UNIVERSITAT DE BARCELONA

Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

UNA REVISIÓ DE NEURAL STYLE TRANSFER

Gerard Puch Camacho

Director: Eloy García
Realitzat a: Departament de Matemàtiques i Informàtica

Barcelona, 13 de juny de 2022

Agraïments

Vull agrair al meu tutor Eloy Garcia Marcos tot l'ajut que m'ha donat, tant coneixements com assessorament a la hora de definir, realitzar i corregir aquest treball, sense ell no hagués estat possible fer-lo.

Vull agrair també al professor Simone Balocco, per l'assessorament en vers a tot el funcionament del TFG, també per ajudar-me a trobar un tutor adequat per l'àmbit en el que volia dedicar el treball, i per transmetre tots els coneixements que vaig adquirir en l'assignatura de Processament d'imatge en el primer semestre d'aquest curs.

Agrair també als meus amics que m'han cedit les seves fotografies per a poder realitzar la part pràctica d'aquest treball, Guillem Bargalló, amb les fotografies de la gossa Clemen, i també a la Cristina Subirà per la fotografia del seu retrat.

Finalment, també vull agrair a tota la gent que m'ha donat suport i ànims durant la realització d'aquest treball, família, companys de pis i amics, moltes gràcies!

Resum

En els darrers anys, ha sorgit una tecnologia capaç de simular el raonament humà i desenvolupar tasques tant complexes que la programació convencional no era capaç de realitzar. Estem parlant de les xarxes neuronals, algorismes de Deep Learning que tenen ús en molts àmbits diversos i amb molts bons resultats, com pot ser: la predicció de successos i simulacions de tot tipus, reconeixement i classificació d'elements, processament de dades i modelització, enginyeria de control, intel·ligència artificial, reconeixements facial; i molts més.

En aquest treball ens centrarem en un tipus concret de xarxes, les xarxes neuronals convolucionals, capaces d'interpretar i classificar imatges. Estudiarem en profunditat i revisarem el problema del Style Transfer a través d'aquesta tecnologia neuronal. Diferenciant així dos fases en el treball, una fase d'investigació i recerca i una fase pràctica on implementar i experimentar amb aquesta tecnologia. Finalment es realitzarà un anàlisi del funcionament, rendiment, i eficiència d'aquesta tecnologia i elavorarem una valoració.

Abstract

In recent years, a technology has emerged, capable of simulating human reasoning and performing complex tasks that conventional programming was unable to perform. We are talking about neural networks, Deep Learning algorithms that are used in many different fields and with many good results, such as: prediction of successes and simulations of all kinds, recognition and classification of elements, data processing and modeling, control engineering, artificial intelligence, facial recognition; and many more.

In this work we will focus on a specific type of network, convolutional neural networks, capable of interpreting and classifying images. We will study in depth and review the problem of Style Transfer through this neural technology. Thus differentiating two phases in the work, a phase of investigation and research and a practical phase where to implement and experiment with this technology. Finally, an analysis of the operation, performance, and efficiency of this technology will be performed and an assessment will be made.

Resumen

En los últimos años, ha surgido una tecnología capaz de simular el razonamiento humano y desarrollar tareas tan complejas que la programación convencional no era capaz de realizar. Estamos hablando de las redes neuronales, algoritmos de Deep Learning que tienen uso en muchos ámbitos diversos y con muy buenos resultados, como puede ser: la predicción de sucesos y simulaciones de todo tipo, reconocimiento y clasificación de elementos, procesamiento de datos y modelización , ingeniería de control, intel·ligència artificial, reconocimientos facial; y muchos más.

En este trabajo nos centraremos en un tipo concreto de redes, redes neuronales convolucionales, capaces de interpretar y clasificar imágenes. Estudiaremos en profundidad y revisaremos el problema

del Style Transfer a través de esta tecnología neuronal. Diferenciando así dos fases en el trabajo, una fase de investigación e investigación y una fase práctica en la que implementar y experimentar con esta tecnología. Por último se realizará un análisis del funcionamiento, rendimiento, y eficiencia de esta tecnología y elaboraremos una valoración.

Continguts

1. Introducció

La intel·ligència artificial és una de les branques de la informàtica en la que sempre s'ha volgut progresar, i des de fa menys d'una dècada s'han fet molts avenços. Es considera que es va establir com a disciplina en la Conferencia de Dartmouth, trobada que va tenir lloc l'estiu de l'any 1956 a la universitat Dartmouth College, ubicada a Hanover, Nou Hampshire (Estats Units).

Però, ja l'any 1950, Alan Turing va presentar el seu famós Test de Turing, un exàmen per discernir si una màquina mostra o no un comportament intel·ligent. Veiem aquest test llavors com la primera definició de intel·ligència artificial.

Tot i que, des de llavors, s'ha anat desenvolupant l'àmbit de la intel·ligència artificial durant tots aquests anys, fluctuant entre períodes irregulars de poc i molt progrés. No és fins l'any 2011 que es començaria mantenir un progrés considerablement major, gràcies la intel·ligència artificial Watson d'IBM, una IA capaç de respondre preguntes formulades en llenguatge natural i creada per l'equip de DeepQA, dirigit per David Ferrucci.

El 2013 es va començar a utilitzar les IA en l'àmbit mèdic per prendre decisions sobre l'autorització de tractaments de càncer de pulmó per la sanitat estatunidenca, un gran incentiu per motivar a investigadors a fer recerca i millorar la tecnologia. D'aquesta manera, a dia d'avui, la intel·ligència artificial ha passat a ser una eina indispensable pel desenvolupament en moltes aplicacions diferents.

L'any 2015, es publica un article anomenat "A Neural Algorithm of Artistic Style", escrit per Leon A. Gatys, Alexander S. Ecker i Matthias Bethge [1], que fa una primera aproximació al problema de la transferència d'estil mitjançant l'ús de les xarxes neuronals, una sub-branca del Deep Learning.

Dins del camp del Deep Learning trobem les xarxes neuronals, una arquitectura computacional que imita el procés biològic del cervell, constituint-ne molts nodes, anomenats neurones, que aprenen i creen patrons propis a partir de les dades que se li passen en un procés d'aprenentatge per a poder processar posteriorment qualsevol dada arbitrària que se li introdueixi.

Aquest treball de fi de grau és una oportunitat per aprendre i fer un primer apropament al món de la intel·ligència artificial avançada, centrant-nos en l'àmbit de les xarxes neuronals per a la resolució d'un problema concret, la transferència d'estil entre imatges, conegut també com a Neural Style Transfer, que explora també una vessant artística que creix i es realitzen grans avenços dia a dia.

1.1 Motivació

Des de sempre, he sigut una persona amant de l'art visual, tant en dibuixos animats, pintures, esquetxos, art digital, etc. Això ha definit gran part dels meus gustos que m'han marcat aspiracions a la meva vida. Com la de ser dibuixant i aprendre conceptes sobre aquest tema. De fet, tinc plantejat, després d'acabar aquesta carrera, formar-me en art digital i belles arts.

Per aquest motiu, de tots els anys que he estat a la carrera, les assignatures que més m'han agradat i implicat han sigut les relacionades amb la visió artificial, com l'assignatura que dur aquest mateix nom, Gràfics i visualització de dades i Processament d'imatges.

Per altra banda, els àmbits del Machine i Deep Learning han sigut un dels temes que no s'han aprofundit gaire i que sempre m'han cridat l'atenció des de que vaig començar la carrera. A més, són una tecnologia amb una gran important a dia d'avui, i tot sembla indicar que aquesta anirà en augment d'ara en endavant, tot aprenentatge que pugui fer al respecte és benvingut.

Abans de plantejar aquest treball no coneixia el que eren les xarxes neuronals, va ser el meu tutor, especialitzat en elles, qui em va proposar aquest àmbit, ja que s'adaptava perfectament a les meves motivacions. D'aquí va sorgir la idea de realitzar un treball centrat en el Neural Style transfer.

1.2 Objectius

Els objectius plantejats en aquest treball, com ja he explicat en l'apartat anterior, giren en torn a ampliar els meus coneixements en la matèria de visió i intel·ligència artificial. Els objectius que proposo, són:

- Explorar les capacitats que té la tecnologia actual per emular obres d'art i reproduir conceptes artístics.

- Aprendre des de cero el funcionament teòric de les xarxes neuronals convolucionals.

- Familiaritzar-me amb tots els conceptes que s'utilitzen a la hora de treballar amb xarxes neuronals
.
- Utilitzar tot el que aprengui durant la investigació d'aquest treball per poder treballar amb xarxes neuronals.

- Entrenar la meva pròpia xarxa neuronal convolucional per a que sigui capaç de reconstruir imatges amb un estil artístic diferent de la original.

- Realitzar un experiment per comprovar l'eficàcia de la tecnologia i elaborar un anàlisi.

1.3 Estructura de la memòria

La resta d'aquest document s'organitza seguint la següent estructura:

- Capítol 2. Revisió del Neural Style Transfer. En aquest capítol es fa una breu introducció sobre el style transfer i l'ús de les xarxes neuronals per a aquesta tasca.

- Capítol 3. Implementació: Es detalla com està elaborat el codi de la part pràctica del treball, la seva instalació i configuració.

- Capítol 4. Metodologia: En aquest capítol es defineix quin és l'experiment a seguir, així com quins recursos es fan servir i amb quina finalitat.

- Capítol 5. Resultats: Detalla les els resultats aconseguits amb els passos descrits en el capítol anterior i defineix l'anàlisi d'aquests

- Capítol 6. Conclusions i futurs treballs.

2. Revisió del Neural Style Transfer

En aquesta secció es sintetitzaran tots els coneixements teòrics que he reunit durant la fase d'investigació d'aquest treball, per poder entendre el funcionament de les xarxes neuronals convolucionals i com poden resoldre el problema de la transferència d'estil artístic. Coneixements necessaris per a la realització de la part pràctica d'aquest treball.

2.1 Conceptes previs

Abans d'entrar en matèria i exposar els continguts teòrics referents al Style transfer, es convenient repasar alguns conceptes bàsics sobre el processament d'imatge, àmbit en el que ens trobem.

Durant el desenvolupament d'aquest treball s'ha treballat principalment amb imatges. Una imatge, en l'entorn digital, no deixa de ser una matriu numèrica. normalment amb una dimensionalitat de 3 dimensions seguint el format "(amplada en píxel, alçada en píxel, nombre de canals)".

Tenim per tant, que cada píxel de la imatge és una llista ordenada de N elements, un per cada canal d'aquesta, en el cas del RGB (el format més comú), el primer element pel canal vermell (R), el segon pel verd (G) i el tercer pel blau (B). La superposició dels valors en cada canal del píxel construeix el color que representa aquest.

Figura 1: Representació d'un píxel RGB en una matriu 3D [26].

El valor del píxel, que representa la intensitat del color en el canal al que pertany, sol està comprès entre 0 i 255 en nombres enters de 8 bits, també és habitual reescalar els valors entre 0 i 1, convertits en un format numèric decimal com pot ser el float. L'ús de valors petits ajuda a disminuir el cost de càlcul de la xarxa neuronal el que millora el seu funcionament.

Una convolució és una operació matemàtica lineal que s'aplica entre dos matrius 2D de qualsevol tamany i dóna com a resultat una tercera matriu amb la superposició de les dues. Aquesta operació és molt utilitzada en el processament d'imatge amb l'aplicació de filtres lineals. Depenent de com sigui la segona matriu, denominada normalment com a màscara o filtre, la convolució tindrà una utilitat o una altra. Existeixen molts tipus de filtres que es poden definir per a fer convolucions: suavitzar imatges, eliminar soroll, realçar la imatge, detectar vores, etc.

Haver repassat aquests conceptes bàsics facilitarà la comprensió del contingut teòric exposat a continuació.

2.2 El problema del Style transfer

El problema de la transferència d'estil (Style transfer) consisteix en agafar una imatge qualsevol i transformar-la a un estil completament diferent, per exemple, a partir d'una imatge capturada en un entorn natural, mitjançant una càmera fotogràfica, transformar-la a un estil artístic concret, simulant, per exemple, al quadre d'un pintor.

Des de un enfoc tradicional, es fan servir filtres i transformacions de color i textura sobre una imatge, per a emular un estil artístic concret sobre la imatge. En els darrers anys, però, l'algorisme ha anat evolucionant, prescindint de transformacions lineals i fent ús de la creixent tecnologia Neural Network, concretament de les Xarxes neuronals convolucionals (CNN, acrònim de l'anglès de Convolutional Neural Networks).

La idea principal és utilitzar dues imatges, una de contingut i una d'estil, per donar lloc a una tercera imatge que serà el resultat de la transferència. A aquestes imatges se les sol anomenar com content image, style image i output image. Agafant la output image (definida inicialment com un 'placeholder') per fer-la semblar a la content image però amb l'estil artístic de la style image.

Al treballar amb xarxes neuronals, és necessari utilitzar el que se'n diu un model entrenat, una instància de una xarxa neuronal que ha sigut sotmesa a un seguit d'iteracions d'un càlcul concret, passant-li les operacions a fer amb les seves solucions, amb la finalitat de que a cada iteració del procés, les neurones de la xarxa vagin ajustant el pesos que les relacionen entre sí. D'això se'n diu entrenament d'un model (o xarxa neuronal).

El problema del Style Transfer defineix i substreu les característiques de les dos imatges a processar, elaborant una funció de pèrdua per mesurar la transformació de la imatge respecte al seu valor original. En aquest cas mesurant les característiques del contingut d'una imatge (contorns, formes, distribució, etc.) i de l'estil de l'altre (patrons de dibuix, colors, textura, etc.) i aproximant-les el màxim per reconstruir-les en una nova imatge.

Es fan dos passades a la xarxa per a realitzar la transformació, la primera serveix per extreure les característiques i filtrar-les (feature maps), elaborant amb elles els diferents valors i mesures. La segona passada serveix per reconstruir la output image amb el resultats de l'anterior iteració fent el procès invers de la primera.

En les properes seccions es realitza una introducció a les xarxes convolucionals així com al problema del style transfer.

2.3 Xarxes Neuronals

Com el seu propi nom indica, les xarxes neuronals són un mètode de càlcul que busca imitar el funcionament de les neurones en un organisme. Definint un model de nodes connectats a entre ells anomenats neurones, generen, transmeten i reforcen conceptes per a arribar a conclusions determinades i consolidar-les com a coneixement. Les xarxes neuronals en intel·ligència artificial no

fan més que traslladar el funcionament biològic a l'àmbit matemàtic. Simulant així un aprenentatge, percebent i memoritzant patrons i estructures.

Els models de xarxes neuronals serveixen per trobar paràmetres i patrons adequats mitjançant un "entrenament". Aquest consisteix a alimentar la xarxa amb conjunts de dades input i output esperat, perquè determini criteris vàlids d'anàlisis per després aplicar-los. Per exemple, les xarxes neuronals poden servir perquè un dispositiu identifiqui fotografies en les quals hi ha una bicicleta, després d'identificar les formes geomètriques que la composen.

2.3.1 Estructura d'una xarxa neuronal bàsica

La xarxa neuronal està composada de nodes, cadascun dels quals s'encarrega de rebre uns valor d'entrada específic, al rebre aquests valors, els nodes els modifiquen en funció del seu propi pes dins la xarxa. A continuació, aquests valors modificats es mouen a les següents neurones de la xarxa. D'aquesta manera, els valors es van modificant fins que s'aconsegueix una conclusió concreta de l'operació a realitzar.

Per tant, les xarxes neuronals, s'organitzen en diferents capes de processament, que són un conjunt de neurones les entrades de les quals provenen d'una capa anterior (o de les dades d'entrada, en el cas de la primera capa) i les sortides són l'entrada d'una capa posterior.

Les neurones de la primera capa reben com a entrada les dades input que alimenten a la xarxa neuronal. És per això que la primera capa es coneix com a capa d'entrada. El Output de la última capa és el resultat visible de la xarxa i es coneix com la capa de sortida. Les capes que hi ha entre la capa d'entrada i la capa de sortida es diuen capes ocultes ja que desconeixem tant els valors d'entrada i sortida que las alimenta.

Una xarxa neuronal, per tant, sempre està composta per una capa d'entrada, una capa de sortida (si només hi ha una capa a la xarxa neuronal, la capa d'entrada coincideix amb la capa de sortida) i pot contenir 0 o més capes ocultes. El concepte de Deep Learning neix arran d'utilitzar un gran nombre de capes ocultes a les xarxes.

2.3.2 Optimització de les xarxes

A la hora d'alimentar i millorar una xarxa neuronal, és necessari establir una funció d'entrenament, que defineix com restablir els pesos dels nodes de la xarxa a cada iteració. L'algoritme d'optimització més utilitzat en les xarxes neuronals se'n diu Descens de gradient. El gradient és el càlcul que ens permet saber com ajustar els paràmetres de la xarxa de manera que se'n minimitzi la desviació a la sortida.

L'algoritme compta amb diverses versions, depenent del nombre de mostres que s'introdueix a la xarxa en cada iteració:

- **Descens del gradient per lots (o batch)**: totes les dades disponibles s'introdueixen d'una vegada. Això suposarà problemes d'estancament, ja que el gradient es calcularà fent servir sempre totes les mostres, i arribarà un moment en què les variacions seran mínimes.

- **Descens del gradient estocàstic**: introdueix una única mostra aleatòria a cada iteració. El gradient es calcularà per a aquesta mostra concreta, introduint així aleatorietat i dificultant l'estancament. El problema d'aquesta versió és la seva lentitud, perquè necessita moltes més iteracions.

- **Descens del gradient (estocàstic) en minilots (o mini-batch)**: s'introdueixen N mostres a cada iteració; conservant els avantatges de la segona versió i aconseguint a més que l'entrenament sigui més ràpid degut a la paral·lelització de les operacions. És, per tant, la versió més òptima de l'algoritme i la que es farà servir per el problema del Style transfer, triant un valor de N que aporti un bon balanç entre aleatorietat i temps d'entrenament. També s'ha de contemplar que no sigui un valor gaire gran per a la memòria de la GPU disponible ja que això pot comportar problemes de d'execució.

L'algorisme de Descens de gradient següeix els següents passos:

1. Introdueix un lot d'entrada amb N mostres aleatòries provinents del conjunt de dades (dataset) d'entrenament, prèviament etiquetat (cosa que significa que coneixem la sortida real).

2. Després dels càlculs pertinents a cada capa de la xarxa, s'obté com a resultat les prediccions a la seva sortida. A aquest pas se'l coneix com a forward propagation (de les entrades).

3. S'avalua la funció de cost (o funció de pèrdua) per el lot. La funció es defineix prèviament per a poder avaluar de la manera més adequada la diferència entre les prediccions de la nostra xarxa i les sortides reals. El valor d'aquesta funció pèrdua és el que es tracta de minimitzar en tot moment mitjançant l'algorisme, i s'hi orienten els passos següents.

4. Es calcula el gradient, és la derivada multivariable de la funció de cost pel que fa a tots els paràmetres de la xarxa. Matemàticament és un vector que ens dóna la direcció i el sentit en el que aquesta funció augmenta més ràpid, per tant, l'objectiu és moure's en sentit contrari si el que tractem és de minimitzar la funció de cost.

5. Un cop obtingut el vector gradient, s'actualitzen els paràmetres de la xarxa restant al valor actual el valor del gradient corresponent, multiplicat per una taxa d'aprenentatge que permet ajustar la magnitud dels passos de la xarxa. Segons es vagi aproximant al mínim global, els passos seran teoricament més petits degut a que el pendent de la funció de cost serà menor.

Tots aquests passos es repetiran de manera successiva mentres el valor de la funció de cost i les mètriques de sortida no comencin a empitjorar de manera sostinguda.

Aquest algoritme ens dóna el valor de la pendent de les funcions de cost però no sobre la seva curvatura. Per calcular com varia el gradient i que no suposi un cost computacional molt elevat, existeixen algunes tècniques d'aproximació on s'estimen aquestes segones derivades amb un ús de memòria limitat, com és el cas de LBFGS; També s'utilitzen optimitzacions sobre el descens del gradient estocàstic (com poden ser les optimitzacions Momentum, Root mean squared propagation o RMSProp, Adam).

2.3.3 Hyperparameters

A la hora de treballar amb xarxes neuronals, és necessari poder ajustar variables i controlar l'execució, això s'aconsegueix amb els Hyperparameters. Són els paràmetres que podem modificar manualment en una xarxa neuronal, com la taxa d'aprenentatge (learning rate) utilitzat en l'algorisme d'optimització, la mida del batch, o qualsevol altre paràmetre ajustable dins de la xarxa.

L'ajust dels hiper-paràmetres d'una xarxa és molt important per aconseguir un bon exercici, i s'han de trobar els que millor s'adaptin al problema que estiguem resolent.

2.4 Xarxes Neuronals Convolucionals

Per el problema del Style transfer és gairebé imperatiu que fem servir xarxes neuronals convolucionals, compostes per una sèrie de capes d'operacions de matriu convolucional. Aquestes xarxes són ideals per a l'anàlisi d'imatges i la identificació d'objectes. Empren un concepte similar als filtres i detectors gràfics, però d'una manera molt potent i complexa. Ideals per extreure i interpretar les característiques d'una imatge.

2.4.1 Definició

Les xarxes neuronals convolucionals consisteixen en múltiples capes de filtres convolucionals d'una o més dimensions. Després de cada capa, generalment s'afegeix una funció per realitzar un mapeig de característiques. El conjunt de les diferents capes convolucionals, seguides pel filtratge i subsampling es coneix com a fase d'extracció de característiques.

La fase d'extracció de característiques es compon de capes alternes de neurones convolucionals i neurones de reducció de mostreig (o pooling). Segons avancen les dades al llarg d'aquesta fase, va disminuint la dimensionalitat d'aquestes. Per tant, les neurones en capes llunyanes són molt menys sensibles a les pertorbacions que hi puguin haver en les dades d'entrada, però, són activades per característiques cada cop més complexes.

Recordant l'estructura de les xarxes neuronals, aquestes estan divides en diverses capes de nodes, seguint un ordre específic, des de la capa d'entrada (input), passant per les capes ocultes (capes convolucionals, de pooling, classificadores), fins acabar en la capa de sortida (output) on s'obté ja el resultat del càlcul.

Les dues característiques principals de les CNN són:

- Els valors numèrics de les matrius convolucionals no estan predefinits per trobar característiques específiques de la imatge, com poden ser les vores. Aquests valors es generen automàticament durant els processos d'entrenament, de manera que podran detectar característiques més complexes que les vores.

- Tenen una estructura en capes, de manera que les primeres capes detecten característiques senzilles de la imatge (vores, blocs de color, etc.) i les últimes capes utilitzen la informació de les anteriors per detectar objectes complexos (com persones, animals, cotxes, etc.).

2.4.2 Estructura i capes

En una xarxa CNN tenim que es realitzen dos operacions dins les capes ocultes:

- **La convolució:** La convolució és una operació matemàtica que transforma dues funcions o matrius en una tercera amb la magnitud de la superposició de les dues originals. Les imatges no deixen ser matrius matemàtiques (de 2 dimensions o 3 dimensions si tenen múltiples canals) amb les que es pot operar. La convolució és una operació molt utilitzada en processament d'imatges que es fa servir també en les xarxes CNN per aplicar filtres aleatoris creats a cada capa i donar un resultat.

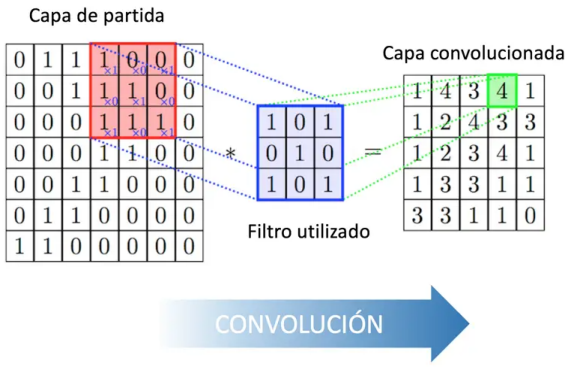


Figura 2: Representació del càlcul matemàtic de la convolució [20].

- **El Subsampling:** Després d'una capa de convolució, el número de neurones que surten és exponencialment més gran que el de l'entrada de la mateixa capa, si no es redueix el nombre de nodes podria implicar uns costos de processament exageradament grans. L'operació de subsampling consisteix en fer un filtratge dels nodes preservant els que continguin la informació més rellevant.

  El criteri que segueix aquesta operació en xarxes CNN es diu Max-Pooling, troba el valor màxim entre una finestra (del tamany que definim, en funció de la reducció que es vulgui fer) i passa aquest valor com a resum de característiques sobre aquesta àrea. Com a resultat, la mida de les dades es redueix per un factor igual a la mida de la finestra amb la que s'està treballant.

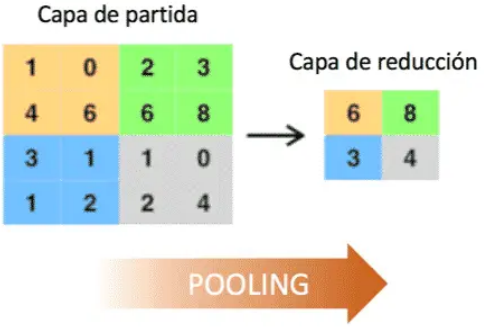


Figura 3: Exemple d'actuació d'un subsampling Max-Pooling [20].

Hi ha altres mètodes de subsampling, com l'average_pooling, on es calcula el valor mitjà dins de la finestra, en lloc del maxim, com en el cas anterior.

Per tant, l'esquema que segueix habitualment una xarxa neuronal convolucional amb les característiques que em descrit anteriorment, és el següent:

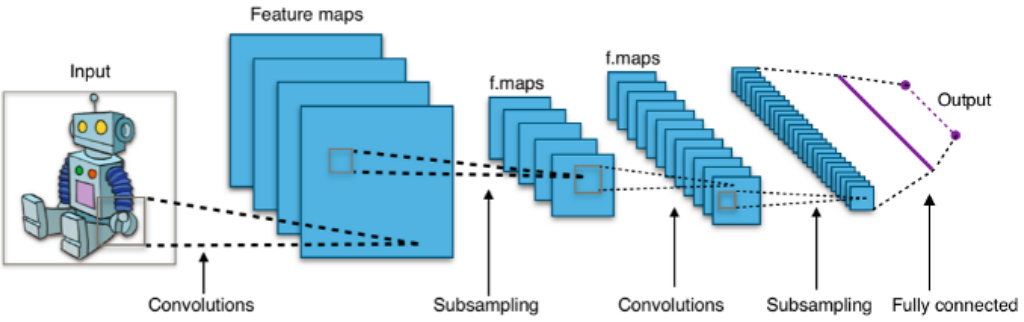


Figura 4: Exemple de xarxa CNN [29].

Observem que a mida que es va avançant les capes de la xarxa, inevitablement, el número de nodes va augmentant exponencialment. Si s'utilitza una xarxa CNN entrenada amb inputs d'un volum de dades gran, comportarà un cost computacional més alt, és un factor molt important a tenir en compte a la hora de fer servir aquest tipus de xarxes. Normalment, per executar xarxes CNN, s'utilitza la GPU del ordinador degut al seu gran nombre de nuclis i per no carregar aquesta costosa tasca a l CPU de la computadora que s'està fent servir. En aquest treball totes les xarxes CNN que s'han fet servir s'han processat a través de la GPU.

En el moment d'executar la xarxa, podem triar, a través de hyperparameters, que destacar, o bé reconstruint l'estil o bé el contingut: Un fort èmfasi en l'estil donarà lloc a imatges que coincideixen amb l'aspecte de l'obra d'art, donant-ne una versió texturitzada, però que gairebé no mostren el contingut de la fotografia.

Quan es posa un fort èmfasi en el contingut, es pot identificar la fotografia, però l'estil artistic no coincideix tant. Realitzem el descens del gradient a la imatge generada per trobar una altra imatge que coincideixi amb les respostes de les features de la imatge original.

2.4.3 Feature maps

El filtres feature map (o mapeig de característiques) són petites seccions de la imatge que contenen diferents característiques, també anomenat activation map.

El filtre llisca sobre la imatge mitjançant l'operació de convolució i genera una feature map a cada posició. Cada mapeig captura diferents característiques de la mateixa imatge. Amb un nombre de

filtres major, podrem generar més features. Un cop extrets els filtres de la imatge, el nombre de filtres utilitzats com a input d'una capa hauria de crear el mateix nombre de feature maps. Així, una imatge d'entrada amb 6 filtres tindrà 6 feature maps.

Per exemple, considerant una imatge de 32×32 píxels, lliscant una finestra de 5×5 píxels per la imatge d'aprenentatge/entrada, amb una amplada de pas d'1 donarà lloc a una feature map de 28×28 valors de sortida. 32 × 32 ==> 28 × 28 (32–5 + 1 × 32–5 + 1), o 784 activations diferents per imatge.

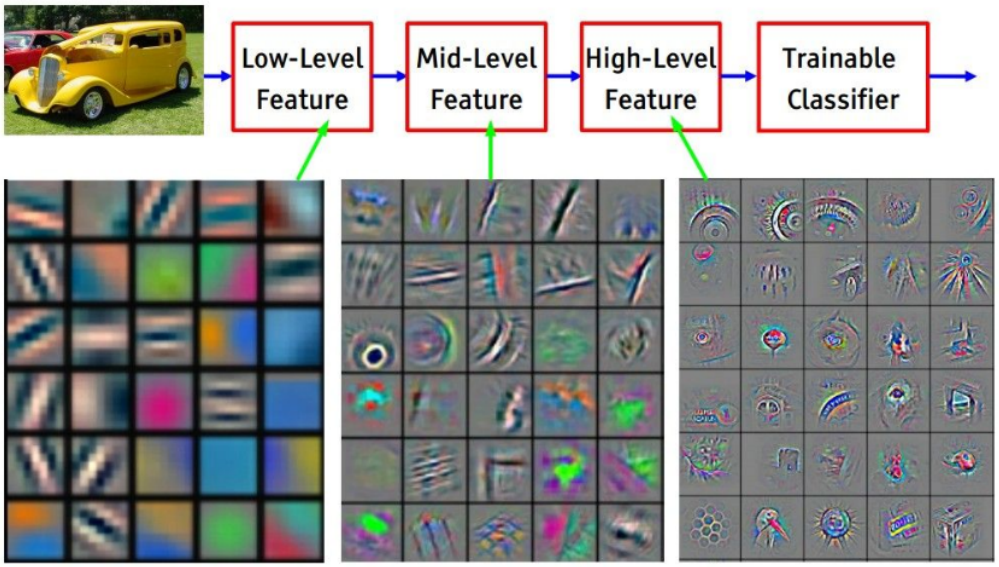


Figura 5: Filtros convolucionales obtenidos tras en entrenamiento de una red [8].

La mida del feature map (o també convolved feature) està controlada per tres paràmetres que hem de definir abans de realitzar el pas de convolució:

- **Size:** La de la feature map.

- **Depth**: La profunditat, correspon al nombre de filtres que fem servir per a l'operació de convolució.

  A la xarxa convolucional que es mostra en la següent imatge, es realitza una convolució de la imatge original del vaixell mitjançant tres filtres diferents, produint així tres feature maps diferents, tal com es mostra. Podeu pensar en aquests tres mapes de característiques com a matrius 2D apilades, per tant, la "profunditat" del mapa de característiques serà de tres.

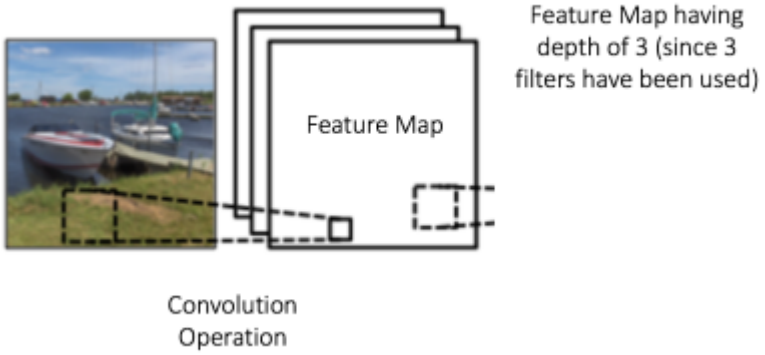


Figura 6: Paràmetre depth de les Feature Maps [8].

- **Stride**: El pas, és el nombre de píxels què llisca la matriu de filtre sobre la matriu d'entrada. Quan el pas és 1, mou els filtres un píxel a la vegada. Quan el pas és de 2, els filtres salten 2 píxels alhora mentre els llisquen. Tenir un pas més gran produirà feature maps més petites.

- **Zero-padding:** De vegades, és convenient omplir la matriu d'entrada amb zeros al voltant de la vora, de manera que es puguin aplicar el filtre als elements de la vora de la matriu d'input. Una bona característica del zero-padding és que ens permet controlar la mida dels feature maps. L'addició de zero-padding també s'anomena convolució ampla (wide convolution), i no utilitzar zero-padding és una convolució estreta (narrow convolution).

2.4.4 Representació del contingut

La xarxa neuronal convolucional desenvolupa representacions de la imatge durant tota la "jerarquía" de processament. Quan més ens endinsem en la xarxa, més s'accentuen les característiques "estructurals" (structural features) i menys el detall dels píxel.

Per elaborar aquestes representacions, es reconstrueixen les imatges utilitzant el mapeig de característiques (feature maps) de la mateixa capa. Des de la capa més baixa tindrem exactament la mateixa imatge de contingut, a mida que anem pujant en les capes, les imatges reconstruides contindran el contingut "d'alt nivell" (High-Level content), per tant en les capes superiors direm que tenim la representació del contingut (content representation).

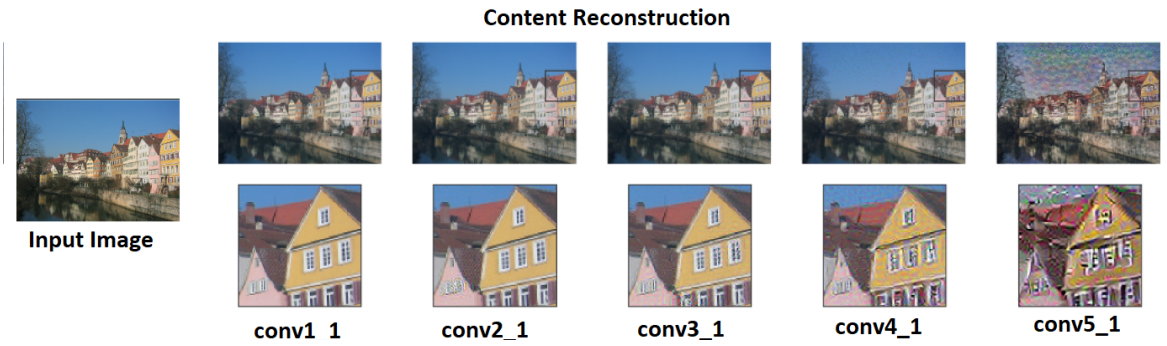


Figura 7: Reconstrucció del contingut segons la profunditat de la xarxa [1].

2.4.5 Representació de l'estil

Per tal d'extreure la representació de l'estil, s'ha de construir un 'feature space' a sobre de les respostes dels filtres en cada capa de la xarxa. Consisteix en les correlacions entre les diferents respostes dels filtres sobre l'extensió espacial de les feature maps.

Aquesta correlació de filtres entre les diferents capes captura la informació de la textura de la imatge que s'ha passat com a input. Això crea noves imatges que coincideixen amb l'estil artístic d'una imatge donada a escala creixent alhora que descarta la informació de la disposició global.

Aquesta representació multiescala s'anomena representació d'estil (style representation). A mesura que ens aprofundim en la xarxa, podem veure que la disposició global o les característiques estructurals es van descartant.

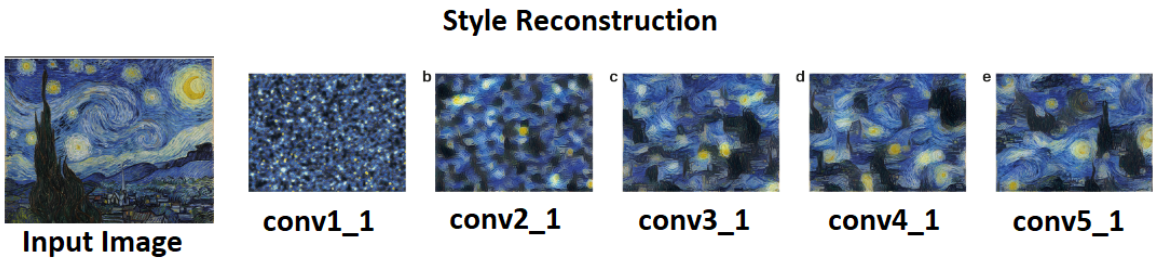


Figura 8: Reconstrucció de l'estil segons la profunditat de la xarxa [1].

2.5 Funcions de pèrdua

Una funció de pèrdua (Loss function), és una funció que avalua la desviació entre les prediccions realitzades per la xarxa neuronal i els valors reals de les observacions utilitzades durant l'aprenentatge. Com més baix sigui el resultat d'aquesta funció, més eficient és la xarxa neuronal. La seva minimització es fa ajustant els diferents pesos de la xarxa neuronal durant el procés d'entrenament.

La funció de pèrdua per al problema de Style transfer es calcula a partir de dues funcions independents, la funció de pèrdua de contingut (Content loss) i la d'estil (Style loss), calculades a partir de la content image i style image, respectivament. Les dues funcions de pèrdua es combinen per calcular la funció de pèrdua total, que és la que utilitza l'algoritme per a l'optimització.

2.5.1 Funció de pèrdua de contingut

La funció de pèrdua de contingut representa una versió ponderada de la distància del contingut per a cada capa individual de la xarxa. La funció agafa les feature maps de la capa L dins l'input X processat en la xarxa i retorna la la distància ponderada del contingut (weighted content distance). Per tant, la distància és $||F_{XL} - F_{CL}||^2$, la mitja al quadrat de l'error entre dos sets de feature maps.

La content image i la output image es passen al nostre model i s'extreuen les sortides de les capes intermèdies, mitjançant l'extractor (una classe ja definida, que extreu i emmagatzema les features a cada capa) per obtindre la feature representation. Aleshores es calcula la distància euclidiana entre la representació intermèdia de la imatge de contingut i la imatge base d'entrada.

Per tant, la pèrdua de contingut per a una capa l es defineix com:

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(F_{ij}^{l} - P_{ij}^{l}\right)^2$$

Calcula l'error al quadrat entre la feature representation de la content image (p) i la imatge input (x) de la capa (l). Aquí, n^l , $n^l w$, $n^l c$ són l'alçada, l'amplada i els canals de la capa l. Per calcular la pèrdua de contingut, la representació intermèdia de dimensions n^l x $n^l w$ x $n^l c$ es desgloça en vectors de dimensions $n^l c$ x n^l *$n^l w$. Desgloçar la feature representation no és un pas obligatori, però és una bona pràctica. El següent diagrama ens ajudarà a visualitzar aquesta transformació.

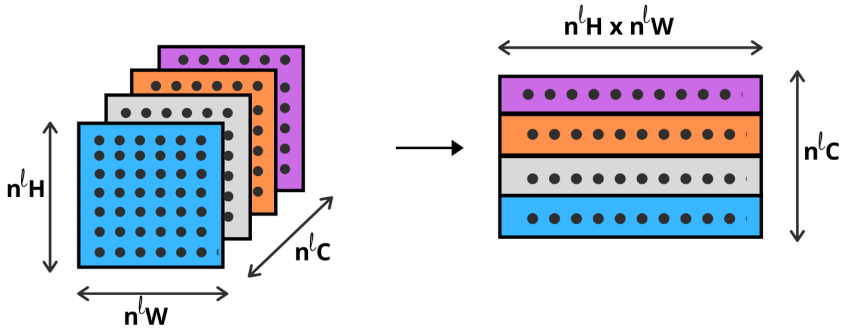


Figura 9: Vectorització de la feature representation del contingut [28].

F^l_i i P^l_i són els vectors de dimensions $n^l c x n^l$ *$n^l w$ que representen la representació intermèdia de la imatge input i la content image.

2.5.2 Funció de pèrdua de l'estil

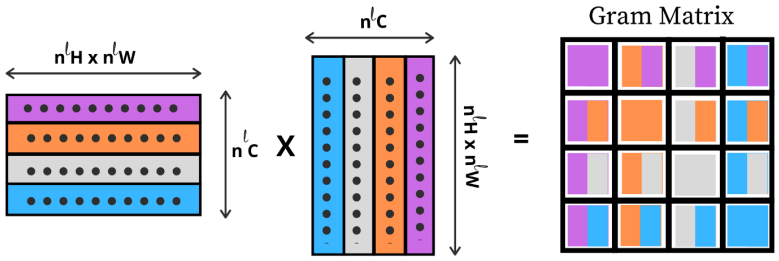
Per calcular la funció de pèrdua d'estil (Style loss), s'ha de definir un espai de característiques (feature space) sobre cada capa de la xarxa que representa la correlació entre les diferents respostes d'un filtre concret.

2.5.2.1 Matriu de Gram

La matriu de Gram (Gram matrix) calcula aquestes correlacions de les features capa a capa. Per calcular la Gram matrix prenem el producte escalar de la representació intermèdia desglossada i la seva transposada, les dimensions d'aquesta són n^l_c x n^l_c, on n^l_c és el nombre de canals en la representació intermèdia de la capa l.

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk}$$

A l'equació anterior, G^l_i és el producte intern entre les feature maps vectoritzades i i j a la capa l. L'equació vectoritzada d'una matriu de Gram es mostra a la figura següent, on G és la matriu de grams de la representació intermèdia A.



$$G_{gram} = A_{unrolled} A^T_{unrolled}$$

Figura 10: Matriu de Gram [28].

2.5.2.2 Funció de pèrdua d'estil total

Per tant, el Style loss de la capa l és l'error al quadrat entre les matrius de Gram de la representació intermèdia de la imatge d'estil i la imatge input:

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G^l_{ij} - A^l_{ij}\right)^2$$

On E és la pèrdua d'estil per a la capa l, N i M són el nombre de canals i l'alçada per l'amplada en la feature representation de la capa l, respectivament. G^l_i i A^l_i són les matrius grams de la representació intermèdia de la imatge d'estil (a) i la imatge input (x) respectivament.

Tenim llavors que la funció de pèrdua d'estil total segueix la següent fòrmula:

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l$$

On w^l és el factor de pes de la contribució de cada capa a la pèrdua total d'estil. S'aconsella que la matriu de Gram sigui normalitzada, dividint cada element pel nombre total d'elements en la matriu.

Aquesta normalització serveix per corregir el fet que les matrius amb una gran dimensió N donen valors més grans a la matriu de Gram. Aquests valors més grans fan que les primeres capes (abans de l'agrupament de les capes) tinguin un impacte més gran durant el descens del gradient.

Les features de l'estil solen estar a les capes més profundes de la xarxa, per la qual cosa aquest pas de normalització és crucial. La sortida de la funció Style loss ha de ser igual que la de la funció Content loss, per tal de poder calcular la distància ponderada, això és quelcom a tenir en compte a la hora de dissenyar les funcions.

2.5.3 Funció de pèrdua total

Un cop calculada la pèrdua de contingut i d'estil de la imatge que hem passat com a input, és necessari correlacionar-les a través d'una funció per obtindre un sol valor:

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$

A l'equació anterior, $L_{o\ a}$ és la pèrdua total, $L\,c_{o\ \ e}$ és la pèrdua de contingut de totes les capes intermèdies i $L_{\gamma\ e}$ és la pèrdua d'estil de totes les capes intermèdies.

Aquí, α i β són els coeficients de ponderació del contingut i la pèrdua d'estil, respectivament, p, a i x són la content image, la style image i la imatge generada o la imatge input (recordem que prèviament s'ha definit un placeholder anomenat output image que contindrà la sortida de la xarxa, aquesta mateixa output image és la imatge passada com el input de la xarxa, mencionat en tota la definició de les funcions de pèrdua).

Realitzant un descens de gradient sobre la funció de pèrdua i en lloc dels paràmetres del model, s'actualitzen els valors dels píxel de la imatge input x, es minimitza la pèrdua. Això fà que la imatge input sigui semblant al contingut i al estil de les altres imatges. Es pot emfatitzar l'estil o la pèrdua de contingut alterant els valors de α i β.

S’executarà els mètodes “backward” amb cada funció de pèrdua per calcular dinàmicament els seus gradients. L'optimitzador requereix una funció de tancament (**closure**), que revalúa el mòdul i retorna la pèrdua total.

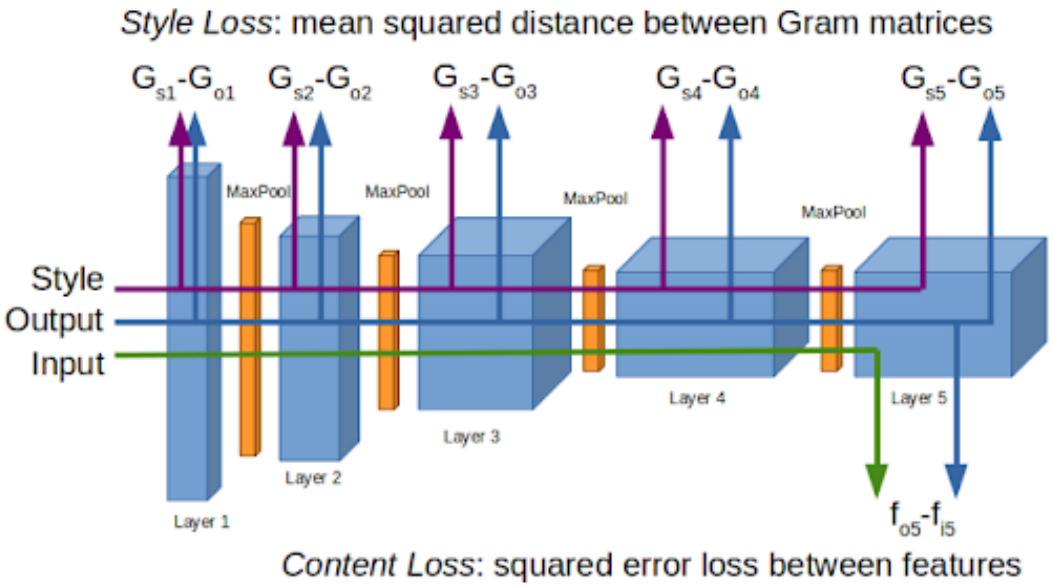


Figura 11: Representació del càlcul de la Total Loss a través de la xarxa CNN [28].

2.6 Processament de les dades

Per fer una imatge compatible amb el model i poder treballar amb ella, la imatge ha de ser preprocessada. Ja sigui mitjaçant els mètodes que ens faciliten les llibreries o implementant els nostres propi mètodes, s’han de realitzar alguns dels preprocessaments bàsics: canviar la mida de les imatges a un màxim 512 x 512 píxel i convertir la imatge al format tensor, una clase matriu multi-dimensional, amb la que es poden guardar les imatges, solen estar normalitzades a un rang de 0 a 1.

També s’ha de normalitzar el tensor amb una mitjana (0,485, 0,456, 0,406) i una desviació estàndard de (0,229, 0,224, 0,225), ja que per els models pre-entrenats que farem servir és necessari (VGG16 i VGG19, els més habituals i òptims per fer Style transfer).

No obstant, un cop acabat el processament de les imatges, no ens hem d’oblidar de tornar a convertir-les a la seva escala original, fent el processament de les dades invers amb l’output de la xarxa.

2.7 Entrenament d’una xarxa neuronal

Les xarxes neuronals (o models) són una tecnologia que permet realitzar operacions molt complexes, molt més que la que pot aconseguir la programació i algorismica regular. No obstant, per programar

aquesta intel·ligència artificial, és condició sotmetre-la a un procés per a que, a partir de datasets de parells de dades entrada-sortida (dades test i els seus resultats), la pròpia xarxa sigui capaç de decidir quines ponderacions i biaixos defineix en els seus nodes a cada capa, a fi d'establir una lògica de funcionament i càlcul per arribar als resultats esperats. Aquest procés llarg que requereix una gran quantitat de dades de test i iteracions al model, es coneix com a entrenament de la xarxa neuronal.

Tota xarxa neuronal que sigui funcional, pot ser entrenada un nombre infinit de iteracions. Tant com si s'està creant des de cero i s'han d'establir els primers pessos de manera aleatoria, com si ja ha sigut entrenada prèviament (se'n diu llavors, que és un model pre-entrenat). Quantes més iteracions es facin i més dades es tinguin, el model serà més eficient.

El temps que pot requerir un entrenament és completament arbitrari però definit en part pels hyperparameters (batch_size, optimizer, epoch, image_size, etc.) i per els recursos que pugui oferir la teva computadora, com per exemple, la memòria ram i nombre de nuclis disponibles en la teva GPU (si s'està usant per a fer funcionar el model).

El conjunt de dades que alimenta un model durant el seu entrenament es coneix com a dataset (conjunt de dades en anglès). Podem distingir tres tipus de datasets, cada un per a una tasca concreta de l'entrenament d'una xarxa neuronal:

- **Training set:** és el conjunt que s'utilitza per a que la xarxa aprengui patrons sobre les dades.

- **Validation set:** Aquest conjunt es fa servir per buscar els millors hiper-paràmetres d'una xarxa neuronal (poden ser: learning rate, epochs, optimizer, batch_size, número de capes, nombre de nodes, funcions d'activació, etc.). S'entrena a la xarxa neuronal amb uns hiper-paràmetres concrets, s'observa el resultat del conjunt de validació, es canvien els valors de nou i van comparant resultats fins trobar els hiper-paràmetres que siguin bons resolent el problema en qüestió. Cal que aquest conjunt tingui dades diferents a les dels altres dos sets, això és important per poder probar la xarxa neuronal amb dades que no ha vist mai.

- **Testing set:** Aquest conjunt es fa servir al final de l'entrenament per comprovar que tan bona és la xarxa neuronal i trobar els millors hiper-paràmetres. Si no es tenen gaires dades, aquest conjunt es pot ignorar i fer servir només el conjunt d'entrenament i el conjunt de validació.

3. Implementació

A continuació es parlarà sobre la part pràctica d'aquest treball, en aquesta secció veurem tot el codi, recursos i algorismes utilitzats i la seva funcionalitat.

Tota l'execució de codi s'ha realitzat en un entorn virtual python, la computadora amb la que s'ha fet reuneix les següents característiques:

- Sistema operatiu: Ubuntu 20.04.03 LTS 64 bits, Linux.
- Processador: Intel Core i7-8550U CPU @ 1,80GHz x 8.
- Targeta gràfica: NVIDIA GeForce GTX 1050 Mobile 4GB.
- Capacitat del disc: 500 GB.

3.1 Programari utilitzat

A continuació hi ha descrit tot el programari que s'ha utilitzat per realitzar aquest treball, llistant i describint totes les eines instalades. El procediment per instalar-ho tot, es troba en l'annex d'aquest treball.

Tot el codi ha estat provat i executat en el sistema operatiu Ubuntu 20.04 de Linux, no s'ha provat en altres entorns, però totes les llibreries utilitzades, a excepció de la magenta, es troben també en Windows. Al ser magenta una llibreria oberta, alternativament es podria programar un codi propi a partir d'aquesta.

3.1.1. Python 3

El llenguatge amb el que està escrit tot el codi del treball és Python, concretament la **versió 3.10.4**. Python és un llenguatge de programació interpretat que segueix la filosofía de mantenir una sintaxis llegible i és molt utilitzat amb el Deep Learning i les Xarxes Neuronals.

3.1.2. Jupyter Notebook[1]

Jupyter Notebook és un entorn informàtic interactiu bassat en web per crear documents notebook, que combinen funcionalitats de software de processament de textos amb el shell i kernel del llenguatge de programació, en aquest cas, Python 3.

Fem servir el Jupyter Notebook per a les implementacions dels diferents algoritmes utilitzats per fer Style transfer, així es facilita i simplifica l'ús d'aquests.

[1] https://jupyter.org/

3.1.3. CUDA ToolKit[2]

CUDA (acrònim de Compute Unified Device Architecture, en català, Arquitectura de còmput de dispositius unificats) és una plataforma de computació paral·lela i model d'Interfície de programació d'aplicacions (API) creada per Nvidia[3] per permetre a desenvolupadors i enginyers de programari accelerar l'execució dels seus codis fent servir Unitats de procesament gràfic (GPU). Un requisit bàsic per el treball amb xarxes neuronals de qualsevol tipus.

La plataforma CUDA és una capa de software que dóna accés directe al conjunt virtual d'instruccions de la GPU i als seus elements de comput paral·lel a efectes d'executar nuclis de cómput (CUDA Kernels). CUDA Toolkit són unes llibreries proporcionades per Nvidia que utilitzen aquesta interfície i serveixen per crear i desenvolupar aplicacions que s'executen en la GPU, tal com és habitual en aplicacions de Deep Learning, com la tècnica de Style Transfer que farem servir en aquest treball.

3.1.4. cuDNN[4]

Nvidia CUDA Deep Neural Network (cuDNN) és una biblioteca de primitives amb acceleració de GPU per l'ús específic de xarxes neuronals.

3.1.5. TensorFlow[5]

Tensorflow és una llibreria de codi obert per el desenvoulpament de Deep Learning a través de múltiples funcionalitats i desenvolupat per Google. Permet crear i entrenar xarxes neuronals per detectar i desxifrar patrons i correlacions anàlogues al aprenentatge i raonament humà.

Es una de les llibreries prinicpals implementacions CNN en Python, juntament amb les llibreries de Pytocrh i Keras (que desenvolupen una tasca similar). Tensorflow és la llibreria base en totes les implementacions que es presenten en aquest treball, la farem servir tant per els tres algoritmes ja implementats com per a la xarxa que entrenarem.

3.1.6. Magenta[6]

Magenta és un projecte de recerca de codi obert que explora l'aprenentatge automàtic en el procés de creació d'art i música fent ús de Tensorflow. Desenvolupa algorismes de Deep Learning i Reinforcement per generar cançons, imatges, dibuixos i altres materials artístics. També permet la creació i entrenament de models. Magenta va ser iniciat per alguns investigadors i enginyers de l'equip de Google Brain, però molts altres han contribuït significativament al projecte.

---

[2] https://developer.nvidia.com/cuda-toolkit
[3] https://www.nvidia.com/es-es/
[4] https://developer.nvidia.com/cudnn
[5] https://www.tensorflow.org/?hl=es-419
[6] https://github.com/magenta

Es farà servir l'entorn Magenta i les seves eines per a entrenar un model de xarxa neuronal propi i treballar amb ell.

3.2 Model d'arquitectura de la xarxa neuronal

Tota xarxa neuronal utilitza el que es coneix com un model d'arquitectura, és a dir, quina definició de l'estructura segueix la xarxa (el nombre de capes, quants nodes, etc). S'estableix des d'un primer moment i no és pot modificar en cap moment.

Si per exemple, es té una xarxa CNN entrenada de 20 capes i es volgues fer Style transfer ja sigui amb més capes per millorar el resultat, o bé amb menys capes per reduir el temps d'execució. Caldria tornar a crear el model des de cero i entrenar-lo (o agafar un model pre-entrenat amb les característiques dessitjades). Per tant, és molt important definir correctament, i des de l'inici, quina arquitectura és la idònia per el problema a resoldre.

Per el problema del Style transfer necessitem definir un model de xarxa clasificadora (per la primera part de la transferència), seguirem el model d'arquitectura proposat per Gatys [1].

3.2.1. VGG-16 i VGG-19

Els dos models més habituals i que segueixen l'arquitectura ja mencionada són els models pre-entrenats VGG-16 i VGG-19, creats per Karen Simonyan i Andrew Zisserman l'any 2015, tal com es mostra en l'article "Very Deep Convolutional Networks for Large-Scale Image Recognition" [3], on podem veure en detall tota l'estructura i configuracions possibles d'aquests models de classificació. El seu nom indica el nombre de capes ocultes (profunditat o depth en anglès) que conté la xarxa (capes de convolució i de subsampling, incloent també les capes 'perceptron', utilitzades quan aquestes arquitectures s'utilitzen per a classificació d'imatges), 16 capes en la primera i 19 capes en la segona.

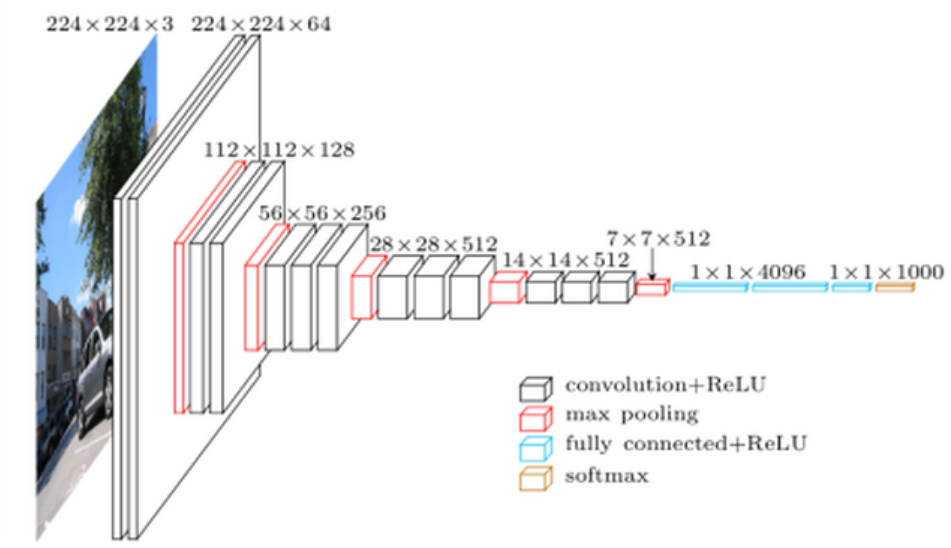


Figura 12: VGG16: Una Red Convolucional Preentrenada [23].

Per què triar un model pre-entrenat VGG? Característiques generals i que poden ser moitu per escollir-lo, són les següents:

- Té una arquitectura fàcil de comprendre i d'implementar.
- La xarxa (model i pesos entrenats) està disponible en les llibreries Python més comunes, com són Tensorflow, Keras i Pytorch.
- Posterior al entrenament, aconsegueix un excel·lent resultat amb les imatges del dataset utilitzat.
- Conté relativament poques capes convolucionals, en el cas de VGG-16: 13 capes convolucionals i 3 denses (sumant un total de 16 capes, com veiem en el nom).

Per poder fer funciona l'execució de la xarxa, VGG conté una enorme quantitat de paràmetres que es poden modificar (alguns configurables per l'usuari), alguns d'ells són:

- **weights:** indica els pesos que seran els utilitzats per inicialitzar el model (pes del contingut, pes de l'estil, pes de la variació total, etc.)
- **include_top:** indica si es carrega la xarxa completament (extracció de característiques i etapa de decisió) o només l'etapa d'extracció de característiques.
- **input_shape:** el tamany en píxel de les imatges a processar.

Els models creats utilitzant la llibreria Tensorflow, com és el cas de les xarxes pre-entrenades VGG, contenen el mètode summary per extreure un resum complet de l'arquitectura de la xarxa. Incloent les capes què la componen, la mida inicial i final (abans i després de la convolució) i el nombre de paràmetres entrenables, com es pot veure a la següent impressió:

```
=================================================================================
#    Layer Name       Layer Input Shape        Layer Output Shape       Parameters
=================================================================================
0    input_5          (None, 224, 224, 3)      (None, 224, 224, 3)             0
1    block1_conv1     (None, 224, 224, 3)      (None, 224, 224, 64)         1792
2    block1_conv2     (None, 224, 224, 64)     (None, 224, 224, 64)        36928
3    block1_pool      (None, 224, 224, 64)     (None, 112, 112, 64)            0
4    block2_conv1     (None, 112, 112, 64)     (None, 112, 112, 128)       73856
5    block2_conv2     (None, 112, 112, 128)    (None, 112, 112, 128)      147584
6    block2_pool      (None, 112, 112, 128)    (None, 56, 56, 128)             0
7    block3_conv1     (None, 56, 56, 128)      (None, 56, 56, 256)        295168
8    block3_conv2     (None, 56, 56, 256)      (None, 56, 56, 256)        590080
9    block3_conv3     (None, 56, 56, 256)      (None, 56, 56, 256)        590080
10   block3_pool      (None, 56, 56, 256)      (None, 28, 28, 256)             0
11   block4_conv1     (None, 28, 28, 256)      (None, 28, 28, 512)       1180160
12   block4_conv2     (None, 28, 28, 512)      (None, 28, 28, 512)       2359808
13   block4_conv3     (None, 28, 28, 512)      (None, 28, 28, 512)       2359808
14   block4_pool      (None, 28, 28, 512)      (None, 14, 14, 512)             0
15   block5_conv1     (None, 14, 14, 512)      (None, 14, 14, 512)       2359808
```

```
16   block5_conv2     (None, 14, 14, 512)      (None, 14, 14, 512)          2359808
17   block5_conv3     (None, 14, 14, 512)      (None, 14, 14, 512)          2359808
18   block5_pool      (None, 14, 14, 512)      (None, 7, 7, 512)                  0
19   flatten          (None, 7, 7, 512)        (None, 25088)                      0
20   fc1              (None, 25088)            (None, 4096)               102764544
21   fc2              (None, 4096)             (None, 4096)                16781312
22   predictions      (None, 4096)             (None, 1000)                 4097000
_________________________________________________________________________________
Total Parameters :  138357544
Total Trainable Parameters :  138357544
Total No-Trainable Parameters :  0
```

Figura 13: Impressió feta a partir de l'execució d'una VGG16, amb un input imatge 224x224 tensor [23].

Per poder fer servir aquest model, és necessari pasar-hi un input adequadament processat. imatges del tamany indicat en el input_shape, convertides a tensors de dimensionalitat 4. El output serà en el mateix format.

La distinció entre el model VGG-16 i VGG-19 es troba en la profunditat (depth) en capes que té la xarxa. Una model VGG-19, al ser més profund suposa un cost computacional més alt que el del VGG-16, però alhora és capaç d'identificar millor característiques més complexes en la imatge. Amb el que es tradueix en un millor acabat en els resultats.

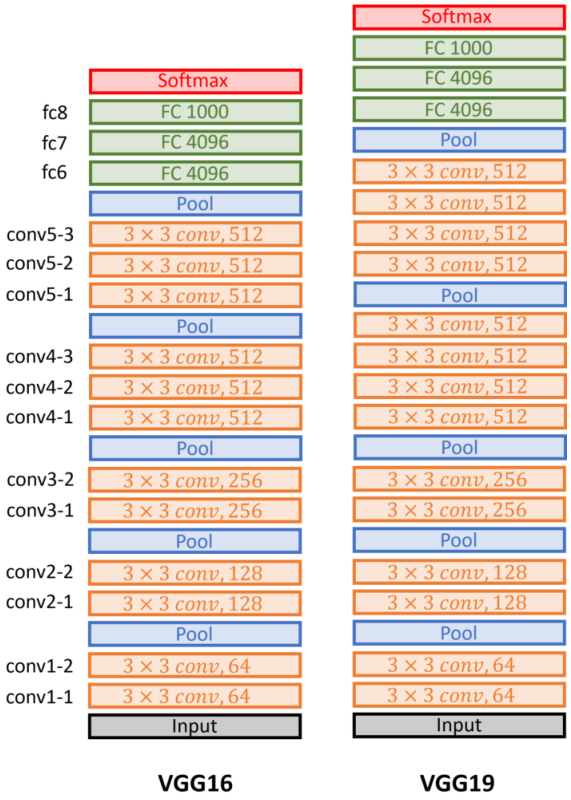


Figura 14: Comparativa de capes VGG-16 i VGG-19 [27].

3.3 Algorismes implementats

Per a poder desenvolupar l'estat de l'art de la tecnologia de transferència d'estil (Style Transfer), farem ús de tres implementacions actuals d'aquesta tecnologia, els he denominat:

Algorisme A: El primer algorisme implementat és una aplicació simple de transferència d'estil proposada por Gatys et al. [1]

Algorisme B: La implementació del Fast style transfer que proposa Ghiasi et al. [2]

Algorisme C: La implementació de Artistic style transfer que proposa Prisma Labs, desenvolupadors de la app mòbil Prisma que es centra en el processament de imatge.

3.3.1 Algorisme A

Com a algorisme de transferència d'estil, rep dos imatges, una imatge de contingut i una d'estil i crea una tercera amb la combinació de les dues, agafant le contingut d'una i l'estil de l'altre. Definint l'input a una resolució màxima de 512x512 píxels, agafa la imatges de contingut i de estil i les escala per poder-les processar mantenint el ratio original i donant un output amb les mides escalades del original.

Aquesta implementació fa ús del model d'arquitectura VGG-19, recordem, una xarxa de clasificació d'imatges pre-entrenada, que està formada per 19 capes de convolució.

A la hora de realitzar l'estil de transferència, és habitual que el resultat tingui molts artefactes d'alta freqüència en la imatge, això se'n denomina 'pèrdua de variació total. Aquesta implementació ho corregeix fent la suma dels quadrats dels valors passats per un passalt, similar al que faria un detector de contorns sobel. Aquesta instrucció, però, ja sol venir implementada en la majoria de llibreries i implementacions.

3.3.2 Algorisme B

La segona implementació està basada en el codi del model CNN artístic Magenta on s'explora la definició d'un algoritme més ràpid per la estilització d'una imatge a fi d'aconseguir un càlcul en "temps real" o d'una manera aproximada. Ideal per l'ús en videos o dispositius mòbils.

Mitjançant de Tensorflow Hub importa el mòdul d'estilització d'imatge artística arbitrària de magenta. Això deriva el processament del model en remot, reduïnt significativament el cost i temps de l'algorisme. Fa ús d'un model d'arquitectura VGG-16, el qual redueix el temps d'execució respecte a l'alternativa VGG-19.

3.3.3 Algorisme C

Finalment, aquesta és una implementació que ha creat l'equip Prisma Labs, creadors de l'App mòbil Primsa de processament de imatges basada en l'arquitectura proposada en l'article de Gatys et al. [1]. Fa ús principalment de la llibreria Keras, que és una biblioteca de xarxes neuronals artificials de codi obert que es poden executar damunt de TensorFlow (entre altres llibreries). Fa ús del model Keras pre-entrenat VGG16, enlloc del VGG19 per a fer una execució més ràpida del càlcul. A fi d'optimitzar l'execució, utilitza l'algorisme quasi-Newton "L-BFGS", que redueix al mínim la quantitat de memòria utilitzada en el càlcul d'una funció concreta, és molt comuna en Machine Learning.

Aquesta implementació realitza un total de 10 iteracions a la xarxa per obtenir el resultat, reduint així el temps d'execucció i el cost computacional de l'algorisme. Cada iteració sol trigar més o menys 25 segons, fent que prengui 4 minuts, aproximadament, l'estilització d'una sola imatge.

3.4 Datasets

A la hora de realitzar l'entrenament d'un model de xarxa neuronal convolucional (CNN) és requisit establir quins datasets (conjunts de dades) es faràn servir en el procés. Els datasets que he utilitzat per entrenar el meu model propi són:

3.4.1 Describable Textures Dataset[7]

El dataset Describable Textures Dataset (DTD), és una col·lecció en d'imatges de textura, dades que es posen a disposició de la comunitat de visió per computador amb finalitats de recerca. Aquesta base de dades consta de 5640 imatges, organitzades segons una llista de 47 termes inspirats en la percepció humana.

Hi ha 120 imatges per a cada categoria. Les mides de la imatge oscil·len entre 300x300 i 640x640 píxel, i les imatges contenen almenys el 90% de la superfície que representa l'atribut de la categoria. Per a cada imatge es proporciona un atribut clau (categoria principal) i una llista d'atributs complementaris.

Per al meu entrenament, he reduït el tamany de cada categoria del dataset a 30, donant un total de 1410 imatges, sumant les 47 categories. D'aquesta manera es redueix el tamany del directori del projecte.

3.4.2 Painter by Numbers[8]

Aquest repositori és un recopilatori de les dades reunides en el concurs "Painter by Numbers" promogut per Kaggle, una competició que proposaba desenvolupar una xarxa neuronal que fos capaç de discernir si un quadre concret era l'autèntic o una imitació.

[7] https://www.robots.ox.ac.uk/~vgg/data/dtd/
[8] https://www.kaggle.com/c/painter-by-numbers

Aquest dataset és el conjunt de dades que es va utilitzar per crear i validar els models predictius participants d'aquella competició, realitzada l'any 2016. El seu training set conté un total de 103249 imatges, que hem escorçat a 2500 imatges en total, pel mateix motiu que l'anterior dataset.

3.4.3 ImageNet 2012[9]

ILSVRC 2012, també conegut com a ImageNet, és un conjunt de dades d'imatges organitzades segons la jerarquia de WordNet (una base de dades lèxica que agrupa paraules en anglès en conjunts semàntics). Cada concepte significatiu a WordNet, possiblement descrit per diverses paraules o frases de paraules, s'anomena 'synset'. Hi ha més de 100.000 synsets a WordNet, la majoria són substantius (més de 80.000). A ImageNet, es proporcionen una mitjana de 1.000 imatges per il·lustrar cada synset.

La versió de prova del dataset del 2012 conté un total de 100.000 imatges, amb algunes imatges adicionals que venen d'una petita actualització, l'any 2019[10].

També existeixen uns subsets de desenvolupament de ImageNet anomenats 'Development Kit', en concret el training set té el nom de 'Task 3'[11]. Aquest subset, juntament amb l'actualització de l'any 2019, són el conjunt de dades imageNet que farem servir, amb un total de 21.800 imatges (encara que l'ideal seria fer servir el dataset 2012 amb 100.000 imatges).

3.5 Entrenant un model propi

A més dels tres models ja pre-entrenats en els algorismes esmentats anteriorment, entrenarem la nostra pròpia xarxa neuronal. Fent ús de la llibreria magenta, la qual ens facilita les funcions per fer-ho, entrenarem un model VGG-16. Utilitzem els datasets DTD, PBN i imageNet per alimentar l'entrenament.

Per poder entrenar el nostre propi model amb magenta, necessitem el següent:

1. Els directori d'imatges per fer servir com a imatges d'estil. Painter by Number dataset (PBN) and Describable Textures Dataset (DTD). PBN training PBN testing DTD dataset
2. El dataset ImageNet.
3. Un model pre-entrenat: VGG model checkpoint[12].
4. Un model pre-entrenat: Inception-v3 model checkpoint[13].
5. Assegurar-se de tenir l'entorn magenta correctament instalat[14].

Un cop complerts aquests requisits ja espodrà entrenar un model per Style transfer propi.

---

[9] https://image-net.org/challenges/LSVRC/2012/
[10] https://image-net.org/data/ILSVRC/2010/patch_images.tar
[11] https://image-net.org/data/ILSVRC/2012/ILSVRC2012_img_train_t3.tar
[12] http://download.tensorflow.org/models/vgg_16_2016_08_28.tar.gz
[13] http://download.tensorflow.org/models/inception_v3_2016_08_28.tar.gz
[14] https://github.com/magenta/magenta/blob/main/README.md

3.5.1 Partint d'un model pre-entrenat

Per a realitzar l'entrenament del nostre propi model, amb l'objectiu de millorar les implementacions dels altres algorismes (A, B i C), partirem d'un model ja entrenat VGG-16 que ens facilita la pròpia llibreria magenta, juntament amb un altre model d'arquitectura Inception-V3.

El format amb el que es guarden els models i els seus checkpoints i que tenen els models VGG-16 i Inception-V3, és el format "checkpoint" de TensorFlow, amb l'extenció '.ckpt'.

3.5.2 Processament dels datasets

El primer pas és preparar el dataset d'imatges d'estil i crear-ne un fitxer en format TFRecord, facilitat per la llibreira TensorFlow.

Per entrenar i evaluar el model en diferents sets d'imatges d'estil, es necessita preparar diferents TFRecord per a cada un d'ells. Utilitzem les imatges de PBN i DTD training per crear el dataset d'entrenament i després crear-ne un altre amb PBN i DTD testing per fer l'evaluació.

Per a poder crear un training set que combini les coleccions de DTD i PBN, he ajuntat totes les imatges en un mateix directori, anomenat DTD_PBN_train, allà serà on s'executarà el constructor de fitxers TFRecord que facilita magenta.

La següent comanda serveix per buscar les imatges DTD i PBN i crear un fitxer TFRecord des de les imatges dins del directori DTD_PBN_train en linux:

```
$ cd /path/to/root
$ path=$(pwd)
$ STYLE_IMAGES_PATHS="$path"/datasets/DTD_PBN_train/*.jpg
$ RECORDIO_PATH="$path"/datasets/dtd_pbn_train.tfrecord

$ cd /path/to/magenta/models/image_stylization
$ python image_stylization_create_dataset.py \
        --style_files=$STYLE_IMAGES_PATHS \
        --output_file=$RECORDIO_PATH \
        --compute_gram_matrices=False \
        --logtostderr
```

Per crear els TFrecord del training dataset de ImageNet, s'utilitza aquesta altre comanda, llibreria externa a magenta, facilitada per Konstantinos [30]:

```
$ train=$path/datasets/imagenet/train/
$ output=$path/datasets/imagenet/trainTFRecord/

$ cd /path/to/imagenet_to_tfrecord_directory
$ python build_imagenet_data.py -train_directory $train -output_directory $output
```

3.5.3 Configuració dels paràmetres del model

Degut a les limitacions tècniques que té la meva computadora (només 4GB de ram en GPU), he establert els hiper-paràmetres del model a entrenar amb valors baixos, ja que l'execució pot ser molt llarga i, fins i tot, donar errors de memòria.

Per tant, les configuracions més rellevants que seguiran els hiper-paràmetres (facilitats en la llibreria magenta) de la nostra pròpia xarxa neuronal convolucional són:

- **train_step**: fa referència al nombre d'iteracions que es farà a la xarxa durant l'entrenament, el valor original és de 8.000.000, però l'he configurat a 80.000 passos per reduir el temps d'entrenament.

- **batch_size:** determina el nombre de cada iteració, i s'ha establert a 4, normalment és un valor més alt, però en el meu ordinador generaba errors de memòria dinàmica.

- **save_sumaries_secs:** és la freqüència, per guardar una còpia del resum (summary) de l'entrenament, el valor defineix els segons.

- **save_interval_secs:** interval de temps en segons per a guardar una còpia del model (checkpoint), útil per si durant el llarg procés de l'entrenament s'atura l'execució, no perdre el progrés.

- **image_size:** el model serà entrenat amb imatges input de 256 píxel, per tant serà necessari que el processament de les dades abans d'entrar a la xarxa siguin d'aquest tamany màxim.

Tota la resta d'hiper-paràmetres ja definits segueixen els valors per defecte (learnig_rate = 1e-5, total_variaton_weight = 1e4, content_weights = 1, style_weights = 0.5e-3).

3.5.4 Execució de l'entrenament

Un cop s'han preparat els datasets i configurat els hiper-pràmetres, es pot procedir, finalment, a entrenar el model. A continuació veiem les commandes a executar en el terminal per a realitzar l'entrenament:

```
$ logdir=$path/logdir/
$ IMAGENET_TFRECORD=$path/datasets/imagenet/trainTFRecord/
$ RECORDIO_PATH="$path"/datasets/dtd_pbn_train.tfrecord
$ VGG_16_CHECKPOINT=$path/vgg_16.ckpt
$ INCEPTION_V3_CHECKPOINT=$path/inception_v3.ckpt

$ python arbitrary_image_stylization_train.py \
        --batch_size=4 \
        --imagenet_data_dir=$IMAGENET_TFRECORD \
        --vgg_checkpoint=$VGG_16_CHECKPOINT \
        --inception_v3_checkpoint=$INCEPTION_V3_CHECKPOINT \
```

```
        --style_dataset_file=$RECORDIO_PATH \
        --train_dir="$logdir"/train_dir \
        --random_style_image_size=True \
        --augment_style_images=True \
        --center_crop=False \
        --logtostderr
```

Després vàries hores d'execució (relatiu al hardware del que es disposi), s'hauran realitzat totes les iteracions, prèviament definides, del nostre entrenament, guardant el resultat en el directori "logdir" passat per paràmetre, en el meu cas, a la carpeta 'logdir' dins del directori arrel d'aquest projecte.

El temps d'execució que ha trigat l'entrenament del model, en el meu cas, ha estat aproximadament d' unes X hores i 80.000 pasos.

3.5.5 Execució de l'entrenament

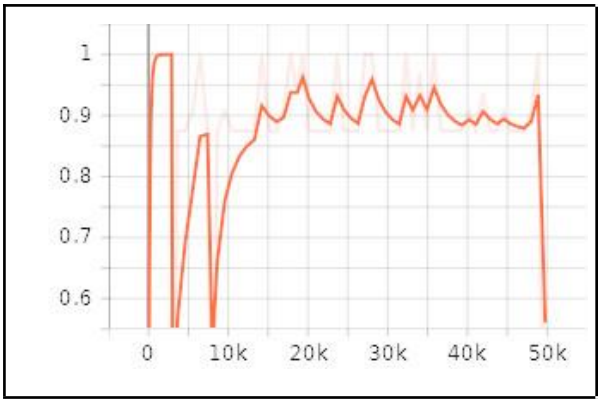
Finalment, tenit el model entrenat correctament, només queda executar-lo per tal de realitzar el càlcul del Style Transfer. La comanda per fer-ho és:

```
$ cd /path/to/root
$ path=$(pwd)
$ MODEL_CHECKPOINT=$path/logdir/train_dir/model.ckpt-50000
$ OUTPUT_DIR=$path/images/output/
$ STYLE_DIR=$path/images/styles/*.jpg
$ CONTENT_DIR=$path/images/*.jpg

$ cd /path/to/arbitrary_image_stylization

$ python arbitrary_image_stylization_with_weights.py \
    --checkpoint=$MODEL_CHECKPOINT \
    --output_dir=$OUTPUT_DIR \
    --style_images_paths=$STYLE_DIR \
    --content_images_paths=$CONTENT_DIR \
    --image_size=256 \
    --content_square_crop=False \
    --style_image_size=256 \
    --style_square_crop=False \
    --logtostderr
```

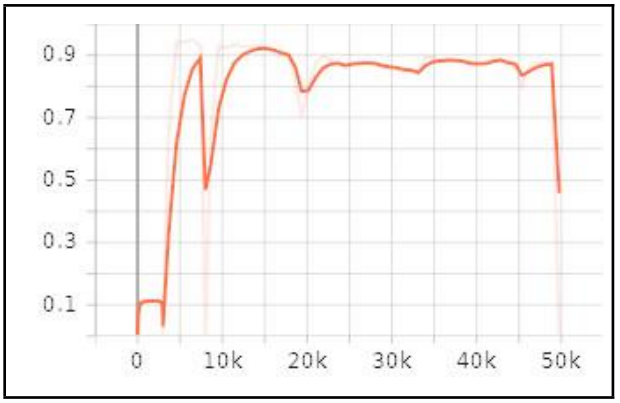
Durant o després de realitzar l'entrenament del model, TensorFlow ens facilita un visualitzador que mostra les dades i gràfiques d'aquest durant l'execució, per obrir el visualitzador TensorBoard, només cal executar les següents comandes:

```
$ logdir=$path/logdir/
$ tensorboard --logdir="$logdir"
```

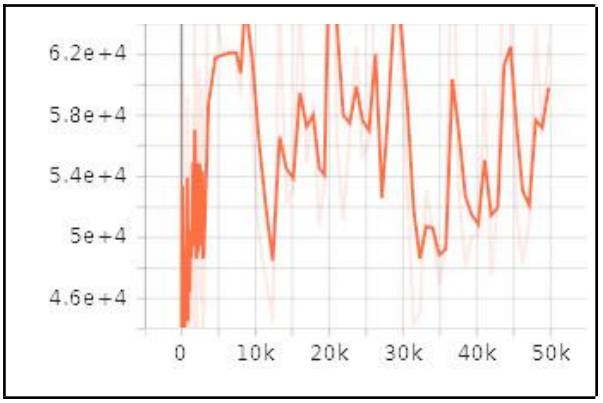
En el cas de l'entrenament del model que he realitzat per aquest treball, tenim les següents gràfiques:



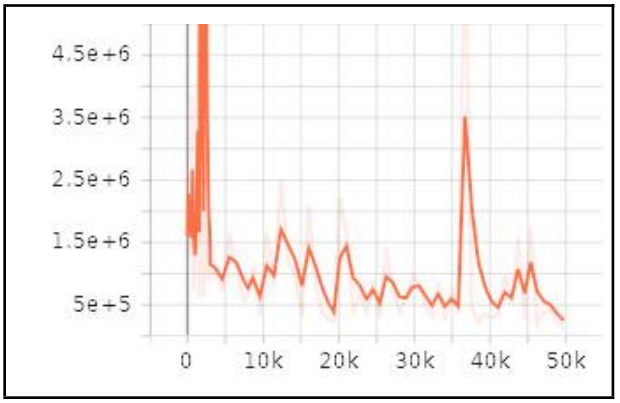
batch_processing



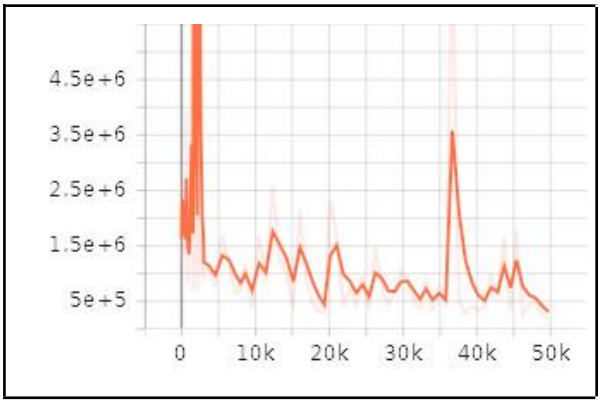
global_step



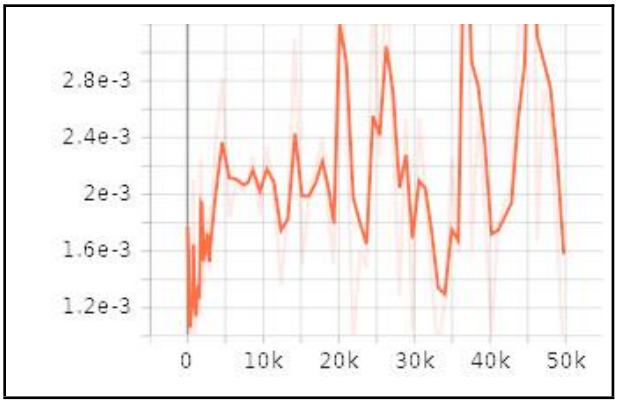
total_content_loss



total_style_loss



total_loss_1



total_variaton_loss

Com podem apreciar, l'entrenament del model és molt irregular, a cada pas, la pèrdua total del contingut fluctúa amb un rang molt alt. En el cas de la pèrdua total d'estil segueix més una progressió descendent (com s'espera que sigui), però en certes iteracions és dispara, tot i que amb menor rang que el contingut. el mateix amb la resta de valors.

Això és degut a que estem fent servir un batch_size molt baix, de 4, per les limitacions tècniques de la computadora que estic utilitzant. Al procesar poques dades a cada pas, és normal que cada poques iteracions es processin valors molt diferents als anteriors. Si definissim un batch_size més alt, i per tant es treballés amb més dades alhora, veuriem una tendència a la baixa.

També cal recordar que aquest entrenament només ha realitzat 50.000 pasos, quan de normal s'han de fer milions d'iteracions al model per poder estabilitzar els valors que hem vist en les gràfiques.

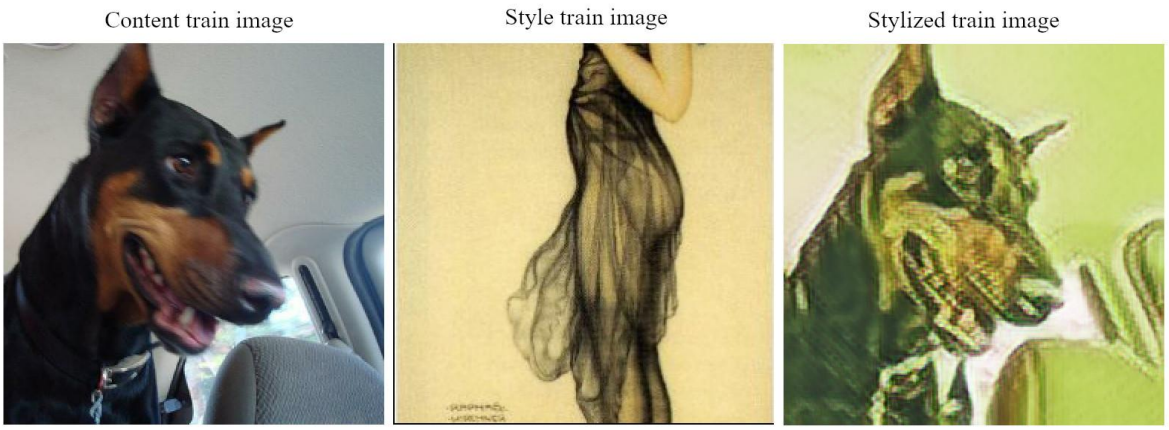


Figura 15: Model pròpi entrenat, estilització d'imatge durant el l'entrenament extreta a través de tensorboard.

4. Metodologia

Ja fa uns quants anys que existeix l'algoritme de la transferència d'estil (Style transfer), formalitzat per primer cop l'any 2015 per **Leon A. Gatys**, **Alexander S. Ecker** i **Matthias Bethge**, en el l'article "***A Neural Algorithm of Artistic Style***". Han sorgit moltes implementacions d'aquest algoritmes per mitjà de diferents eines i entorns. Després de varis anys, la implementació d'aquest algoritme ha anat reiterant-se i s'han trobat noves maneres d'optimitzar i millorar. Agafarem diferents implementacions per observar l'Estat de l'art actual del Style transfer. Ens abstindrem de fer servir implementacions que sumen altres eines (com l'ús de síntesis de textura, màscares, etc.) per obtindre variants de l'algoritme.

4.1 Experiment a realitzar

Definirem un seguit de imatges pròpies de contingut (no extretes d'internet, assegurant que no han servit en l'entrenament de cap de les xarxes utilitzades) amb característiques ben diverses, com poden ser: imatges de dibuix pla i realistes, retrats de persones en fotografia i en pintura, imatges amb molts i amb pocs elements, de diferents resolucions, etc. Després és calcularà els resultats del Style transfer amb un llistat de 20 imatges d'estil variades.

Amb aquest procediment s'espera sotmetre els models dels tres algoritmes ja implementats a una prova d'estrès, cercant els punts febles d'aquests i definint el seus motius.

Posteriorment entrenarem un model propi i després el sotmetrem a la mateixa prova. Compararem els resultats i n'extreurem conclusions de l'eficàcia d'aquesta tecnologia base.

4.2 Imatges utilitzades

Per a definir el conjunt d'imatges de contingut que utilitzarem en aquest experiment, he realitzat una recerca i tria de fotografies d'àmbit personal (meves o de coneguts) per garantir que són imatges completament alienes a les que s'han fet servir en l'entrenament de qualsevol dels models implementats. A continuació llistarem el nom de les imatges triades i el motiu d'haver estat triada per a realitzar aquesta prova d'estrés:

1. barcelona.jpg	Imatge amb molt contrast, molts detalls de diferents tamanys.
2. beautygirl.jpg	Imatge amb distribució de retrat d'una persona, amb poca varietat de color.
3. car.jpg	Imatge d'un cotxe, molts reflexes i poca varietat cromàtica.
4. castle.jpg	Imatge del pati interior d'un edifici, amb un gran contrast amb les ombres.
5. cave.jpg	Imatge d'un paisatge amb molts contrastos i detalls petits.

6. clemen.jpg	Imatge d'una gossa dormint, amb diferents detalls amb diferent definició.
7. clemen2.jpg	Imatge de la cara d'una gossa, amb elements al fons suavitzats.
8. excursionist.jpg	Imatge de grup de persones amb molts detalls petits.
9.fallesvalencia.jpg	Imatge d'una escultura de gran complexitat en forma i molts colors diferenciats, amb molts detalls molt petits.
10. house.jpg	Imatge d'una caseta en el camp, amb molts detalls petits i colors ben diferenciats.
11.lowresolution.jpg	Imatge d'un grup de persones al voltant d'una foguera amb molta poca definició.
12 .naturetrain.jpg	Imatge d'un paisatge natural, amb molts elements en pantalla molta textura i tonalitats dels colors presents.
13. painting.jpg	Imatge d'una pintura a pinzell d'una senyora, amb poca varietat de colors.
14. ski.jpg	Imatge d'un paisatge natural nevat amb una caseta al mig, molts detalls petits i molt color blanc.
15. skysunset.jpg	Imatge d'un núvol en el cel durant la posta de sol, vores poc diferenciades, pocs colors però ben diferenciats.
16. twofriends.jpg	Imatge d'un primer pla de dos persones amb el fons suavitzat.
17. vegetal.jpg	Imatge d'un dibuix fet a llapis sense colorejar, vores ben definides.
18. venecia.jpg	Imatge d'un canal d'aigua a venecia, l'aigua suposa molta superfície plana i amb reflexes, elements amb detalls molt petits.

Per triar el conjunt d'imatges d'estil he agafat algunes imatges de les implementacions de Style transfer que he trobat durant la fase d'investigació del treball i les he complementat amb imatges cercades per web amb característiques diferents. A continuació es llisten les imatges d'estil i el seu motiu de tria:

1. block.jpg	Imatge plana amb vores molt marcades i colors molt diferents.
2. city_paint.png	Imatge amb molts detalls petits i moltes tonalitats en els seus colors.
3. colorsplash.jpg	Imatge abstracte amb molts colors diferents.
4. crystalforest.jpg	Imatge poligonal amb colors freds.
5. forest.jpg	Imatge de pintura a pinzell amb colors càlids
6. gothic.jpg	Imatge abstracte amb parells de colors contraris.
7. japanese.jpg	Imatge de pintura a pinzell amb tinta, en blanc i negre.

8. Kandinsky.jpg	Imatge abstracte amb molts colors i formes complexes de diferents tamanys.
9. marilyn.jpg	Imatge de retrat amb colors contraris.
10. neonmecha.jpg	Imatge d'un dibuix digital amb molts elements i de bona calitat, amb un rang de colors limitat.
11. picasso.jpg	Imatge d'una pintura amb colors ben diferenciats.
12. picasso2.jpg	Imatge d'una pintura amb formes polígonals molt marcades, pintat amb poc color, amb una gran escala de grisos.
13. polygon.jpg	Imatge digital poligonal i amb colors càlids i freds contraris.
14. psychedelic.jpg	Imatge d'un dibuix digital abstracte que fa ús de colors molt variats amb contorns ben delimitats.
15. scream.jpg	Imatge d'una pintura a cera, amb formes molt fluides i poques rectes.
16.stained_glass_flower.jpg	Imatge d'un dibuix a llapis amb molts colors tonalitats diferents i vores ben definides.
17.starry_night.jpg	Imatge d'un quadre fet a pinzell amb formes en espiral sense gaire varietat cromàtica.
18. synthwave.jpg	Imatge 3D amb poca textura i moltes línies, pocs colors però variats.
19. van_gough.jpg	Imatge d'un quadre fet a pinzell amb traç molt fluid sense gaire varietat cromàtica.
20. wave.jpg	Imatge de puntura amb molta rugositat i molts detalls petits, amb pocs colors molt delimitats.

Les imatges de contingut i estil es troben dins del directori del projecte, a la carpeta amb el nom 'images'.

5. Resultats

En aquest apartat analitzarem tots els resultats obtinguts a partir de l'experiment proposat en el capítol anterior. Elaborant una bona síntesi de la informació rellevant a través d'aquests per definir un anàlisis.

5.1 Resultat de l'algorisme A

En aquesta implementació del Style transfer feiem servir un model d'arquitectura VGG-19, el qual, teoricament, requereix un cost computacional més alt per fer-lo funcionar a canvi de resultats més definits respecte els models VGG-16. Recordem però, que aquesta és la implementació bàsica que proposa TensorFlow sense cap tipus de millora.

Figura 16: Algorisme A, estilització d'imatges beautygirl amb gothic.

Realitzant l'experiment amb aquest algorisme, efectivament veiem que quan (posant èmfasi en el "quan") es realitza la transferència d'estil, es preserva l'estructura de la imatge de contingut transformant la informació amb patrons propis de la imatge d'estil. Conserva molt bé els petits detalls, les diferències de color i el contrast de la imatge original i transporta tots els elements artístics pertinents de la imatge de referència artística, millor que en les xarxes VGG-16 provades.

No obstant, també presenta errors greus en molts dels resultats: es pot percebre soroll molt evident en la textura i en el color de zones concretes de la imatge, apareixen petits artefactes d'aberració cromàtica (separació visual i explícita dels diferents canal de color en un píxel o regió). També és habitual que es puguin apreciar arbitràriament petites regions de la imatge on els píxel formen una barreja de colors, normalment marronosa i/o verdosa, amb una textura uniforme.

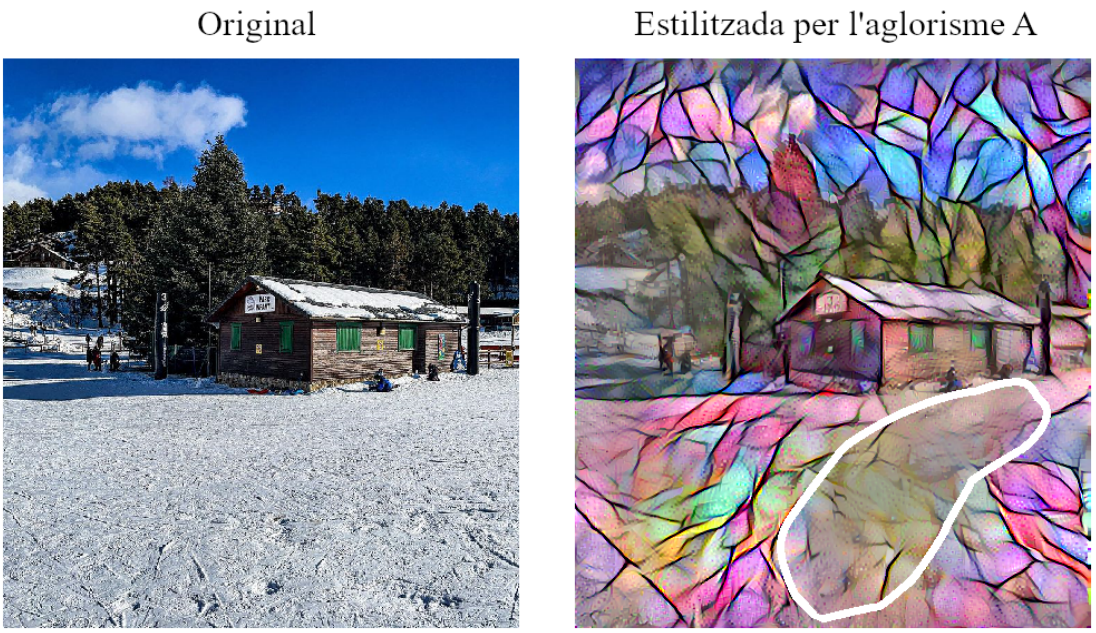


Figura 17: Algorisme A. Resultat d'estilització amb error (marcat en blanc) de les imatges ski i stainde_glass_flower.

No es pot arribar a discernir si el problema radica en imatges d'estil concretes, ja que aquests errors succeeixen en moltes de les imatges amb imatges d'estil diferents i en regions arbitràries de la imatge.

El que si podem observar, és que quan la xarxa no és capaç de realitzar la transferència d'estil en alguna regió de la imatge, sol preservar els valors de la imatge de contingut original, com pot ser el color o les formes.

Una possible explicació d'aquest fenòmem podria ser que quan la xarxa no és capaç de realitzar la transferència, ja sigui per que la textura o l'esctructura distin molt de la imatge d'estil (una variació massa gran), es saltés la transformació en aquella zona de la imatge. Tot i així descarto aquesta teoria degut a que sovint sorgeixen aquests errors en regions molt semblants (comparteixen característiques semblants, en textura, forma i/o color) a altres de la mateixa imatge en que si s'ha fet la transformació.

5.2 Resultat de l'algorisme B

Per l'algorisme B, cal remarcar que l'execució de la xarxa ha sigut molt ràpida en cada una de les imatges. Es pot apreciar com la transferència d'estil elabora una textura de a partir dels elements més importants de la imatge artística, després l'aplica en la imatge de contingut però preservant tots els elements estructurals i seguint la relació entre els colors i formes de la imatge d'estil. Donant un resultat molt bó en la majoria de casos.

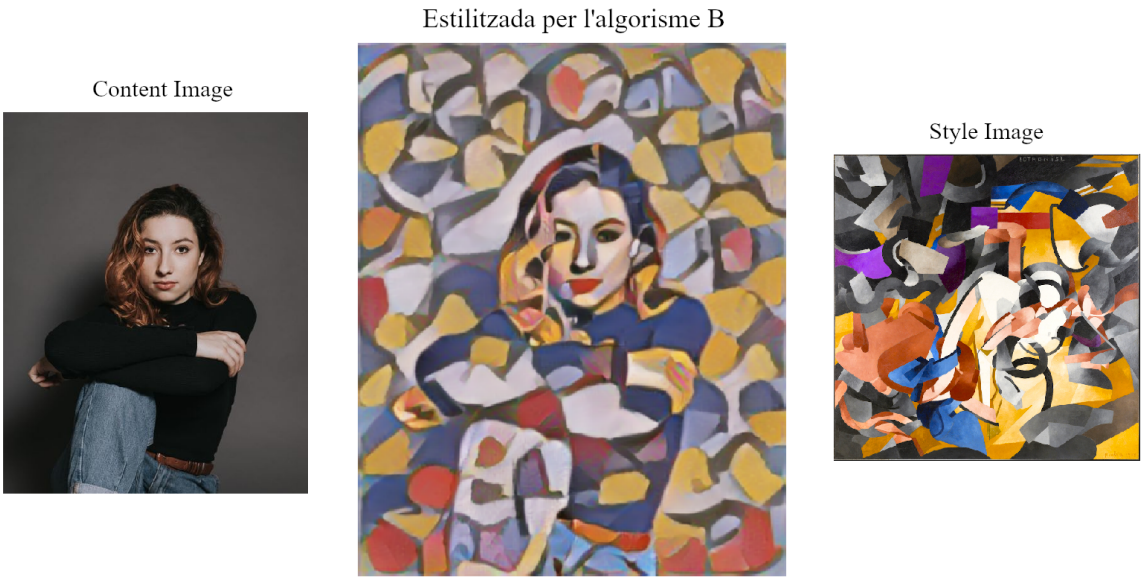


Figura 18: Algorisme B, estilització d'imatges beautygirl amb gothic.

En tots els resultats de la transferència, predominen els colors de la imatge d'estil, es mantenen els contrastos i variacions de color de la imatge de contingut. Aconseguint així una molt bona simbiosi entre les dues imatges, motiu pel el que el resultat resulta agradable de veure.

Observem en el cas de l'estil 'block', una imatge que amb pocs colors, ben delimitats i diferenciats i sense contrasts ni textures, aplicar l'estil realitzant una barreja dels colors de la original i definint tonalitats entre aquests. També observem amb aquest estil que si la xarxa no troba elements similars dins l'estructura entre les dues imatges, la reconstrucció no s'assembla gaire a la imatge de contingut, perdent molta informació rellevant. Això ho podem veure també, en molts casos, amb les imatges d'estil 'polygon', 'psychedelic', 'stainde_glass_flower' i 'synthwave'.

Amb la imatge de contingut 'vegetal', la qual és un dibuix fet a mà, es pot observar que tot i no tenir cap tipus de color pintat o textura, la transferència omple l'espai "pla" amb contingut. Això també es pot apreciar en les imatges contingut 'beautygirl', 'ski' i 'skysunset'.

5.3 Resultat de l'algorisme C

En aquest algorisme es buscaba una implementació òptimitzada del model VGG-16. L'execució d'aquesta ha requerit un ús de la memòria ram major que la resta per elaborar resultats amb les mateixes característiques. Es degut a això i les limitacions tècniques del meu dispositiu, que algunes de les imatges resultat d'aquest algorisme que es troben en el directori del projecte, estan a la meitat de resolució, tot i així els resultats són similars als del tamany original.

Figura 19: Algorisme C, estilització d'imatges beautygirl amb gothic.

Com a primer punt a destacar, el temps d'execució d'aquesta implementació és major que de la xarxa VGG-16 del segon algorisme, però molt més baix que de la xarxa VGG-19 del primer.

Observant els resultats del Style transfer podem apreciar que preserva en la transformació del contingut, els color de la imatge d'estil, evitant distar gaire d'aquests a la hora de traduir-se en les variacions de color que presenta la imatge de contingut original. Com es pot veure en el cas de la imatge d'estil 'block' que en altres implementacions barrejava els colors d'aquesta en la transferència.

Es pot apreciar que en l'aplicació de l'estil artístic a la imatge de contingut, trobem una certa aleatorietat en els colors i formes ignorant una mica, l'estructura que tenien originalment aquestes en la imatge de d'estill, creant noves formes per a la traducció artística i adaptant-se millor (que en els altres algorismes) a l'estructura de la imatge de contingut, donant uns resultats més "orgànics" en la transferència d'estil.

En contraposició, tenim que al realitzar la transferència d'estil, no es traspassen amb el mateix pés que en altres implementacions les formes i patrons que apareixen en la imatge d'estil. Perjudicant així la transferència amb estils amb formes i vores més definides, com són el cas de les imatges 'polygon', 'crystalforest', 'block', 'gothic' i 'stained_glass_flower'. Tot i que això passi, s'aconsegueixen resultats prou satisfactoris.

En resum, podem valorar positivament aquesta implementació ja que dóna bons resultats en un temps d'execució raonable, elaborant una imatge que combina l'estil artístic transformant-la organicament a l'estructura de la imatge de contingut.

A continuació, en la següent imatge, veurem la comparativa dels resultats dels tres algorismes Style transfer implementats, que estilitzen la imatge 'beautygirl' amb l'estil de 'gothic':

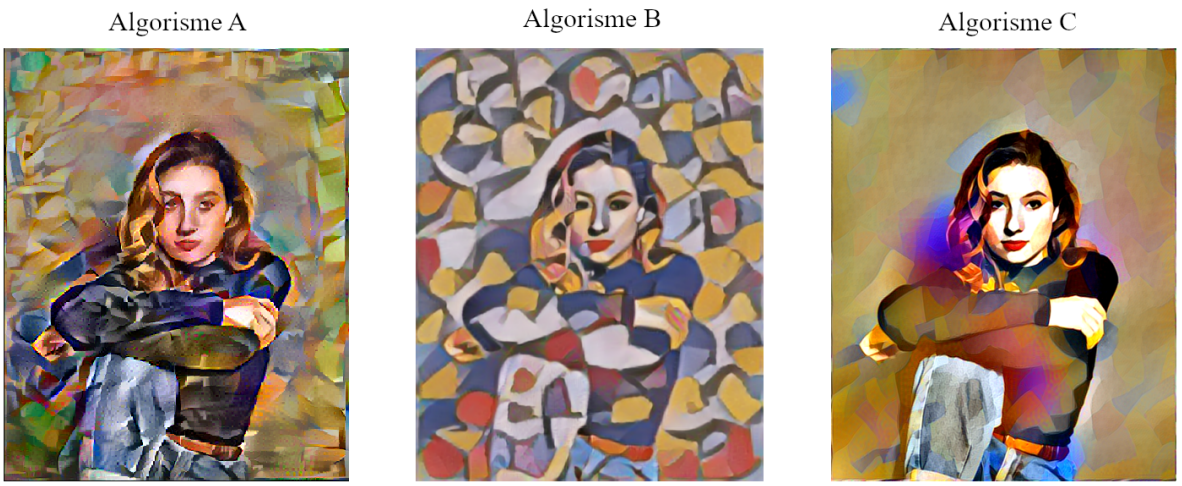


Figura 20: Estilització d'imatges beautygirl amb gothic amb els algorismes A, B i C.

Com es pot apreciar, tot i tenir les mateixes imatges content i style d'entrada, el resultat de cada un dels models implementats es completament diferent al de la resta. El resultat que dona una xarxa CNN al fer Style transfer ve determinat principalment de la configuració dels hiper-paràmetres abans de fer el càlcul, l'entrenament al que es sotmet prèviament el model i a la implementació que s'hagi fet en codi.

Per tant, encara que utilitzem només una de les implementacions, igualment podem aconseguir resultats molt diversos si configurem els hiper-paràmetres i/o sotmetem els models a entrenaments de manera diferent.

5.4 Resultat del model propi

Malauradament, amb un model entrenat per nosaltres amb la configuració definida en el capítol 3 d'aquest treball i amb 50.000 iteracions en l'entrenament. Els resultats de l'experiment realitzat no poden ser satisfactoris. En aquest punt, la xarxa encara no és capaç de realitzar una transferència d'estil i ha de seguir-se entrenant.

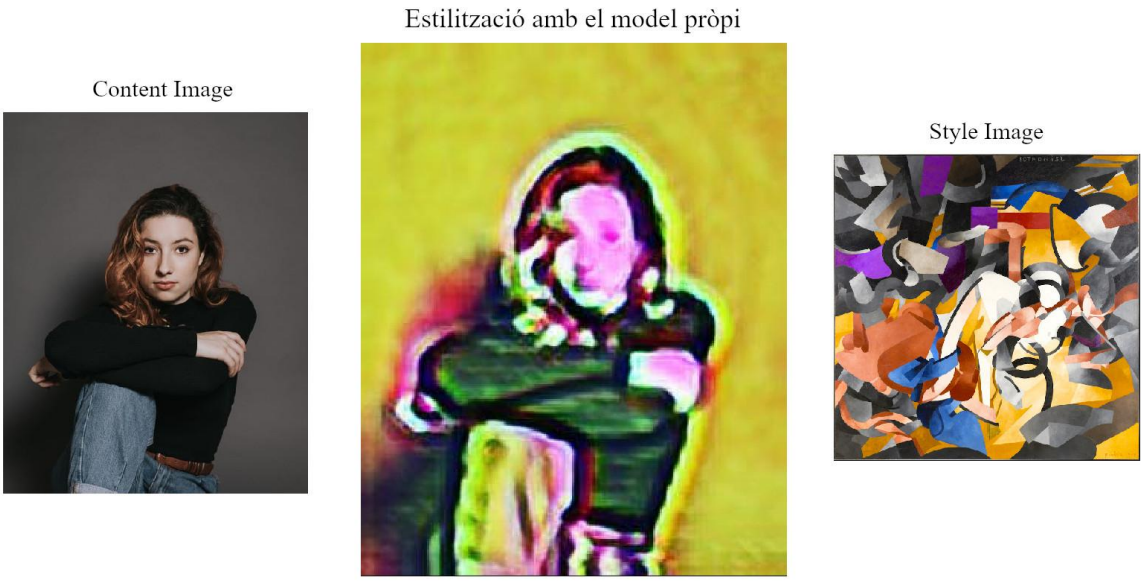


Figura 21: Model pròpi entrenat, estilització d'imatges beautygirl amb gothic.

Tot i així, observant el creixement durant el seu entrenament, vist en els punt 3.5.5 d'aquest treball, podem concloure que el model fa una gran progressió des de les primeres iteracions fins el punt on es troba actualment, de moment, és capaç d'extreure colors de la imatge d'estil i aplicar-los a la imatge de contingut, no obstant, sense modificar l'estructura d'aquesta.

Per millorar la xarxa actual s'hauria de fer el següent: augmentar el nombre de dades dels datasets definits en el apartat 3.4 d'aquest treball, reconfigurar els hiper-paràmetres definits en la xarxa i entrenar el model moltes més iteracions, com a mínim les recomanades per la documentació de magenta, 8.000.000 pasos.

Tot i no poder-ho provar i entrenar el model correctament. amb aquests canvis s'hauria de millorar la xarxa per a fer-la capaç de realitzar el neural Style transfer.

6. Conclusions i futurs treballs

Per acabar, faré una valoració sobre el compliment dels obnjectius plantejats al inici d'aquest treball i elaborarem les conclusions.

En primer lloc, hem realitzat una fase d'investigació en profunditat sobre les xarxes neuronals. Partint sense cap coneixement previ, he sigut capaç de comprendre el seu funcionament així com les seves aplicacions. He adquirit una gran quantitat de coneixements nous que em facilitaran, l'aprenentatge en altres àmbits de la intel·ligència artificial, Machine Learning i Deep Learning.

Em realitzat una recerca de quin és l'estat de l'art d'aquesta tecnologia i implementat tres propostes funcionals en el nostre codi. i realitzat un experiment pròpi per provar els límits d'aquesta tecnologia, cosa que em vist tant en la prova d'estrès plantejada com en el procés d'entrenament d'un model de xarxa neuronal propi.

Com a proposta de futur treball, podriem recuperar la idea inicial d'aquest TFG, desenvolupar una aplicació capaç de processar una imatge amb Neural Style Transfer prenent una imatge qualsevol com a estil, ja fos com a aplicacció web o mòbil. Proposta deixada de banda per ser massa ambiciosa, ara podria ser una bona proposta de treball, després d'haver realitzar aquest TFG.

En àmbit personal, puc concloure que aquest treball ha suposat per mi la porta d'entrada al món de la intel·ligència artificial i les xarxes neuronals. Ha sigut molt satisfactori aprendre tants coneixements nous i haver-los pogut treballar a través d'un àmbit tant vital per mi com és l'art il·lustrat. Acabo aquest treball preguntant-me quin serà el següent pas a seguir en el món del Machine i Deep Learning i plantejant-me seguir els estudis en aquest àmbit. Ja sigui en màster o en qualsevol altre tipus de formació professional.

Bibliografia

[1] Gatys, L. A., Ecker, A. S., & Bethge, M. (2015). A neural algorithm of artistic style. ***arXiv preprint arXiv:1508.06576, https://arxiv.org/abs/1508.06576.***

[2] Ghiasi, G., Lee, H., Kudlur, M., Dumoulin, V., & Shlens, J. (2017). Exploring the structure of a real-time, arbitrary neural artistic stylization network. ***arXiv preprint arXiv:1705.06830, https://arxiv.org/abs/1705.06830.***

[3] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. ***arXiv preprint arXiv:1409.1556, https://arxiv.org/abs/1409.1556.***

[3] Domina Neural Style Transfer – Explicación y ejemplo práctico (Pytorch). ***The Machine Learners, https://www.themachinelearners.com/neural-style-transfer-en-pytorch/.***

[4] Alexis Jacq. Neural Transfer using PyTorch. ***PyTorch, https://pytorch.org/tutorials/advanced/neural_style_tutorial.html.***

[5] PinkLAB. Ubuntu 20.04 (Mint 20.1) CUDA 11.1, cuDNN 8.0.5, Tensorflow 2.4. ***Youtube, https://www.youtube.com/watch?v=4gcqGxBIUnc.***

[6] Ringa Tech. Tu primera red neuronal en Python y Tensorflow. ***Youtube, https://www.youtube.com/watch?v=iX_on3VxZzk.***

[7] Ringa Tech. Redes Neuronales Convolucionales - Clasificación avanzada de imágenes con IA / ML (CNN). ***Youtube, https://www.youtube.com/watch?v=4sWhhQwHqug.***

[8] Chris Kevin. Feature Maps. ***Medium, https://medium.com/@chriskevin_80184/feature-maps-ee8e11a71f9e.***

[9] Transferencia de estilo neuronal. ***TensorFlow, https://www.tensorflow.org/tutorials/generative/style_transfer.***

[10] Transferencia de estilo artístico con TensorFlow Lite. ***TensorFlow, https://www.tensorflow.org/lite/examples/style_transfer/overview.***

[11] Transferencia rápida de estilos para estilos arbitrarios. ***TensorFlow, https://www.tensorflow.org/hub/tutorials/tf2_arbitrary_image_stylization.***

[12] Greg Surma. Style Transfer - Styling Images with Convolutional Neural Networks. ***Medium, https://gsurma.medium.com/style-transfer-styling-images-with-convolutional-neural-networks-7d215b58f461***
.

[13] Image Style Transfer Using Convolutional Neural Networks. ***GitHub, https://github.com/superb20/Image-Style-Transfer-Using-Convolutional-Neural-Networks.***

[14] Ander Fernández Jauregui. Cómo programar Neural Style Transfer en Python. ***Anderfernandez.com, https://anderfernandez.com/blog/programar-neural-style-transfer-python/.***

[15] Cawotte. Neural Style Transfer with Keras. ***GitHub, https://github.com/Cawotte/Neural_Style_Transfer.***

[16] Harish Narayanan. Convolutional neural networks for artistic style transfer. ***GitHub,*** ***https://github.com/hnarayanan/artistic-style-transfer*****.**

[17] Magenta. Fast Style Transfer for Arbitrary Styles. ***GitHub,*** ***https://github.com/magenta/magenta/tree/main/magenta/models/arbitrary_image_stylization#train-a-model-on-a-large-dataset-with-data-augmentation-to-run-on-mobile*****.**

[18] Red neuronal convolucional. ***Wikipedia,*** ***https://es.wikipedia.org/wiki/Red_neuronal_convolucional*****.**

[19] Convolución. ***Wikipedia,*** ***https://es.wikipedia.org/wiki/Convoluci%C3%B3n*****.**

[20] Diego Calvo. Red Neuronal Convolucional CNN. ***Diegocalvo.es,*** ***https://www.diegocalvo.es/red-neuronal-convolucional/#:~:text=Arquitectura,como%20una%20red%20perceptr%C3%B3n%20multicapa.*****.**

[21] Jaime Durán. Todo lo que Necesitas Saber sobre el Descenso del Gradiente Aplicado a Redes Neuronales. ***Medium,*** ***https://medium.com/metadatos/todo-lo-que-necesitas-saber-sobre-el-descenso-del-gradiente-aplicado-a-redes-neuronales-19bdbb706a78*****.**

[22] Sahil Singla. Experiments on different loss configurations for style transfer. ***Medium,*** ***https://towardsdatascience.com/experiments-on-different-loss-configurations-for-style-transfer-7e3147eda55e*****.**

[23] Mariano Rivera. Reeuso de Redes Preentrenadas. ***Postgrados Cimat,*** ***http://personal.cimat.mx:8181/~mrivera/cursos/aprendizaje_profundo/preentrenadas/preentrenadas.html#vgg16-una-red-convolucional-preentrenada*****.**

[24] ¿Cómo funcionan las Convolutional Neural Networks? Visión por Ordenador. ***Aprendemachinelearning.com,*** ***https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/*****.**

[25] Vicente Rodríguez. Conceptos básicos sobre redes neuronales. ***Vicentblog,*** ***https://vincentblog.xyz/posts/conceptos-basicos-sobre-redes-neuronales*****.**

[26] Juan Pablo Soto Quirós. Convolución Matricial Aplicado al Procesamiento de Imágenes. ***Tecnològico de Costa Rica,*** ***https://www.tec.ac.cr/sites/default/files/media/uploads/presentacion_pablosoto.pdf*****.**

[27] #013 CNN VGG 16 and VGG 19. ***Data Hacker,*** ***https://datahacker.rs/deep-learning-vgg-16-vs-vgg-19/*****.**

[28] Aman Kumar Mallik. Neural Style Transfer Using PyTorch. ***Medium,*** ***https://towardsdatascience.com/implementing-neural-style-transfer-using-pytorch-fd8d43fb7bfa*****.**

[29] Javier Nogueira. Art Style Transfer using different Neural Networks. ***Medium,*** ***https://towardsdatascience.com/art-style-transfer-using-neural-networks-a28f5888746b*****.**

[30] Konstantinos Monachopoulos. ImageNet-to-TFrecord. ***GitHub,*** ***https://github.com/kmonachopoulos/ImageNet-to-TFrecord*****.**

ANNEX
-
UNA REVISIÓ DE NEURAL STYLE TRANSFER

Índex:

A. Instalació del programari

Entorn virtual Python 3.10.4

Assumint que s'ha instalat el llenguatge de programació Python 3 en el sistema operatiu Ubuntu 20.04. A continuació veurem els passos a a seguir per a crear un entorn virtual python per poder treballar de manera aïllada a la resta del sistema i evitar posibles conflictes.

1. Creem un directori on posar l'entorn virtual i hi accedim en el CMD, com per exemple:

```
$ D: cd \venv_envs
```

   També es pot crear manualment i en el path escriure "cmd".

2. Creem l'entorn virtual amb la següent comanda:

```
$ python -m venv NOM_DEL_ENTORN_VIRTUAL
```

   Observem que s'ha creat una carpeta amb el nom que hem definit, allò és on està el nostre entorn virtual.

3.Activem l'entorn virtual amb el fitxer "activate.bat" dins del directori "Scripts"

```
$ cd ./NOM_DEL_ENTORN_VIRTUAL/Scripts
$ activate / deactivate
```

D'aquesta manera es crea l'entorn virtual on s'executarà la resta del treball.

Jupyter Notebook

Per instalar jupyter notebook cal executar les següents comandes en un entorn python:

```
$ pip install jupyterlab
$ pip install jupyter notebook
```

CUDA Toolkit 11.7, cuDNN 11.6 i TensorFlow 2.9.1

Aqui estan les referències per instalar CUDA Toolkit (CUDA Toolkit 11.7 Downloads | NVIDIA Developer) i cuDNN (cuDNN Download). Podeu seguir també les instruccions de (Setting Up CUDA, CUDNN, Keras, and TensorFlow on Windows 11 for GPU Deep Learning).

Per instalar TensorFlow (recordem dins del entorn virtual definit anteriorment):

```
$ pip install tensorflow
$ pip install –upgrade tensorflow-hub
```

Llibreries auxiliars de python

A continuació veurem un llistat de les llibreries auxiliars de python necessàries per fer funcionar el codi:

MatplotLib	`$ pip install matplotlib`
Numpy	`$ pip install numpy`
PIL	`$ pip install PIL`
Scipy	`$ pip install numpy scipy`

Magenta 2.1.3

Per tal de instalar l'entorn magenta en el nostre entorn virtual python i fer ús de les seves eines, s'han de seguir els següents passos:

1. Instalar el paquet pip de Magenta:

```
$ pip install magenta
```

1.1 Instalar dependencies adicionals necessàries per l'entorn (abans de fer el pip de Magenta):

```
$ sudo apt-get install python3.8-dev
$ sudo apt-get install libpython2.7-dev python-numpy
$ install TF.Slim: pip install --upgrade tf_slim
$ install tf-model garden: pip3 install tf-models-official
```

2. Clonar el repositori Magenta en local (ja està inclòs dins del projecte):

```
$ git clone https://github.com/tensorflow/magenta.git
```

3. Instalar les dependències canviant al directori base de magenta i executant la comanda:

```
$ pip install -e .
```

B. Directori del projecte

A continuació es farà una descripció del directori que conté tot el codi i recursos utilitzats en aquesta pràctica, partint des de l'arrel d'aquest:

➢ algoritmes / ▹	➢ algoritme_A.ipynb ➢ algoritme_B.ipynb ➢ algoritme_C.ipynb ➢ images / ○ (content images)*.jpg ○ styles / ■ (style images) *.jpg ➢ results / ○ algA / ■ (imatges output de algoritme_A.ipynb)*.jpg ○ algB / ■ (imatges output de algoritme_B.ipynb)*.jpg ○ algC / ■ (imatges output de algoritme_C.ipynb)*.jpg
➢ datasets / ▹	➢ DTD_PBN_train / ○ (DTD i PBN imatges training set)*.jpg ➢ imagenet / ○ test / ■ (imatges testing set)*.JPEG ○ testTFRecord / ■ imagenet_test.tfrecord ○ train / ■ (categories del trainig set en format WordNet)n* / ● (imatges categoria)*.JPEG ○ trainTFRecord / ■ (training set TFRecords) train-*-of-010124 ○ val / ■ (imatges validation set)*.JPEG ○ valTFRecord / ➢ dtd_pbn_train.tfrecord
➢ ImageNet-to-TFrecord-master / ▹	➢ build_imagenet_data.py ➢ imagenet_lsvrc_2015_synsets.txt ➢ imagenet_metadata.txt
➢ images / ▹	➢ (content images)*.jpg ➢ styles / ○ (style images) *.jpg ➢ output / ○ (imatges output del model entrenat per nosaltres)*.jpg
➢ implementacions exemple / ▹	➢ ex1_Transferencia_de_estilo_neuronal.ipynb ➢ ex2_Transferencia_rapida_de_estilos_arbitrarios.ipynb ➢ ex3_Transferencia_de_estilo_artistico_TensorFlow_Lite.ipynb ➢ ex4_Styling_images_with_Convolutional_Neural_Networks.ipynb

<table>
<tr><td></td><td>➢ ex5_Style_Transfer_Pytorch.ipynb
➢ ex6_Fast_Neural_Style_network.ipynb
➢ images /
○ (imatges de contingut)*.jpg
○ styles /
■ (imatges d’estil)*.jpg</td></tr>
<tr><td>➢ logdir / ▹</td><td>➢ train_dir / (directori que crea magenta al entrenar un model concret)
○ (fitxers tfevents) events.out.tfevents.*.puki-P6689-MD61022
○ graph.pbtxt
○ (fitxers checkpoint) model.ckpt-*
○ (fitxers index checkpoint) model.ckpt-*.index
○ (fitxers metadata checkpoint) model.ckpt-*.meta</td></tr>
<tr><td>➢ magenta / ▹</td><td>➢ setup.py (fitxer per instalar l’entorn magenta)
➢ setup.cfg
➢ magenta / (directori amb la llibreria magenta)
○ …</td></tr>
<tr><td colspan="2">➢ inception_v3.ckpt
➢ vgg_16.ckpt
➢ multistyle-pastiche-generator-monet
➢ LLEGEIX-ME.txt</td></tr>
</table>

Els fitxer marcats en vermell han sigut borrats al moment de lliurar l’entrega, degut a les limitacions de tamany imposades per a la realització d’aquesta, tots els fitxers eliminats es troben en un enllaç de google drive facilitat en el fitxer LLEGEIX-ME.txt juntament amb les instruccions d’on posar-los dins del directori.