Amour* (15 min de vídeo antic, extretes 4645 fotos. Filtrades 2264 fotos per utilitzar). A partir d'ara abreviat a PLF.

També s'han necessitat vídeos per tal d'entrenar la part de coloració, aquest s'han extret de:

- Gravant seqüències del videojoc *Forza Horizon 5*
    - Seqüencia en 1 cotxe i 1 escenari: 12053 fotos extretes d'una gravació de 24 minuts a 10 *frames* per segon
    - Seqüència en 3 cotxes i 3 escenaris: 15662 fotos extretes d'una gravació de 26 segons a 10 *frames* per segon
- Gravant el videojoc de *demo*.

Per tal d'entrenar la part de danys no estructurals, s'han necessitat fer unes modificacions als danys que es generaven durant l'entrenament de la xarxa. S'han hagut de modificar, ja que semblaven molt poc realistes i molt exagerats.

```python
def online_add_degradation_v2(img):
    img = convertToJpeg(img,random.randint(40,100))
    converter = PIL.ImageEnhance.Color(img)
    img = converter.enhance(0.8)
    img = blur_image_v2(img)
    #img = synthesize_gaussian(img, 1, 10)
    img = synthesize_low_resolution(img)
    img = convertToJpeg(img,random.randint(40,100))
    img = synthesize_speckle(img, 1, 10)
    img = blur_image_v2(img)
    img = synthesize_gaussian(img, 1, 10)
    img = synthesize_low_resolution(img)
    img = convertToJpeg(img,random.randint(40,100))
    img = synthesize_speckle(img, 1, 10)
    converter = PIL.ImageEnhance.Color(img)
    img = synthesize_gaussian(img, 1, 10)
    img = synthesize_low_resolution(img)
    img = convertToJpeg(img,random.randint(40,100))
    img = synthesize_speckle(img, 1, 10)
    converter = PIL.ImageEnhance.Color(img)
    img = converter.enhance(0.6)
    img = convertToJpeg(img,random.randint(40,100))
```

Figura 39. Síntesi de soroll modificada

Figura 40. Diferencia en imatges generades en soroll i la original abans de la modificació de la funció de soroll

El primer canvi important és que s'ha llevat el funcionament *random* de manera que controlem completament la generació i, ens donarà imatges que no tindran tantes diferències entre aquestes imatges que, posteriorment ajudarà a l'entrenament no tenir tanta variabilitat i ens donarà millors resultats. La manera d'aplicar el soroll ara, a part de ser completament controlada s'ha afegit la reducció de la saturació que, es considera una part important, ja que, les càmeres antigues siguin les fotos o els vídeos tenen menys profunditat de color i, no tenien la capacitat de processar el color correctament.

Figura 41. Comparació imatge amb soroll i imatge original després de la modificació de la funció de soroll

6.2 Bringing Old Back Photos Back to Life

La manera d'avaluar els resultats d'aquesta xarxa serà a través de mesures quantitatives. Aquestes mesures es basen en posar-hi un nombre al resultat obtingut, analitzant la quantitat de soroll que conté la imatge, per exemple. S'han utilitzat les mateixes mesures que utilitzen en el paper per a tenir una comparativa no només amb els resultats obtinguts en les diferents proves, sinó que també es pugui comparar amb els resultats del paper.

6.2.1 Mesures quantitatives emprades

6.2.1.1 FID: Fréchnet inception distance [17]

Aquesta mesura té com a objectiu ser una millora a *l'inception score*[28]. La idea base de l'*inception score* és passar les dades per la xarxa Inception-v3 per tal d'obtenir un conjunt d'etiquetes. Com a resultat, idealment, quan es passin imatges amb objectes concrets, s'obtindrà una distribució en poca entropia, mentre que quan es passin un conjunt d'imatges diverses, s'obtindrà alta entropia. Posteriorment, es mesura la *KL-divergence* per tal d'avaluar les dues distribucions. El *FID*, per tal d'intentar millorar-ho el que realitza és comparar les estadístiques de les imatges generades amb les estadístiques de les imatges reals per tal de

[28] https://medium.com/octavian-ai/a-simple-explanation-of-the-inception-score-372dff6a8c7a

poder-ho comparar i no avaluar les estadístiques de forma aïllada. Aquesta mesura és millor com menor sigui el valor que s'obtingui.

6.2.1.2 PSNR: Peak Signal-To-Noise Ratio [18]

PSNR, representa una ràtio entre el màxim valor del senyal, en aquest cas la lluminositat del píxel, i el soroll que afecta la representació fidel de la imatge. Per tal de saber si una imatge és d'alta qualitat segons aquesta mètrica, es compara el resultat de *PNSR* entre la imatge sense soroll en la imatge en soroll.

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

Figura 42. Funció del PNSR

6.2.1.3 SSIM: structural similarity index measure [19]

El *SSIM*, extreu 3 característiques d'una imatge, luminància, contrast i estructura. Aquestes característiques es comparen entre les dues imatges i els resultats es combinen per obtenir la mesura de similitud. La comparació d'aquests factors té com a objectiu indicar quan de similars són les dues imatges. Com més alt és el valor més similars són i, el seu valor es troba entre 0 i 1.

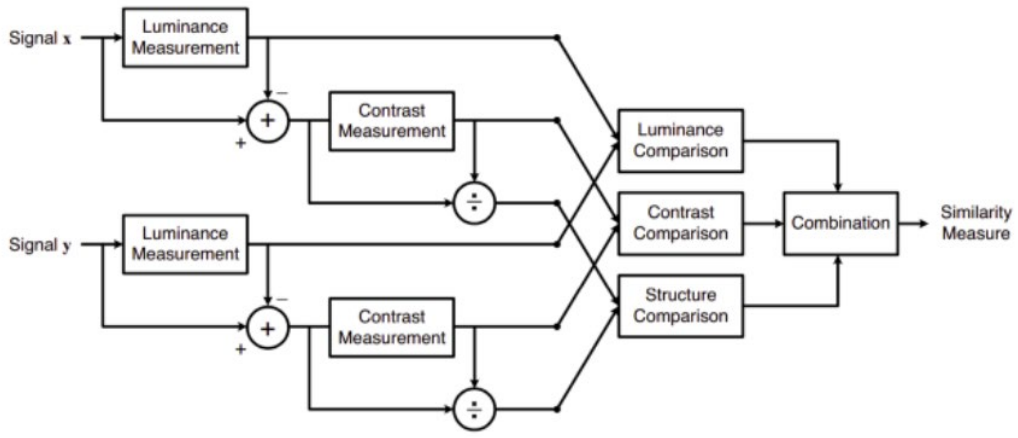


Figura 43. Estructura de còmput de SSIM. Font: https://www.cns.nyu.edu/pub/eero/wang03-reprint.pdf

6.2.1.4 LPIPS: Learned Perceptual Image Patch Similarity [20]

El *LPIPS*, intenta ser una mesura de similitud que es basi en xarxes neuronals, de manera que pugui tenir en compte els matisos que tenim els humans al comparar les imatges i que les mesures com *PNSR* que, al ser un còmput no és capaç de tenir en compte un conjunt de factors.

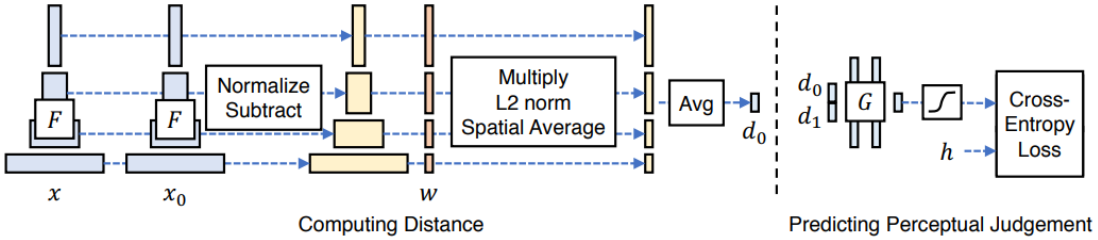


Figura 44. Esquema pel còmput de la distància amb LPIPS. Font: https://richzhang.github.io/PerceptualSimilarity/

6.2.2 Proves fetes

Per tal d'analitzar els resultats d'aquesta xarxa, s'han realitzat dos tipus de comprovacions. La primera consisteix a provar-ho en diverses quantitats d'imatges, és a dir, augmentant o disminuint les dades d'entrenament com a tal, així com revisar quin efecte té desactivar la *CGAN*. Aquests tipus de proves s'han realitzat en sets de 500, 1000, 2000 i 4000 imatges.

La segona part de les proves s'ha realitzat només amb una quantitat d'imatges fixada. Per tal d'avaluar els resultats de les diverses proves s'han creat un conjunt de test que consisteix en 1000 imatges que passaran per la xarxa entrenada i, amb les imatges generades s'obtindrà el valor de les mesures quantitatives corresponents. El conjunt de test seran imatges que cap dels entrenaments haurà vist anteriorment.

Nº dades	Blanc i negre
	Imatges de la 2WW
500	500
1000	1000
2000	2000
4000	4000

Taula 4. Quantitat d'imatges en blanc i negre segons l'entrenament i les diferents bases de dades que es tenen

Nº dades	Antigues a color	
	frames de *BNW*	*frames* de PLF
500	386	114
1000	719	281
2000	1293	707
4000	2490	1510

Taula 5. Quantitat d'imatges antigues a color segons l'entrenament i les diferents bases de dades que es tenen

Nº dades	Noves a color	
	DIV2K	*PASCAL VOC*
500	46	454
1000	58	942
2000	113	1887
4000	574	3426

Taula 6. Quantitat d'imatges noves a color segons l'entrenament i les diferents bases de dades que es tenen

6.2.2.1 Entrenament amb les comandes recomanades del paper

Per començar a realitzar les proves s'ha volgut agafar com a referència les comandes que recomanen a la pàgina del paper, modificant les parts que impedien l'execució per limitacions de memòria. Les característiques principals d'aquesta prova són:

Els temps d'entrenaments són:

Nº dades \ Pas	Domini A	Domini B	*Mapping*	Total
500	1,6 h	0,5 h	3,8 h	5,9 h
1000	2,7 h	1 h	6,5 h	10,2 h
2000	5,3 h	2 h	12,5 h	19,8 h
4000	15,8 h	4 h	36,1 h	55,9 h

Taula 7. Temps d'entrenament Bringing Back Old Photos Back To Life amb les comandes recomanades

Els resultats obtinguts són:

Nº dades \ Mesura	FID	SSIM	LPIPS	PNSR
500	73,41	0,7994	0,2355	24,27
1000	75,51	0,7998	0,25	23,72
2000	74,98	0,8037	0,2493	24,11
4000	72,25	0,8059	0,2550	24,42

Taula 8. Resultats obtinguts Bringing Back Old Photos Back To Life amb les comandes recomanades

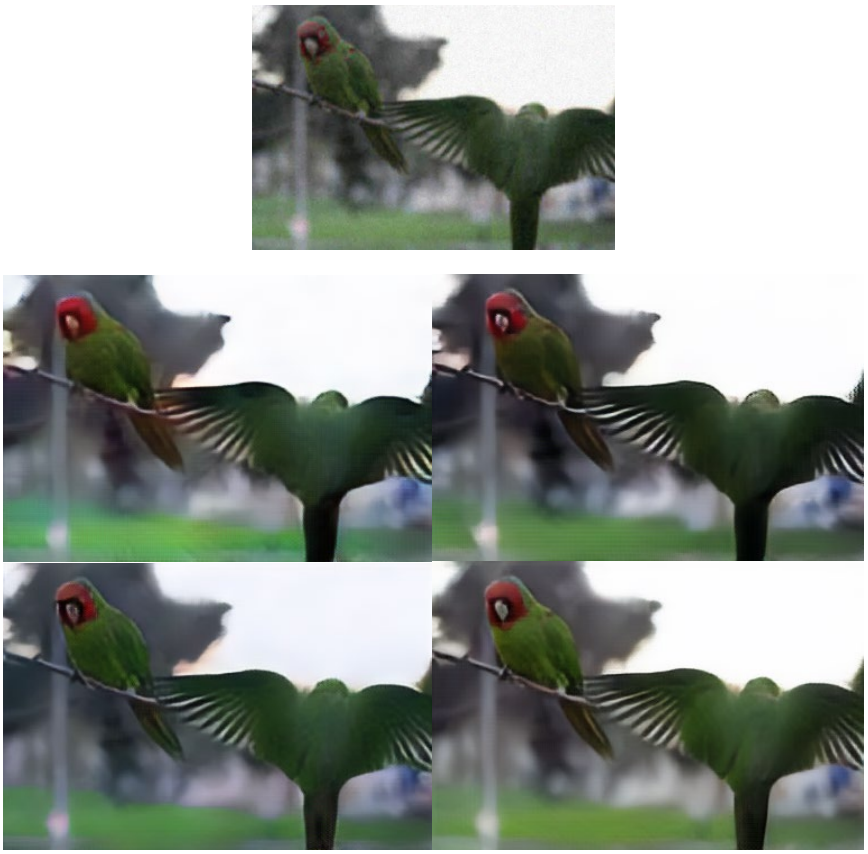


Figura 45. Resultats amb les comandes recomanades Bringing Old Photos Back To Life. a) Imatge d'entrada b) resultat en 500 imatges d'entrenament c) resultat en 1000 imatges d'entrenament d) resultat en 2000 imatges d'entrenament e) resultat en 4000 imatges d'entrenament

Com es pot veure en la Figura 45, les comandes recomanades donen bons resultats fins i tot en poques dades, la zona en que es veu més diferència és la zona de la cara de l'ocell. També hi ha canvis de lluminositat sobre els dos últims, d i e.

6.2.2.2 Activant desactivant el CGAN

Després, per tal de fer un canvi a la versió vista anteriorment, s'ha tornat a realitzar les mateixes execucions desactivant la *CGAN*. Això fa que el discriminador de la xarxa ja no tingui l'etiqueta de la imatge d'entrada i, per tant el discriminador pot tenir més dificultats per tal de diferenciar imatges reals de falses.

Els temps d'entrenaments són:

Nº dades \ Pas	Domini A	Domini B	*Mapping*	Total
500	1,2 h	0,5 h	4 h	5,7 h
1000	3,2 h	1 h	6,6 h	10,8 h
2000	4,7 h	2,1 h	12,1 h	18,9 h
4000	14,4 h	4,3 h	36,1 h	54,8 h

Taula 9. Temps d'entrenament Bringing Back Old Photos Back To Life descativant la CGAN

Els resultats obtinguts són:

Nº dades \ Mesura	*FID*	*SSIM*	*LPIPS*	*PNSR*
500	65,27	0,7035	0,2085	24,84
1000	68,65	0,6352	0,2448	24,07
2000	72,16	0,5767	0,256	23,39
4000	73,25	0,6872	0,258	23,95

Taula 10. Resultats obtinguts Bringing Back Old Photos Back To Life desactivant la CGAN

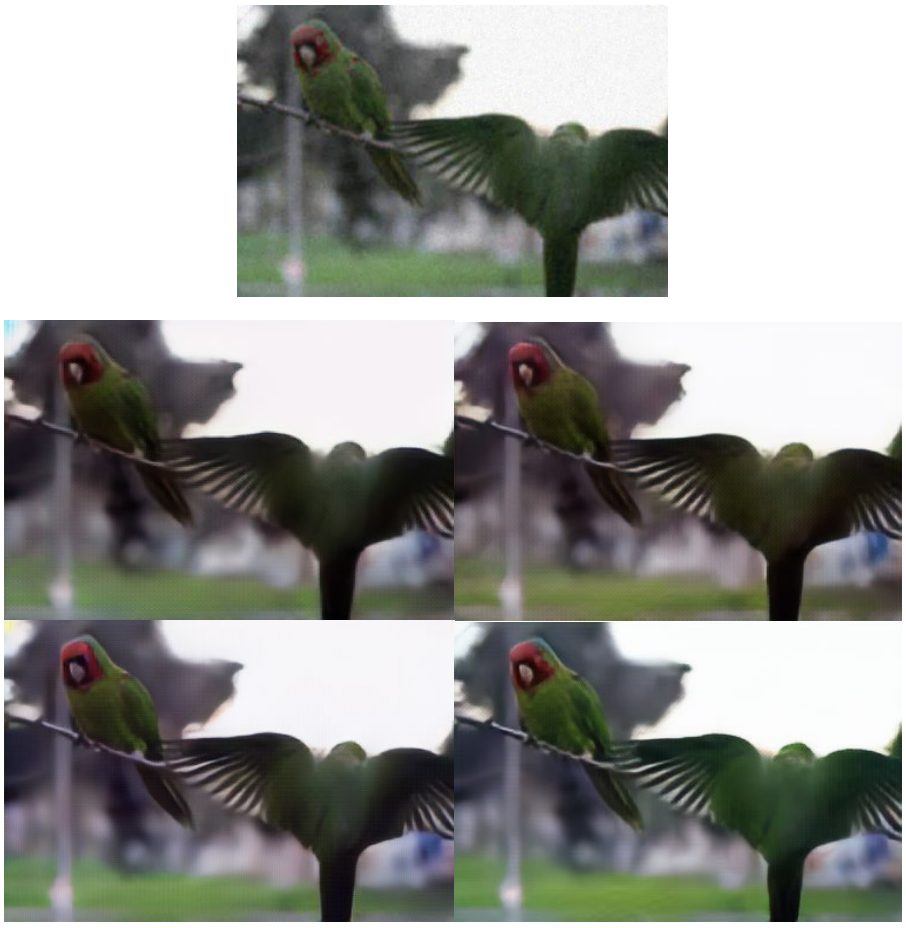


Figura 46. Resultats desactivant la CGAN Bringing Old Photos Back To Life. a) Imatge d'entrada b) resultat en 500 imatges d'entrenament c) resultat en 1000 imatges d'entrenament d) resultat en 2000 imatges d'entrenament e) resultat en 4000 imatges d'entre

Comparant els resultats visualment, mirant la Figura 45 i la Figura 46, no es veu gran diferencia si es comparen les imatges amb mateixa quantitat de dades d'entrenament. D'altra banda, si es revisa de forma quantitativa, tenim que els resultats de la Taula 8 amb els de la Taula 10, es veu que clarament que desactivar al *conditional GAN* afecta a l'entrenament,

empitjorant el resultat en totes les mesures quan s'utilitzen moltes dades i, en el cas d'entrenar en menys dades, el FID surt pitjor per el cas base.

6.2.2.3 Variant el batch size i les dimensions de les imatges

L'última prova només s'ha realitzat amb un set de 1000 imatges, provant les següents combinacions:

Nom representatiu	Dimensions de les imatges	*Batch size*
32	32x32px	128
64	64x64px	64
128	128x128px	32

Taula 11. Característiques de l'entrenament Bringing Back Old Photos Back To Life amb variació de batch size i dimensió d'imatges

Els temps d'entrenaments són:

size \ Pas	Domini A	Domini B	*Mapping*	Total
32	2,5 h	0,8 h	5,6 h	8,9 h
64	2,3 h	0,2 h	5,5 h	8 h
128	2,9 h	0,5 h	5,9 h	9,3 h

Taula 12. Temps d'entrenament Bringing Back Old Photos Back To Life amb diferent batch size i diferents dimensions de les imatges

Els resultats obtinguts són:

Nº dades \ Mesura	*FID*	*SSIM*	*LPIPS*	*PNSR*
32	107,3127	0,487	0,3175	17,03
64	77,72	0,5784	0,3091	21,51
128	68,59	0,6886	0,2304	24,45

Taula 13. Resultats obtinguts Bringing Back Old Photos Back To Life amb diferent batch size i diferents dimensions de les imatges

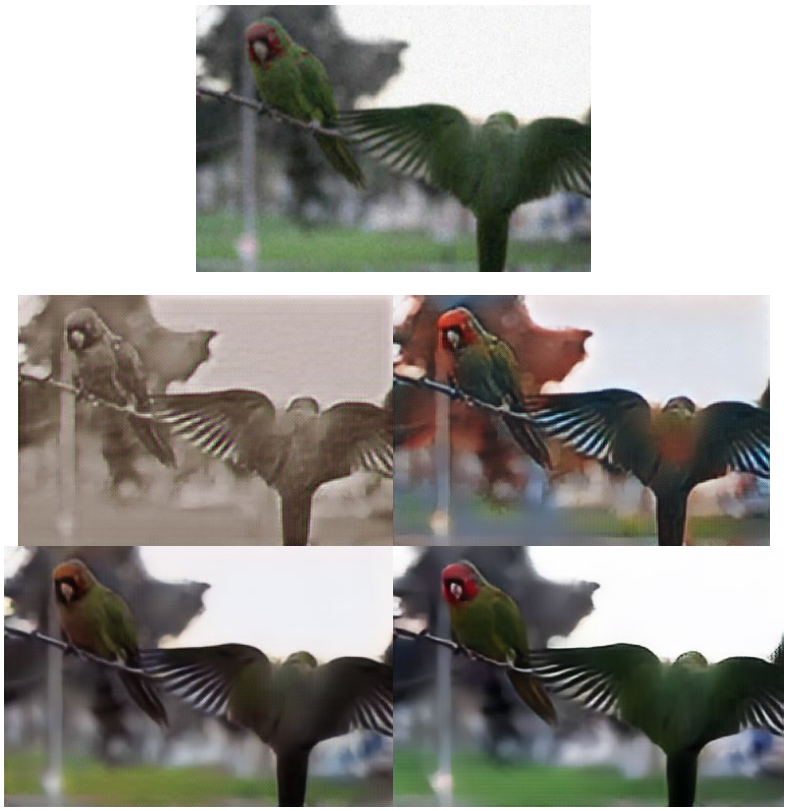


Figura 47. Resultats amb diferents batch size i dimensions de les imatges Bringing Old Photos Back To Life. a) Imatge d'entrada b) resultat en imatges de 32x32 píxels c) resultat en imatges de 64x64 píxels d) resultat en imatges de 128x128 píxels d'entrenament e) resultat en imatges de 256x256 píxels

Veient els resultats obtinguts a la Figura 47, es veu que els resultats són bastant dolents en les dimensions més petites de les imatges. Al comparar-ho quantitativament, agafant el millor experiment, el cas de 128 píxels, de la Taula 13 amb la fila de 1000 de la Taula 8 es veu que els resultats són bastant satisfactoris, concretament les mesures *FID, LPIPS i PNSR* són molt millors amb 128 píxels que qualsevol quantitat de dades amb 64 píxels.

6.3 DeOldify

Per tal de realitzar les proves en *deoldify*, primer s'ha mirat quin conjunt de dades utilitzaven per a dur a terme l'entrenament. Al veure que s'utilitzava *ImageNet*[29] que és un data set de 166 GB, s'ha decidit emprar el *DIV2K* que s'havia utilitzat anteriorment per tal de veure quins resultats s'obtenien. Després al veure que no es pintava de forma correcta les imatges s'ha decidit acotar el problema a un joc en concret per tal de tenir un conjunt d'elements concrets per tal que la xarxa els pugui identificar i pintar de forma distingida.

[29] https://www.kaggle.com/competitions/imagenet-object-localization-challenge/data

6.3.1 Proves en fotos

Els primers tests s'han realitzat en les imatges per entendre el seu funcionament. Per tal de dur a terme el primer entrenament, s'ha utilitzat el data set de *DIV2K*. Els resultats obtinguts són molt dolents, això és degut al fet que la quantitat de dades que es tenien de cada tipus eren molt limitats. Això explica per què les zones més identificables de cada imatge si que tenen una mica de color, el cas més correcte és com ha pintat el cel però, tot i això, arriba a detectar el gos tot i no poder-lo pintar correctament. *DeOldify*, té un paràmetre de *render factor* que s'aplica per pintar les imatges un cop la xarxa està entrenada. Això, permet obtenir resultats més colorits o menys, per tal de tenir una comparativa per al pintat d'imatges el valor de *render factor* s'ha establert en 25 per les imatges.

Figura 48. Resultat obtingut Deoldify amb Stable Colorizer entrenat amb el data set DIV2K

Per tal d'obtenir millors resultats, s'ha decidit acotar millor el problema. Per fer-ho s'ha agafat el joc de *Forza*. S'han gravat vídeos a dins del joc i després s'han passat el *frames* perquè els pinti com si fossin imatges qualsevols. La primera prova s'ha gravat un vídeo de 24 minuts a 10 *frames* on surti només un cotxe i un dels escenaris del joc (obtenint 12053 fotos després de netejar fotos de menús). Amb aquest nou punt de vista, s'han pogut obtenir molt millors resultats:

Figura 49. Resultat obtingut Deoldify amb Stable Colorizer entrenat amb imatges del Forza. A) Cotxe que ha vist durant l'entrenament b) Segon cotxe i, en escenari molt diferent c) Tercer cotxe en un altre tipus d'escenari

Amb aquest tipus d'entrenament es produeix un alt nivell *d'overfitting* ja que, com es pot veure, no és capaç de pintar correctament el cel. Els cotxes els pinta tots de color blau ja que és la única informació de color que té i, per tant, aquesta part seria correcta segons el que s'ha entrenat però, el cel principalment en la tercera imatge falta més detall de color en algunes zones. Per el segon entrenament s'ha decidit gravar un altre fragment de aproximadament la

mateixa durada (26 minuts a 10 *frames* per segon, donant 15662 fotos filtrant menús) però amb diferents vehicles i diferents escenaris per veure com es comporta al pintar les imatges.

Figura 50. Resultat obtingut Deoldify amb Stable Colorizer entrenat amb imatges del Forza en diversos cotxes i escenaris. A) Primer cotxe, en la selva b) Segon cotxe a la muntanya c) Tercer cotxe en escenari tipus ciutat

Arriba a pintar de forma molt més adequada els diferents vehicles i els entorns que ha vist durant l'entrenament tenen més color i són més correctes. Fins i tot, ha sigut capaç de pintar els números dels *combos* que es poden fer dins del joc.

6.3.2 Proves en vídeos

El processament de vídeos és l'apartat més interessant, ja que la xarxa treballa en *frames* individuals i, per tant, podria tenir problemes a pintar *frames* seguits, ja que podria haver-hi incoherència temporal. L'entrenament d'aquesta part de la xarxa conté una part per reduir el soroll entre *frames* que es pot produir en els vídeos. Per fer el processament dels vídeos, el *render factor* s'ha canviat a 30.

La primera prova ha sigut amb els vídeos del joc de *Forza* per tal de veure si el procediment també funciona amb vídeos. Aquest ha sigut el resultat obtingut d'aquesta prova:

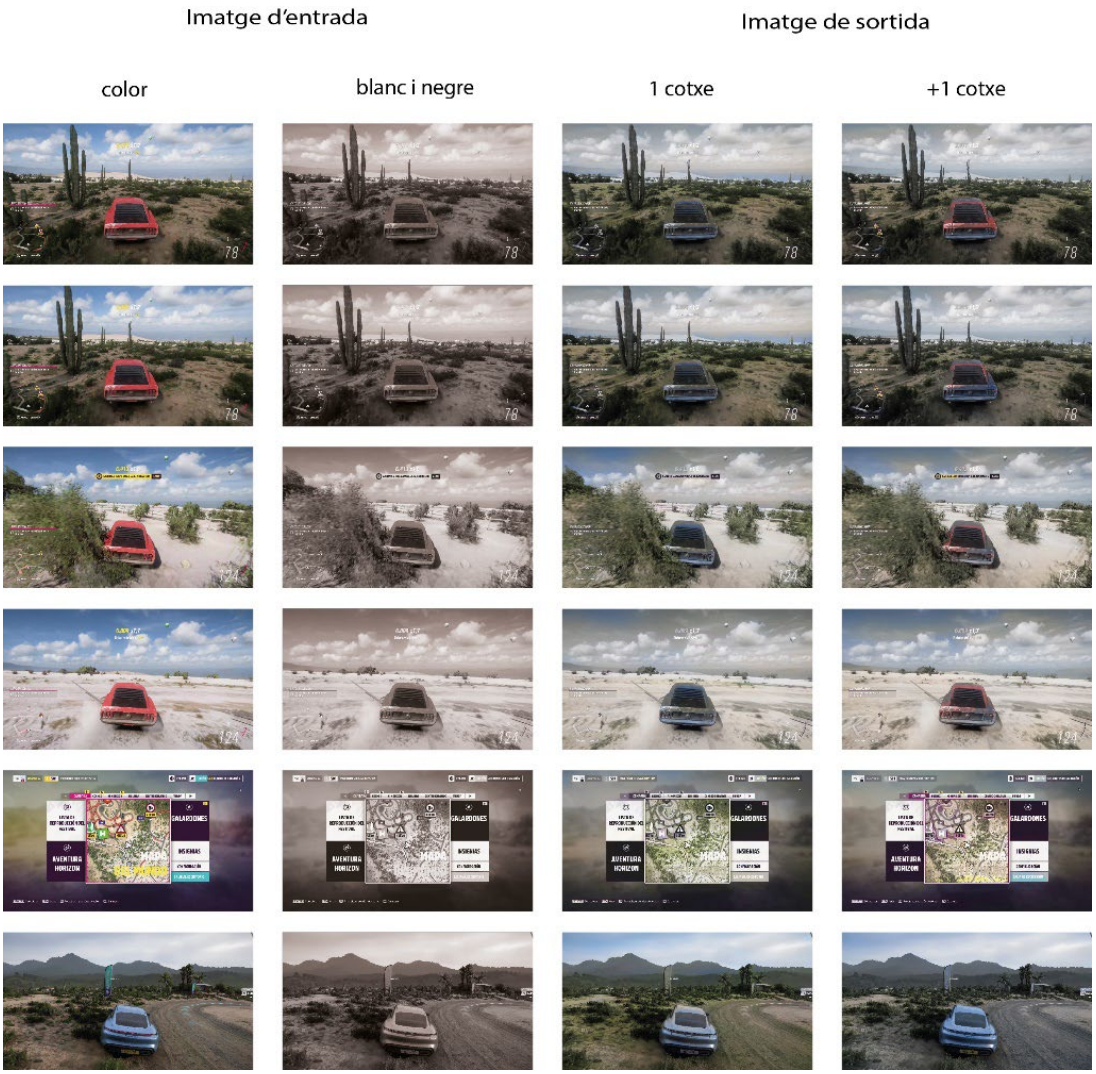
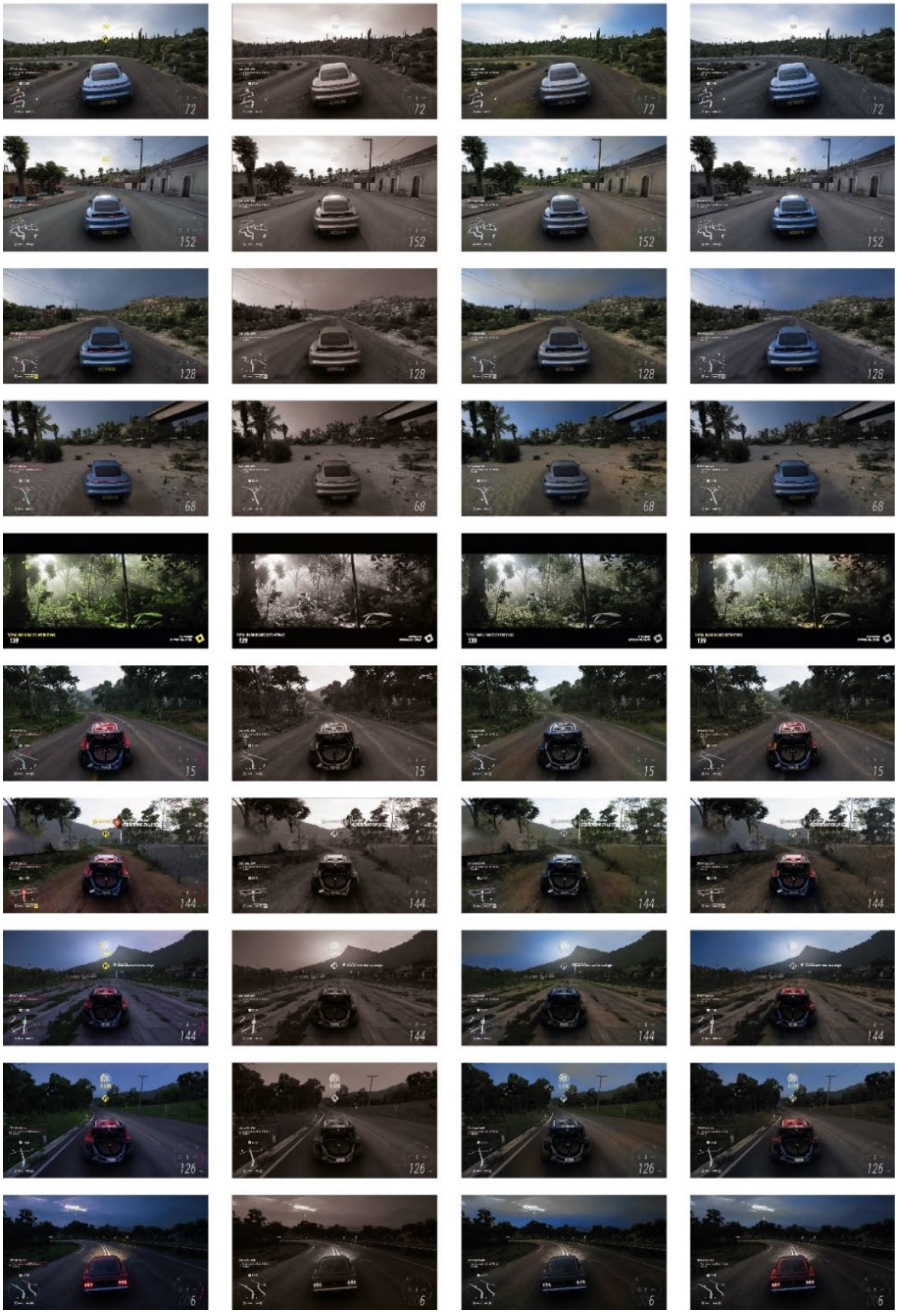


Figura 51. Frames del joc Forza Horizon 5 en color, blanc i negre, resultat de l'entrenament amb un cotxe i un escenari i resultat de l'entrenament de 3 cotxes i tres escenaris diferents



Entrenament realitzat	SSIM
1 cotxe 1 escenari	0.9461
3 cotxes 3 escenaris	0.9521

Taula 14. Resultat del SSIM, comparant l'entrenament de un cotxe en un escenari i tres cotxes i tres escenaris

Tant revisant els resultats obtinguts a la Figura 51 com revisant les mesures quantitatives presentades a la Taula 14, la segona xarxa dona millors resultats. Es poden trobar *frames* en què la diferència entre els dos sigui més alta en el segon entrenament però, al tenir més dades per entrenar pinta millor, no només els vehicles sinó també els escenaris. Les zones desèrtiques són l'exemple més clar, la xarxa entrenada amb només un escenari, ho deixa tot molt gris en comparació al resultat obtingut en diversos escenaris. Cap dels dos ha vist aigua i per tant, es veu com no sap la xarxa com reaccionar i per tant els que tenen aigua no pinta la zona en concret. Una limitació que s'ha vist és, que zones molt similars i que, el principal element diferenciador és el color, no les pot pintar correctament, per exemple les icones del minimapa.

La següent prova, intentava analitzar si seria possible transferir els colors del *Forza* a un joc. Això permetria tenir una millor idea si es podria emprar aquesta tècnica en els videojocs, ja que es podrien entrenar xarxes i tenir diferents versions del joc segons la xarxa que es triï. També seria una manera similar de reproduir el sistema de *Stadia* en que el servidor enviï la imatge en blanc i negre i el dispositiu de l'usuari té la xarxa per pintar-ho amb les textures o l'estil corresponent. El resultat obtingut ha sigut el següent:

Figura 52. Resultats de pintar vídeos de dins del joc A) Resultat de pintar-ho amb 1 cotxe i un escenari b) Resultat de pintar-ho amb diversos cotxes i diversos escenaris

Els resultats obtinguts són acceptables, es transfereixen els colors parcialment i, s'obtenen resultats diferents segons l'entrenament que ha tingut la xarxa. Les coses a tenir en compte seria que zones amb textures relativament diferents amb les que s'ha entrenat la xarxa pot fer que es confongui, pintant zones molt planes com a zones d'aigua i pintant l'aigua com si fos terra. Ara bé, s'ha de tenir en compte si aquestes operacions també es poden realitzar en temps real ja que ara s'han pintat els vídeos passant-ho per el sistema que té el *DeOldify* però hem de saber si es pot fer en temps real per tal d'aplicar-ho als videojocs.

6.3.3 Proves dins del joc

Les proves dins del joc han presentat una lleugera dificultat, ja que, és un còmput complex. Per tant, al realitzar les proves s'ha trobat una limitació, com més resolució és la imatge que es tracta més li costarà processar-la però, també més resolució tindrà i al revés, com menys petita és la imatge que es passa per la xarxa millor funciona en temps real però la resolució del joc és pitjor. S'haurà d'explorar maneres de millorar el rendiment.

La primera prova intenta solucionar aquest problema exportant la xarxa aquest cop a 512 per 512 píxels. Amb la xarxa processant imatges d'aquestes dimensions ens permet obtenir relativament bons resultats. D'altra banda s'ha trobat un problema d'eficiència ja que els *fps*[30] són molt pocs i, per tant no es podria emprar en un videojoc real i donar una sensació acceptable amb el hardware utilitzat per aquest TFG, en un sistema més potent i especialitzat podria anar molt més ràpid.

Figura 53. Processat en temps real dins del joc, passant la xarxa neuronal amb dimensió de 512x512 píxels. A) Xarxa entrenada amb 1 cotxe i 1 escenari b) Xarxa entrenada amb diferents cotxes i diferents escenaris.

[30] *Frames per second*, nombre d'imatges per segon

La segona prova ha sigut posant-hi la xarxa, exportada amb dimensions 256 per 256 píxels. Això permet que la seva execució sigui molt més ràpida però, en escalar-ho per cobrir tota la pantalla el resultat es veu pixelat.

Figura 54. Processat en temps real dins del joc, passant la xarxa neuronal amb dimensió de 256x256 píxels. A) Xarxa entrenada amb 1 cotxe i 1 escenari b) Xarxa entrenada amb diferents cotxes i diferents escenaris.

Per tant, per aquesta part, es veu que falten millores en el rendiment o millores en la forma de treballar en la xarxa neuronal amb *Unity*. Tot i així, es veu que si que es transmeten els colors del joc Forza a la demo creada. Les característiques més rellevants són:

- Els objectes clarament definits en forma, textura i posició i que, es pinten de forma correcta. El clar exemple són els arbres o el cel, on el color és correcte i l'objecte està pintat en la zona correcta.

- Els objectes que es poden definir clarament però no s'ha vist la forma concreta. L'exemple d'aquest cas és el vehicle, en que en certs entrenaments no pinten tot el cotxe completament o fan una mescla de colors tot i ser un cotxe amb color uniforme.

- Els objectes més difícils de definir sense informació de color. El cas clar és el terra i el llac. Aquestes zones són difícils de diferenciar, només es té la informació de textura però, per els casos vists, no arriba a ser informació suficient per diferenciar-los i pintar-los correctament.

7 Conclusions

Dels objectius que s'han comentat al principi d'aquest document, ara s'exposaran els diferents punts comentats en l'apartat d'objectius i s'indicarà si s'han pogut solucionar, per tal de remarcar que s'ha aconseguit segons el que es tenia planejat i, també per poder definir quina feina quedaria pendent per fer-la en un futur.

- La col·lecta de dades: s'ha pogut dur a terme de forma correcta i, per tant, s'ha pogut desenvolupar tot el treball posterior. Amb les dades també s'han pogut modificar les dades per tal d'obtenir dades sintètiques per poder entrenar correctament les xarxes neuronals.

- Proposta de *Deep Learning* per corregir soroll: s'ha aconseguit trobar una arquitectura que dugui a terme aquesta tasca. S'han pogut fer diversos experiments amb aquesta xarxa i s'ha après com funcionen les xarxes neuronals per tal de poder dur a terme la segona part del treball.

- Proposta de *Deep Learning* per pintar imatges: s'ha pogut investigar quin és el funcionament d'aquesta arquitectura i com es pot emprar per a posar-hi color a les imatges, vídeos i finalment poder-ho posar en el motor de jocs *Unity*.

- Realització del conjunt d'experiments a les dues xarxes: canviar els paràmetres per tal de veure si s'obté millors resultats. S'han pogut dur a terme un conjunt de tests en cadascuna de les xarxes i s'ha vist com modificar els paràmetres afecta el resultat. També s'ha pogut analitzar l'entrenament amb unes dades i quin efecte té veure noves dades, com és el que s'ha provat amb la *demo*.

- Treballar el pintat d'imatges en vídeos: s'ha pogut veure que en vídeos el funcionament era correcte i s'arribava al nivell de detall que es volia arribar, obtenint resultats molt realistes i en poca incoherència temporal. També s'ha pogut passar a la demo que s'ha realitzat i així veure com és la importació de la xarxa a Unity i com funciona l'execució. Al posar-ho dins del joc també s'ha pogut veure el seu funcionament en temps real.

- Realització de la *demo*: utilitzant un motor de jocs, s'ha pogut aprendre com funciona la importació de xarxes neuronals al motor *Unity* .Un cop importat, s'ha après a com es

pot utilitzar la xarxa per passar-li els *frames* com a input i posar els *frames* resultants de la xarxa a la pantalla. També s'ha vist com muntar l'escenari per a que s'assembli el joc i s'ha tractat els paquets importats per tal que no hi hagi problemes, principalment d'escala.

- Anàlisi de resultats: amb la primera xarxa s'ha pogut mirar com funciona el sistema d'anàlisi quantitatiu així com s'han descobert mesures que no es coneixien. També ha permès entendre com es duen a terme les anàlisis de resultats.

- Redacció de la memòria: s'ha pogut completar la memòria incloent-hi tota la informació necessària per a poder entendre la feina feta i quins camins es poden seguir si es vol desenvolupar o millorar el sistema actual.

S'està molt content de la feina que s'ha pogut dur a terme, s'han obtingut molts coneixements sobre les xarxes neuronals, sobretot en com estan estructurades i quin objectiu tenen. Ha pogut ser una ampliació dels coneixements obtinguts a AIPI i Visió per Computador, entenent millor el funcionament de les imatges i, relacionant les eines que s'han vist en aquestes assignatures com serien les convolucions amb les xarxes de *Deep Learning*. Com a punts a millorar, hagués estat bé tenir una millor organització del temps per tal de dedicar-hi més temps a la *demo* i, millorar així el resultat obtingut. Així com també hagués sigut una bona manera de concloure el treball tenir un sistema com el *Stadia* aprofitant l'esquema de client servidor que es tenia per l'assignatura de Sistemes Multijugador però, a la vegada, quedava una mica fora de l'àmbit d'aquest TFG.

7.1 Treball futur

De treball futur es volen nomenar un conjunt de millores que s'han anat trobant a mesura que s'anava desenvolupant el projecte. A més, es vol nomenar diverses tècniques que es podrien provar per tal d'aconseguir millors resultats.

Primer, durant la realització del projecte, s'ha publicat un nou treball d'investigació, amb el paper *Bringing Old Films Back to Life* [21] desenvolupat pels que han realitzat el *Bringing Old Photos back To Life*. Aquest és un paper que s'ha tret aquest 2022 presentat a la *CVPR*[31] que, correspon a una millora del seu antic paper per tal de poder tractar vídeos. Aquest nou paper,

[31] Conference on Computer Vision and Pattern Recognition

s'hauria d'estudiar per tal d'aplicar aquests nous models als vídeos com s'ha realitzat en aquest TFG.

Una altra qüestió que es podria millorar com a treball futur, és el funcionament de la xarxa dins del joc. Revisar si es pot accelerar l'execució de la xarxa o aprofitar els *frames* i realitzar transicions entre uns *frames*. També seria interessant poder comprovar el funcionament en diverses targetes gràfiques per veure si així es podrien solucionar els dos problemes que es té actualment: que tenir més d'una targeta gràfica per jugar no aporta millor rendiment i que l'execució de la forma que està dissenyada actualment és molt lenta. També es pot analitzar quin és l'impacte del paràmetre *render factor* i, mirar quin és el seu impacte i com es pot fer per canviar-lo a la xarxa exportada.[32]

Una altra punt interessant, seria l'estudi de com integrar aquestes eines dins d'una plataforma *could computing*. Per exemple, analitzant com es passarien les dades del servidor al client, es passaria la imatge i el client la pintaria o bé, es passarien les dades codificades (en *latent space*) i el client només tindria el generador.

[32] Issue al repositori de DeOldify sobre l'exportació on es comenten estratègies que es podrien provar

8 Agraïments

Per la realització d'aquest treball ha estat indispensable l'ajuda del tutor, el Dr. Xavier Llado Bardera que ha estat molt disponible i ha ajudat molt en guiar el treball. També agrair a Jordi Corominas, el CEO d'Additio, per donar flexibilitat en els horaris de les pràctiques en empresa per poder dur a terme el TFG. Finalment, agrair a Núria Puig, Miquel Sangrador i Miguel Villalobos per revisar el treball i donar-me suport moral en moments estressants.

9 Bibliografia

[1] Google, «Stadia,» [En línia]. Available: https://stadia.google.com/. [Últim accés: 19 Abril 2022].

[2] NVIDIA, «GeForce Now,» [En línia]. Available: https://www.nvidia.com/en-us/geforce-now/. [Últim accés: 19 Abril 2022].

[3] B. Z. D. C. P. Z. D. C. J. L. i. F. W. Ziyu Wan, «Old Photo Restoration via Deep Latent Space Translation,» *arXiv,* 20 Abril 2020.

[4] Mathworks, «fullyConnectedLayer,» [En línia]. Available: https://es.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.fullyconnectedlayer.html. [Últim accés: 22 Abril 2022].

[5] T. Wood, «What is an Activation Function?,» [En línia]. Available: https://deepai.org/machine-learning-glossary-and-terms/activation-function. [Últim accés: 22 Abril 2022].

[6] A. Z. a. Z. C. L. a. M. L. a. A. J. Smola, Dive into Deep Learning, 2020.

[7] IBM, «Convolutional Neural Networks,» 20 Octubre 2020. [En línia]. Available: https://www.ibm.com/cloud/learn/convolutional-neural-networks. [Últim accés: 17 4 2022].

[8] jantic, «DeOldify,» 2 Novembre 2021. [En línia]. Available: https://github.com/jantic/DeOldify. [Últim accés: 19 Abril 2022].

[9] Atlassian, «¿Qué es scrum?,» [En línia]. Available: https://www.atlassian.com/es/agile/scrum. [Últim accés: 17 4 2022].

[10] A. Z. Karen Simonyan, «VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION,» *arXiv,* 2015.

[11] Y. Dong, «ResBlock — A simple fix helped make deep networks possible,» 24 Septembre 2019. [En línia]. Available: https://medium.com/ai%C2%B3-theory-practice-business/resblock-a-trick-to-impove-the-model-8ba11891c52a. [Últim accés: 23 Febrer 2022].

[12] X. Z. S. R. J. S. Kaiming He, «Deep Residual Learning for Image Recognition,» *arXiv,* 10 Desembre 2015.

[13] T. Wood, «What is Backpropagation?,» [En línia]. Available: https://deepai.org/machine-learning-glossary-and-terms/backpropagation. [Últim accés: 22 Abril 2022].

[14] Google, «Overview of GAN Structure,» 24 Maig 2019. [En línia]. Available: https://developers.google.com/machine-learning/gan/gan_structure. [Últim accés: 3 Març 2022].

[15] J.-Y. Z. T. Z. i. A. A. E. Phillip Isola, «Image-to-Image Translation with Conditional Adversarial Networks,» *arXiv,* 26 Novembre 2018.

[16] I. G. D. M. i. A. O. Han Zhang, «Self-Attention Generative Adversarial Networks,» *arXiv,* 14 Juny

2019.

[17] N. Jean, «Fréchet Inception Distance,» 15 Juliol 2018. [En línia]. Available: https://nealjean.com/ml/frechet-inception-distance/. [Últim accés: 7 Maig 2022].

[18] Geek for Geeks, «Python | Peak Signal-to-Noise Ratio (PSNR),» 6 Febrer 2020. [En línia]. Available: https://www.geeksforgeeks.org/python-peak-signal-to-noise-ratio-psnr/. [Últim accés: 7 Maig 2022].

[19] D. Pranjal, «All about Structural Similarity Index (SSIM): Theory + Code in PyTorch,» 3 Setembre 2020. [En línia]. Available: https://medium.com/srm-mic/all-about-structural-similarity-index-ssim-theory-code-in-pytorch-6551b455541e. [Últim accés: 7 Maig 2022].

[20] P. I. A. A. E. E. S. O. W. Richard Zhang, «The Unreasonable Effectiveness of Deep Features as a Perceptual Metric,» *arXiv,* 10 Abril 2016.

[21] B. Z. D. C. J. L. Ziyu Wan, «Bringing Old Films Back to Life,» *arxiv,* 2022.

[22] L. M. i. S. C. Alec Radford, «Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks,» *arXiv,* 7 Gener 2016.

[23] O. R. P. F. i. T. Brox, «U-Net: Convolutional Networks for Biomedical Image Segmentation,» *arXiv,* 18 Maig 2015.

[24] M. H. H. R. T. U. B. N. i. S. H. Hochreiter, «GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium,» *arXiv,* 12 Gener 2018.

[25] A. Mikhailiuk, «Deep Image Quality Assessment,» 15 Març 2021. [En línia]. Available: https://towardsdatascience.com/deep-image-quality-assessment-30ad71641fac. [Últim accés: 15 Març 2022].

[26] R. Ocampo, «Automatic Image Quality Assessment in Python,» 28 Agost 2018. [En línia]. Available: https://towardsdatascience.com/automatic-image-quality-assessment-in-python-391a6be52c11. [Últim accés: 14 Març 2022].

[27] D. M. K.Silpa, «COMPARISON OF IMAGE QUALITY METRICS,» *INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT),* vol. 1, núm. 4, 30 Juny 2012.

[28] M. Nayak, «An Introduction To Conditional GANs (CGANs),» 9 Maig 2019. [En línia]. Available: https://medium.datadriveninvestor.com/an-introduction-to-conditional-gans-cgans-727d1f5bb011. [Últim accés: 12 Febrer 2022].

[29] E. Kan, «What The Heck Are VAE-GANs?,» 17 Agost 2018. [En línia]. Available: https://towardsdatascience.com/what-the-heck-are-vae-gans-17b86023588a. [Últim accés: 26 Decembre 2021].

[30] J. Rocca, «Understanding Variational Autoencoders (VAEs),» 24 Septembre 2019. [En línia]. Available: https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73. [Últim accés: 27 Decembre 2021].

[31] N. Tomar, «What is UNET?,» 19 Gener 2021. [En línia]. Available:

https://medium.com/analytics-vidhya/what-is-unet-157314c87634. [Últim accés: 26 Gener 2022].

[32] W. contributors, «Machine learning,» 2022. [En línia]. Available: https://en.wikipedia.org/wiki/Machine_learning&oldid=1081891971. [Últim accés: 13 Abril 2022].

[33] W. Contributors, «Computer vision,» 9 Abril 2022. [En línia]. Available: https://en.wikipedia.org/w/index.php?title=Computer_vision&oldid=1081781547. [Últim accés: 14 Abril 2022].

[34] W. contributors, «Artificial intelligence,» 15 Abril 2022. [En línia]. Available: https://en.wikipedia.org/w/index.php?title=Artificial_intelligence&oldid=1082866670. [Últim accés: 16 Abril 2022].

[35] W. contributors, « Deep learning,» 12 Abril 2022. [En línia]. Available: https://en.wikipedia.org/w/index.php?title=Deep_learning&oldid=1082296147. [Últim accés: 15 Abril 2022].

[36] W. contributors, «Artificial neural network,» 12 Abril 2022. [En línia]. Available: https://en.wikipedia.org/w/index.php?title=Artificial_neural_network&oldid=1082329948. [Últim accés: 14 Abril 2022].

[37] W. contributors, «Supervised learning,» 24 Febrer 2022. [En línia]. Available: https://en.wikipedia.org/w/index.php?title=Supervised_learning&oldid=1073728937. [Últim accés: 15 Abril 2022].

[38] W. contributors, «Unsupervised learning,» 30 Març 2022. [En línia]. Available: https://en.wikipedia.org/w/index.php?title=Unsupervised_learning&oldid=1080102229. [Últim accés: 15 Abril 2022].

[39] A. Sharma, «Original U-Net in PyTorch,» 17 Març 2022. [En línia]. Available: https://towardsdev.com/original-u-net-in-pytorch-ebe7bb705cc7. [Últim accés: 28 Març 2022].

[40] Naoki, «Up-sampling with Transposed Convolution,» 13 Novembre 2017. [En línia]. Available: https://naokishibuya.medium.com/up-sampling-with-transposed-convolution-9ae4f2df52d0. [Últim accés: 18 Abril 2022].

[41] V. Feng, «An Overview of ResNet and its Variants,» 15 Juliol 2017. [En línia]. Available: https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035. [Últim accés: 20 Març 2022].

[42] P. F. T. B. Olaf Ronneberger, «U-Net: Convolutional Networks for Biomedical Image Segmentation,» *arXiv,* 18 Maig 2015.

[43] N. Jean, «Fréchet Inception Distance,» 15 Juliol 2018. [En línia]. Available: https://nealjean.com/ml/frechet-inception-distance/. [Últim accés: 7 Maig 2022].

10 Manual d'usuari i/o instal·lació

Per tal de provar el joc:

- WASD o ↑ ← ↓ → per moure el vehicle.
- B: per activar o desactivar la xarxa neuronal.
- Botó ESC: per sortir del joc. A vegades es queda congelat al sortir del joc. No s'ha trobat la causa.
- 1: Activar la xarxa neuronal 256 px, 1 cotxe i 1 escenari
- 2: Activar la xarxa neuronal 256 px, 3 cotxes i 3 escenaris
- 3: Activar la xarxa neuronal 512 px, 1 cotxe i 1 escenari
- 4: Activar la xarxa neuronal 512 px, 3 cotxes i 3 escenaris

Per problemes tècnics, al canviar de xarxa neuronal pot petar el joc. Per aquesta raó hi ha quatre executables per a que es pugui provar cadascuna de les xarxes si apareix el problema. Es suposa que el problema es degut a l'assignació de la memòria RAM.

El joc ha estat desenvolupat en la versió 2019.4.38f1 de Unity

L'executable només funciona per Windows. Per tal de poder-lo utilitzar:

- Descarregar el ZIP.
- Descomprimir el ZIP en una carpeta.
- Executar el fitxer "DemoDeoldify.exe".
- Si apareix un missatge del sistema indicant que no es coneix l'autor per tal d'executar el programa cal anar a "més informació" i prémer "Executar".
- El joc hauria d'obrir-se directament en la escena.