Classificació Automatitzada d'Imatges Histològiques mitjançant una Xarxa Neuronal Convolucional. Una aplicació per al Tractament del Càncer Colorrectal.

Jan Borràs Ros
Màster en Bioinformàtica i Bioestadística.
Machine Learning

Nom Director/a
Edwin Santiago Alférez Baquero

Gener de 2019

FITXA DEL TREBALL FINAL

Títol del treball:	*Classificació Automatitzada d'Imatges Histològiques mitjançant una Xarxa Neuronal Convolucional. Una aplicació per al Tractament del Càncer Colorrectal.*
Nom de l'autor:	*Jan Borràs i Ros*
Nom del consultor/a:	*Edwin Santiago Alférez Baquero*
Nom del PRA:	*Ferran Prados Carrasco*
Data de lliurament (mm/aaaa):	*01/2019*
Titulació o programa:	*Màster en Bioinformàtica i Bioestadística*
Àrea del Treball Final:	*Machine Learning*
Idioma del treball:	*Català*
Paraules clau	*Càncer Colorrectal, PyTorch, Xarxes Neuronals Convolucionals.*

Resum del Treball (màxim 250 paraules):

En aquest treball es vol entrenar una Xarxa neuronal convolucional amb capacitat per classificar 8 tipus de teixits propis de la histologia del càncer de còlon. Posteriorment es provarà l'eficàcia de l'algoritme per classificar les regions d'unes imatges obtingudes de biòpsies de pacients amb càncer colorrectal. Per construir la Xarxa Convolucional s'ha emprat el llenguatge de programació Python mitjançant la llibreria de funcions PyTorch desenvolupada i mantinguda per Facebook Inc. Conjuntament amb PyTorch s'ha utilitzat la llibreria de funcions Fastai, que permet aplicar diferents bones pràctiques mitjançant PyTorch. Les bones pràctiques són mètodes que s'apliquen a la Xarxa Convolucional per tal de millorar l'exactitud final i que han estat desenvolupats per altres investigadors en l'aplicació d'altres models de Xarxa Neuronal Convolucional.

S'ha utilitzat l'arquitectura resnet34 com a punt de partida, s'ha calibrat els pesos de les capes convolucionals per donar solució al nostre problema i s'ha assolit una exactitud d'aproximadament 95% en la classificació dels 8 tipus de teixits. Amb aquests resultats s'ha passat a provar l'algoritme en l'aplicació d'un cas real. Per això s'ha provat de classificar les imatges anomenades WSI (de Whole Slided Images en anglès) que són imatges obtingudes de biòpsies reals de pacients amb càncer de còlon. S'ha fragmentat les imatges i s'ha fet classificar aquests fragments a l'algoritme. Posteriorment s'ha tornat a construir les WSI amb els fragments classificats.

Com a resultat de l'estudi s'ha obtingut la configuració de Xarxa Neuronal Convolucional que ha permès obtenir una exactitud de 95% en el reconeixement dels 8 tipus de teixit, s'ha après a aplicar diferents bones pràctiques en la construcció d'una Xarxa Neuronal Convolucional i s'ha obtingut una aplicació d'aquest model en un cas real amb WSI de pacients amb càncer de còlon.

Abstract (in English, 250 words or less):

In the present study the main target is to train a Convolutional Neural Network with the capacity to classify 8 different colorrectal cancer tissue types from cancer histology. After that we tested the performance of the algorithm to classify the regions of real biopsy images obtained from colorrectal cancer patients. To build the Convolutional Network we have used Python with PyTorch library, developed and mantained by Facebook Inc. Altogether with PyTorch we have used Fastai library which allows to use a set of functions to test good practices with the aim to improve final accuracy. This good practices methods have been developed by other researchers in the application of other Convolutional Neural Network models.

We have implemented the resnet34 deep neural network architecture to the Tissue classification problem and we obtained a 95% accuracy in the classification of the 8 tissue types. With this results we assessed the efectivity of the algorithm to classify the regions in the whole slided images from real patients with colorrectal cancer. We fragmented the images and classified the fragments with the algorithm. Finally we have rebuilt the WSI using the classified fragments.

As final product we have obtained the setup to build a Convolutional Neural Network which recognizes 8 different colorrectal cancer tissue types, we have learned how to apply different good practices in the building of a Convolutional Neural Network and we obtained an application of this model to a real problem with colorrectal cancer patients WSI.

Índex

Índex de figures

1 Plantejament del treball

1.1 Context i justificació del Treball

Avui dia el càncer colorrectal (CRC) és la tercera causa de càncer més freqüent al món, i també un dels que causa més morts. Pel què fa Catalunya, estudis de l'Institut Català d'Oncologia mostren que si es tenen en compte els dos sexes, el càncer de colon és la principal causa de càncer. Aquest tipus de càncer es forma a partir de cèl · lules canceroses que apareixen en l'intestí gros, i té una especial incidència en la població de més de 50 anys. És a partir d'aquesta edat, o fins i tot abans que ja es comença a fer proves tant òptiques com analítiques per analitzar la possible presència de la malaltia en el pacient. Alguns exemples més significatius d'aquestes proves són la colonoscòpia i l'anàlisi de sang oculta en femta. Aquestes serveixen per fer una primera valoració i en funció del resultat obtingut en aquestes primeres proves es procedirà o no a l'extracció d'una mostra de teixit (Biòpsia). El diagnòstic precoç de la malaltia pot ajudar al tractament de la malaltia i reduir significativament les taxes de mortalitat.

Les biòpsies permeten obtenir mostres de la capa de mucosa del teixit epitelial del recte, que és on s'origina el càncer a partir d'uns pòlips que en presència de certs factors de risc poden esdevenir cancerosos. Un cop s'ha obtingut les mostres es procedeix a la preparació de les imatges histològiques a partir de les quals es podrà realitzar els anàlisis pertinents. El metge farà una observació exhaustiva d'aquestes imatges mitjançant el microscopi i determinarà si hi ha presència de cèl · lules canceroses, i si n'hi ha, determinarà en quina fase de la malaltia es pot trobar el pacient. El camp que estudia aquest tipus de tractaments s'anomena histopatologia, i els últims anys ha rebut progressos força significatius a mans de la intel · ligència artificial. És possible classificar les imatges de les biòpsies de manera automatitzada i per tant es vol explorar el mètode capaç de fer aquesta tasca.

L'algoritme de classificació que s'entrenarà per realitzar aquesta tasca s'anomena Xarxa Neuronal Convolucional (CNN en anglès, del terme "Convolutional Neural Network"). Aquest mètode està àmpliament estudiat i la seva aplicació s'ha realitzat en molts camps de la ciència i la tecnologia, per això es vol crear l'algoritme mitjançant la transferència del coneixement a l'utilitzar una arquitectura de CNN preentrenada, entrenar-la i provar la seva eficàcia a l'hora de classificar les imatges histològiques de càncer colorrectal.

1.2 Objectius

1.2.1 Objectius Generals

1. Fer una aproximació alternativa als tractaments del càncer amb un enfocament des de la intel · ligència artificial.
2. Utilitzar els mètodes d'anàlisi de dades automatitzats per fer un screening de dades mèdiques.
3. Oferir alternatives de tractament potencials per al sector de la salut a l'altura del últims avenços tecnològics.
4. Construir una CNN per classificar imatges histològiques de càncer colorrectal.

1.2.2 Objectius Específics

1. Constatar l'estat actual de la tecnologia (State of the art).
2. Recopilar software necessari i configurar els entorns de treball.
3. Preparar base de dades amb les imatges histològiques de càncer colorrectal.
4. Configurar la Xarxa Neuronal Convolucional.
5. Comprovar l'efectivitat de l'algoritme en l'aplicació d'un cas real.

1.3 Enfocament i mètode seguit

Es parteix de la base de coneixement de l'autor alhora de desenvolupar el treball. Es seguiran les pautes que es poden trobar en diferents llibres especialitzats juntament amb els coneixements obtinguts en assignatures com Machine Learning per desenvolupar el contingut del present informe. A partir de les directrius indicades i mitjançant el llenguatge de programació Python s'utilitzarà diferents eines informàtiques per desenvolupar la tasca mitjançant un ordinador amb sistema operatiu Ubuntu 18.04.1 LTS.

Hi ha disponibles moltes llibreries de programari lliure per a desenvolupar algoritmes de Deep Learning a internet. La llibreria que s'ha escollit per desenvolupar aquest cas concret és PyTorch, desenvolupat i mantingut per l'empresa Facebook Inc. Per programar en Python s'ha escollit la plataforma Anaconda, la qual s'instal · larà també en l'entorn de treball d'Ubuntu.

De cara a la redacció de la memòria s'ha escollit R Markdown com a eina de treball i editor de textos. Aquest permet redactar l'informe, incloure codi en diversos llenguatges, executar el codi i compilar-lo a LATEX o HTML entre altres. LATEX és una potent eina alhora de redactar informes en format PDF. La correcte utilització de les eines esmentades fa que aquest sigui un estudi reproduïble, el qual és fàcilment demostrable si es disposa de les eines utilitzades en l'informe.

1.4 Planificació de la feina

El treball s'engloba dins l'assignatura del Treball de Final de Màster amb 15 crèdits ECTS. 1 crèdit ECTS equival a unes 25 hores de treball aproximades de manera que es disposa d'unes 375 hores de dedicació. Aquestes hores es repartiran entre les diferents etapes i PACs que contempla el pla de treball del màster.

- 20% del temps de dedicació per definir el pla de treball i els continguts.
- 60% del temps de dedicació per desenvolupar la recerca.
- 20% del temps de dedicació per redactar la memòria i fer la defensa oral.

	Data Inici	-	Data Final
PAC1 - Pla de treball	02/10/18	-	15/10/18
PAC2 - Desenvolupament del treball - Fase 1	16/10/18	-	19/11/18
Llegir Bibliografia	16/10/18	-	23/10/18
Recopilar Software Necessari	24/10/18	-	26/10/18
Configurar Entorns de Treball	29/10/18	-	02/11/18
Processat de Dades Utilitzades	02/11/18	-	12/11/18
Redactar PAC2	12/11/18	-	19/11/18
PAC3 - Desenvolupament del treball - Fase 2	20/11/18	-	17/12/18
Configurar CNN	20/11/18	-	03/12/18
Provar l'algoritme	04/12/18	-	10/12/18
Contrastar eficàcia CNN	11/12/18	-	17/12/18
PAC4 - Redacció de la Memòria	16/10/18	-	02/01/19
Tancament de la Memòria	18/12/18	-	02/01/19
Elaborar presentació oral	03/01/19	-	10/01/19
Defensa Pública TFM	14/01/19	-	23/01/19

Figura 1: Calendari amb la planificació de la feina a realitzar durant l'assignatura.

1.5 Breu sumari de productes obtinguts

Del present treball se n'obtindrà el codi per programar una CNN capaç de classificar 8 tipus de teixit histològic a partir d'imatges de biòpsia. També se n'obtindrà el coneixement que permetrà seguir desenvolupant la tecnologia més enllà del pla de treball d'aquesta assignatura.

1.6 Breu descripció dels altres capítols de la memòria

L'estudi inclou 4 grans capítols en els que es recull tota la informació i feina necessaris per fer reproduïble i transparent el treball realitzat.

- 1. Plantejament del treball: Capítol que recull tot el plantejament inicial del treball.
- 2. Introducció: Quan es tracta un tema tant complex que engloba tantes branques de coneixement es fa important resumir els conceptes més importants a tenir en compte a l'hora d'entendre tot el procés. Per això s'ha resumit 4 temes clau que permetran al lector entendre la feina des d'una perspectiva més propera a la de l'autor.
- 3. Materials i Mètodes: Inclou tots els procediments i especificacions de les dades que s'ha emprat en l'elaboració de la feina.
- 4. Resultats: Resumeix els progressos realitzats durant l'estudi.
- 5. Conclusions: abstraccions realitzades a partir dels resultats obtinguts i possibilitats de futur per a la tecnologia. Progressos realitzats en contraposició de l'estat de la tecnologia.
- 6. Glossari: Resum dels termes més utilitzats en la memòria.
- 7. Bibliografia: Citacions a altres articles científics.
- 8. Annex: Capítol que recull totes aquelles tasques que no es poden incloure en la memòria per la seva extensió. Principalment recull tot el codi Python utilitzat en l'estudi.

2 Introducció

2.1 Explicació del Problema

En el context d'aquest treball es pretén abordar un problema sortit de la metodologia de l'anàlisi de dades mèdiques amb aplicació específica en el tractament del càncer colorrectal.

El càncer és una malaltia que pot prendre estructures molt complexes, pot fins i tot formar nous teixits dins del propi tumor (J. N. Kather et al. (2016)). És per aquest motiu que un estudi enfocat a caracteritzar aquests teixits pot ser una bona manera de determinar l'evolució de la malaltia. Ara bé, d'aquest fet cal destacar-ne que el càncer és molt divers i el seu tractament depèn únicament de les decisions que pugui prendre el metge. Per això cal dotar els metges de les eines més vàlides amb les que pugui prendre la millor decisió per al tractament del pacient, com per exemple trobar mètodes que minimitzin el marge d'error en la presa de decisió dotant-lo d'algoritmes per la detecció automàtica de quadres clínics concrets (T. D. P. B (2017), Evans et al. (2014)).

Per altre banda, l'avenç de la tecnologia ha permès que es desenvolupin mètodes de classificació automàtica els quals han assolit paràmetres d'exactitud mai aconseguits amb anterioritat per un algoritme de classificació, fent que per primera vegada a la història es pugui plantejar utilitzar algoritmes de deep learning en aplicacions mèdiques. En el present estudi es pretén aprofundir en el desenvolupament d'aquests algoritmes i s'abordarà el problema intentant una millora dels resultats obtinguts en estudis anteriors (J. N. Kather et al. (2016)). Per això s'aplicaran una sèrie de bones pràctiques que milloren l'exactitud d'altres CNN per veure si és possible millorar l'exactitud obtinguda fins al moment.

L'algoritme consisteix en un classificador d'imatges mitjançant la tecnologia de les CNN que permet determinar quin tipus de teixit és el que hi ha en la imatge sense la necessitat de classificar les imatges manualment. Això és parcialment cert, ja que aquest utilitza les dades que se li proporcionen en un principi, imatges classificades manualment en aquest cas, per a fer les classificacions amb noves dades que no han estat classificades per cap usuari. El problema plantejat en aquest article consisteix en entrenar la CNN construïda a partir d'una CNN preexistent, per classificar els teixits de les imatges en 8 teixits diferents rellevants en l'estudi del càncer de colon (veure Materials i Mètodes). Per últim s'aplicarà aquest algoritme entrenat per fer una classificació d'imatges de biòpsies (WSI) per veure l'eficàcia de l'aplicació en un cas real mitjançant un canal de color RGB (Pantanowitz, Farahani, and Parwani (2015)).

2.2 El Càncer Colorrectal

En aquest apartat de la introducció es pretén donar dades objectives per veure el nivell d'importància que té la recerca en millorar els tractaments d'aquest tipus de càncer.

El càncer colorrectal és una malaltia que té una incidència global per sobre de 1.3 milions de casos nous per any, amb 694000 morts a tot el món a l'any (Pollheimer et al. (2010)). Un informe de l'any 2010 de l'Institut Català d'Oncologia mostra com a Espanya el càncer colorrectal és la segona causa de mort en homes després del càncer de pulmó, i la segona en dones després del càncer de mama. L'estimació indica que a Espanya cada any es diagnostiquen aproximadament 26.000 nous casos de càncer de còlon i gairebé la meitat dels pacients moren a causa de la malaltia. El mateix informe també indica que la millor manera d'augmentar l'índex de supervivència és amb un diagnòstic precoç de la malaltia, abans que aquesta s'hagi estès més enllà de l'intestí (ICO (2010)). És en aquest fet on s'intenta enfocar la recerca del present estudi, en l'elaboració d'eines de detecció precoç de la malaltia.

Durant la progressió de la malaltia l'estructura del càncer de colon varia notablement. Des d'una perspectiva histopatològica els tumors sòlids estan compostos a la vegada de diferents teixits com ja s'ha indicat, aquests teixits inclouen des de les cèl · lules clonals que invadeixen teixits, estromes, infiltracions en les cèl · lules del SI, zones de cèl · lules necròtiques i illes de cèl · lules no malignes. Quantificar la composició de cada tipus de teixit és una tasca important en la histopatologia del càncer colorrectal(J. N. Kather et al. (2016)). És

per això que s'analitzen imatges de talls histològics fetes a partir de biòpsies de pacients per determinar si aquests tenen o no la malaltia, i si la tenen per poder determinar en quin estadi de la malaltia es troben.

Figura 2: Mortalitat segons el tipus de càncer en homes a Espanya. Figura extreta de l'informe ICO de l'any 2010.

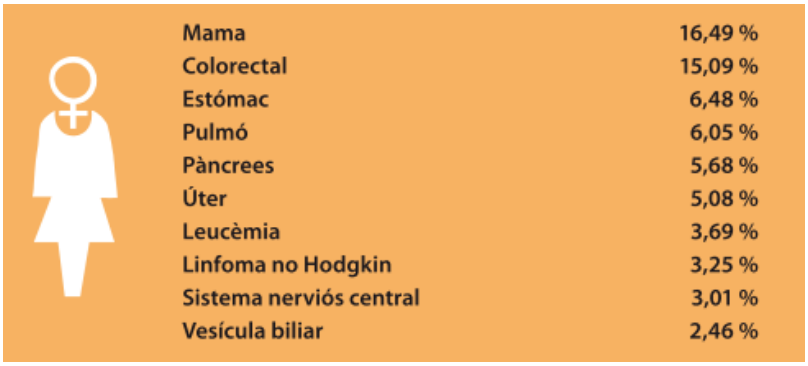

Figura 3: Mortalitat segons el tipus de càncer en dones a Espanya. Figura extreta de l'informe ICO de l'any 2010.

Les imatges de talls histològics són la principal font d'informació utilitzada en aquest estudi, ja que es vol estudiar un mètode de classificació d'aquestes imatges sense necessitat de la supervisió d'un metge especialitzat.

2.3 El Machine Learning com a Solució a Malalties

El Machine Learning (ML) és una ciència que permet enregistrar el coneixement dels humans i aplicar-lo a una màquina perquè el pugui interpretar i utilitzar. El seu desenvolupament ha estat en la seva major part durant els anys que van precedir l'any 2012 gràcies als avenços que ha fet la ciència de la informació, però és una ciència que va començar ja a mitjans del segle 20 amb les aportacions d'Arthur Lee Samuel, pioner en intel · ligència artificial i un dels primers en anomenar el terme "Machine Learning". La ciència que estudia la utilització del ML per oferir solucions a la ciència ja estava en desenvolupament fa 40 anys. L'estudi de l'aplicació en càncer aleshores estava molt lluny dels resultats obtinguts actualment, però ja mostraven la potencialitat que té aquesta tecnologia (Stenkvist et al. (1978)). És important tenir un context ampli dels fets per poder entendre millor els problemes.

Aquesta ciència ha tingut millores significatives en els últims anys i des d'aleshores han començat a sortir al mercat tot tipus de millores que utilitzen aquesta tecnologia com a base del seu funcionament. Les aplicacions mèdiques són només un exemple de la seva aplicació, però molt rellevant per la societat degut a la potencialitat que té per oferir solucions per al tractament de malalties (Evans et al. (2014)). Fins avui el

principal problema del ML en aplicacions mèdiques ha estat l'obtenció de les dades. Actualment ja estan començant a sortir molts conjunts de dades que solucionen aquest problema (J. N. Kather et al. (2016) i Fabio A. Spanhol et al. (2015)) i han permès desenvolupar aquesta versió de CNN. És especialment important seguir ampliant els conjunts de dades públics per tal de millorar el desenvolupament de la tecnologia de cara al futur.

Un dels problemes més importants que afecta al ML és el desenvolupament de característiques per a operar els algoritmes. Constantment es desenvolupen nous mètodes per estudiar la informació des de noves perspectives (Fabio Alexandre Spanhol et al. (2016)). És per això que el Deep learning (DL) s'ha tornat una de les eines amb més eficàcia alhora de classificar imatges histològiques, ja que el propi algoritme entrena els seus filtres per extreure les característiques de les imatges a través de les anomenades "capes convolucionals" (Goodfellow, Bengio, and Courville (2016)). El marge d'error en Deep Learning rara vegada està per sobre del 15% quan es tracta de l'anàlisi d'imatges histològiques (J. N. Kather et al. (2016), Fabio Alexandre Spanhol et al. (2016)).

Un altre factor a tenir en compte en el progrés del ML ha estat el desenvolupament de les llibreries de programari obert com Python o R. Si bé aquestes llibreries no han estat essencials sí que han permès que comunitats d'investigadors col · laborin en la confecció de nous mètodes i funcions per millorar la tecnologia conjuntament. Es podria dir que no són la causa però sí una conseqüència del desenvolupament del sector de la intel · ligència artificial, ja que la col · laboració entre centres de recerca facilita i accelera la investigació. Cada vegada té més importància la utilització d'eines de programari obert en el desenvolupament del ML.

2.4 Deep Learning i Transferència del Coneixement

El Deep Learning s'engloba dins el Machine Learning i és un mètode que permet analitzar informació en forma de model no lineal, que processa la informació a través de diferents capes de neurones per extreure característiques rellevants de les dades originals i que permet realitzar tasques de classificació d'objectes (Goodfellow, Bengio, and Courville (2016)). L'ajustament de les diferents capes de neurones és el que permet adaptar l'algoritme al problema que es vulgui tractar, per tant és important tenir en compte la composició de les capes intermedies per oferir una solució ajustada al problema.

El DL té la seva aplicació més significativa actualment dins el camp de la visualització de dades mèdiques, mitjançant les anomenades Xarxes Neuronals Convolucionals. Aquest sector s'encarrega de crear eines que facilitin l'exposició de les dades obtingudes d'estudis mèdics per fer-les més transparents i facilitar la transferència del coneixement (Evans et al. (2014)).

Una CNN és una aplicació del DL en què s'analitza informació principalment en forma d'imatges. És d'especial importància el fet que aquestes poden treballar amb dades molt diferents de les dades amb les que s'ha treballat en les fases d'entrenament i són capaces de generalitzar a problemes molt diversos amb un ajustament mínim de les capes intermedies (S. C. B et al. (2018)). Entre les seves aplicacions més rellevants podriem destacar-ne la capacitat de fer reconeixement de patrons en vídeos i imatges, la seva aplicació en sistemes de recomanació (Marketing), classificació d'imatges amb aplicació en el sector mèdic i el processat de llenguatge natural (NLP).

Les anomenades capes convolucionals consisteixen en un seguit de filtres (mitjançant kernels) els quals s'entrenen per processar la informació que aporten per exemple els píxels en una imatge, i al final es relaciona tots els processos de les dades mitjançant una capa completament connectada (Fully-Connected Layer) que relaciona totes les característiques que s'han analitzat en les capes inferiors (Goodfellow, Bengio, and Courville (2016)).

A continuació es mostren els tipus de capes convolucionals més rellevants per donar una idea sobre quin és el processat que es fa sobre la informació:

- Relu - Consisteix en agafar la informació d'entrada i només filtrar aquells valors que estan per sota de 0. Els valors per sobre de 0 es passen com a dades de sortida sense cap modificació.
- Linear - Aplica un model lineal a les dades de manera que cada valor del dataset estigui processat per aquest model, seguint l'esquema $a * x + b = y$.

- Maxpool - Maximitza el valor de sortida en funció dels valors d'entrada de manera que es filtren els valors més baixos.
- Sigmoid - Aquest tipus de processat és el que s'utilitzava més en el passat. Afegeix una capa en la xarxa neuronal que es basa en la forma de la funció sigmoide per activar/desactivar el filtre. Aquest mètode no era òptim per la configuració de les CNN i per això el seu ús ha estat substituït pel de les "Relu".
- Fully Connected Layer - Mentre que a resta de capes convolucionals donen forma a característiques de baixa dimensionalitat, aquesta el que fa és connectar la informació en última instància per donar forma final al model no lineal. Normalment s'utilitza una capa "softmax" que redueix la dimensionalitat de les dades i minimitza l'efecte de dades extremes.

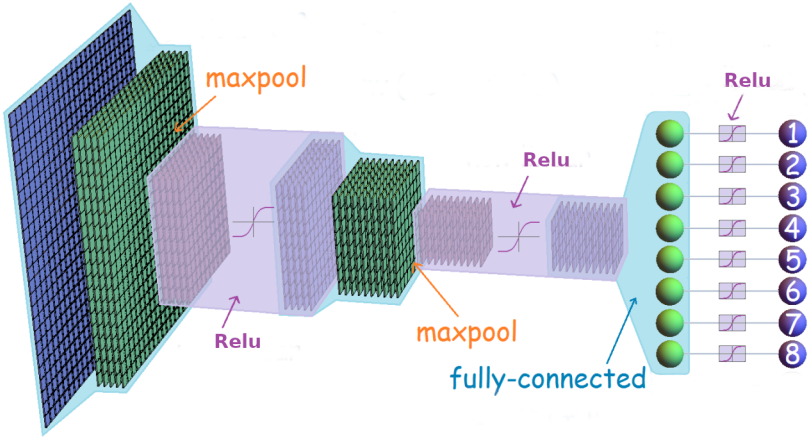

Figura 4: Exemple de CNN amb 8 classes finals. Imatge modificada a partir d'una figura del web gitbooks.io

En l'exemple de la figura 4 hi ha 6 capes que processen la informació, 3 Relu en punts diferents, 2 Maxpool també separades i una Fully-Connected Layer a la penúltima capa convolucional. Aquest exemple no té la mateixa configuració que la CNN que s'ha entrenat, l'algoritme que s'ha treballat durant l'estudi té fins a 34 capes convolucionals.

Les CNN com ja s'ha dit són un mètode molt versàtil i que permet donar solució a un ampli espectre de problemes amb un ajustament mínim del model. Això el que permet és que succeeixi el que s'anomena transferència del coneixement, i que és el que permet que molts dels estudis actuals en aquest camp esdevinguin possibles gràcies a la col · laboració entre comunitats científiques. Aquestes poden compartir els seus progressos a través d'internet per oferir les potencialitats dels seus descobriments i veure la seva possible aplicació en problemes de diferent naturalesa. Un exemple n'és aquest estudi, que intenta donar solució a un problema que tracta imatges histològiques dins el camp de l'histopatologia, i que en un principi els filtres que s'utilitzaven s'havien entrenat per classificar imatges d'objectes com avions, cotxes, animals i altres. Aquesta transferència del coneixement permet que l'algoritme i la solució que se'n derivi sigui fàcilment reproduïble, amb la mateixa facilitat amb la que s'ha ajustat l'arquitectura inicial per resoldre el el present problema (S. C. B et al. (2018)).

Un altre exemple de transferència del coneixement és la CNN Imagenet, la que ha estat entrenada amb més de 14 milions d'imatges anotades a mà i actualment s'utilitza per entrenar altres configuracions de CNN més específiques com resnet. La tecnologia té la base en aquest fet que és la transferència del coneixement.

Per aprendre el temari relacionat amb l'estructuració de CNNs s'ha realitzat el curs MOOC cs231n de la Universitat d'Standford cs231n.stanford.edu (última visita el dia 2/1/2019). Conjuntament amb aquest curs s'ha consultat els llibres Goodfellow, Bengio, and Courville (2016) i Aurélien (2017).

3 Materials i Mètodes

3.1 Configuració de l'Ordinador

Per poder dur a terme l'estudi aquí proposat es disposa d'un ordinador personal amb el que es duran a terme totes les tasques de tractament i anàlisi de dades. En cap moment s'ha utilitzat un ordinador extern, tot el procés s'ha dut a terme completament en aquest ordinador que s'específica a continuació.

3.1.1 Hardware

L'ordinador disposa d'un processador Intel® Core™ i7-6700K CPU @ 4.00GHz × 8 nuclis, una targeta gràfica NVIDIA GeForce GTX 1080 8 GB GDDR5X, 16GB de memòria RAM i un disc dur SSD M.2 de 500GB Samsung 860EVO.

3.1.2 Firmware

El firmware engloba tot aquell programari que permet utilitzar el hardware instal · lat a l'ordinador. En aquest cas s'utilitza els drivers oficials d'NVIDIA 410.78 i CUDA 9.1 per poder configurar i accedir a la targeta gràfica mitjançant PyTorch. També s'instal · la els drivers necessaris per fer funcionar totes les peces de hardware indicades en l'apartat anterior.

3.1.3 Software

El software utilitzat en l'estudi ha estat un sistema operatiu Ubuntu 18.04.1 LTS, R Studio muntat directament sobre el directori principal mitjançant R 3.5.1, Mendeley Desktop per gestionar la bibliografia, el gestor d'entorns de treball ANACONDA, i dins l'entorn de treball generat amb CONDA s'ha instal · lat les llibreries necessàries per executar tot el codi de l'estudi i la "Jupyter Notebook". Aquestes llibreries inclouen Python 3.6.6, cudNN 7.2.1, PyTorch 0.3.1 i la llibreria de "Fastai" 0.7 que inclou tota una serie de paquets i funcions utilitzats per facilitar la construcció de la CNN. També s'han utilitzat paquets de Python com NumPy, Pandas o Pillow.

Una de les claus de l'estudi ve donada per l'utilització de PyTorch per construir l'estructura de la xarxa neuronal. PyTorch és una llibreria de funcions construïda pel departament d'Intel · ligència Artificial de Facebook Inc. que ofereix una alta flexibilitat per construir algoritmes de Deep Learning i la capacitat d'utilització de "Tensors" dins la GPU. Els Tensors són el tipus de matriu que s'utilitzen per construir l'estructura dels algoritmes de DL.

L'ús d'entorns virtuals per treballar facilita molt l'ordre i la reproductibilitat de la feina, sobretot per tenir un inventari personalitzat i definit per cada projecte amb tots els paquets i llibreries necessaris. Una vegada s'ha configurat tota la CNN es pot guardar la configuració i compartir aquesta informació per poder construir-la en un altre servidor per exemple.

3.2 Conjunt de Dades

Les dades han estat obtingudes de l'article de JN Kather de l'any 2016 (J. N. Kather et al. (2016)) i es fan disponibles a través del web (última visita dia 26/12/2018):

DOI: 10.5281/ zenodo.53169

Aquestes dades estan protegides sota Creative Commons Attribution 4.0 International License

Els scripts generats en l'estudi s'han pujat en un repositori de GitHub per tal de fer més transparent tot el procés que s'ha dut a terme. El podeu trobar en aquest enllaç (última visita dia 26/12/2018):

Enllaç al repositori de GitHub: github.com/Deak3/CRC_CNN_Histopathology

Es garanteix que totes les dades han estat correctament anonimitzades i que segueixen els estaments de la declaració de Helsinki.

Les 10 imatges WSI, de 5000 $\times$ 5000 píxels, han estat obtingudes de l'arxiu de patologia de la Universitat de Heidelberg, al Centre Mèdic de Mannheim. S'ha seleccionat les imatges que tenien càncer i s'ha tingut en compte que en algunes la regió cancerosa era més àmplia que en d'altres. Si bé les imatges no han estat preparades per l'investigador, aquestes han estat obtingudes de l'article de JN Kather (J. N. Kather et al. (2016)), tant les WSI com les imatges utilitzades en la fase d'entrenament del model. Les altres imatges utilitzades en l'estudi són les que s'han utilitzat per entrenar el model. Aquestes es separen segons 8 tipus de teixits, i tenen una grandària de 150 $\times$ 150píxels (o bé 74μm x 74μm).

3.2.1 Dades per entrenar l'algoritme

Classificació dels talls histològics segons el tipus de teixit i la seva denotació dins la CNN, també es mostren diferents talls histològics representatius de cada tipus:

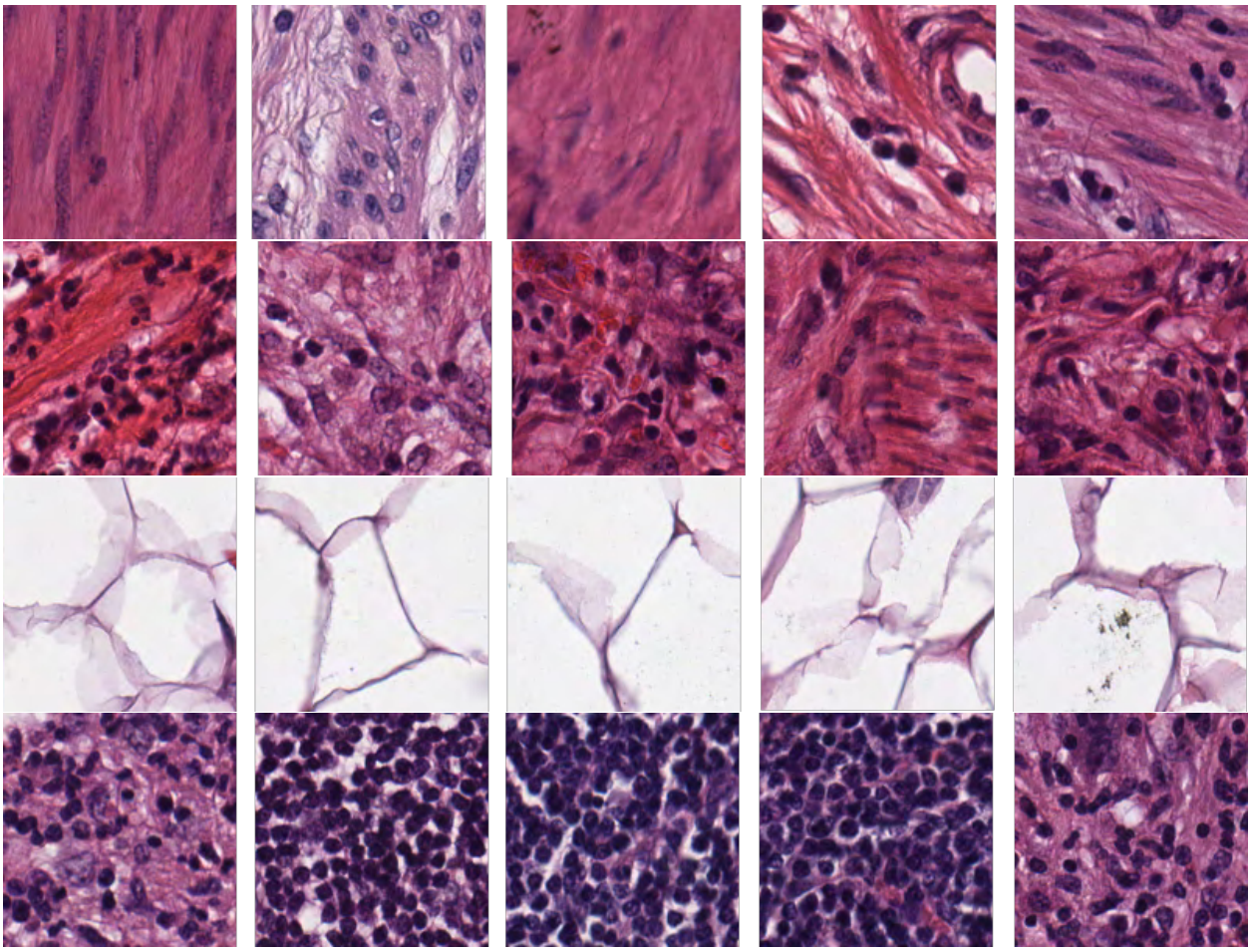

Figura 5: Mostres emprades per l'entrenament de la CNN.

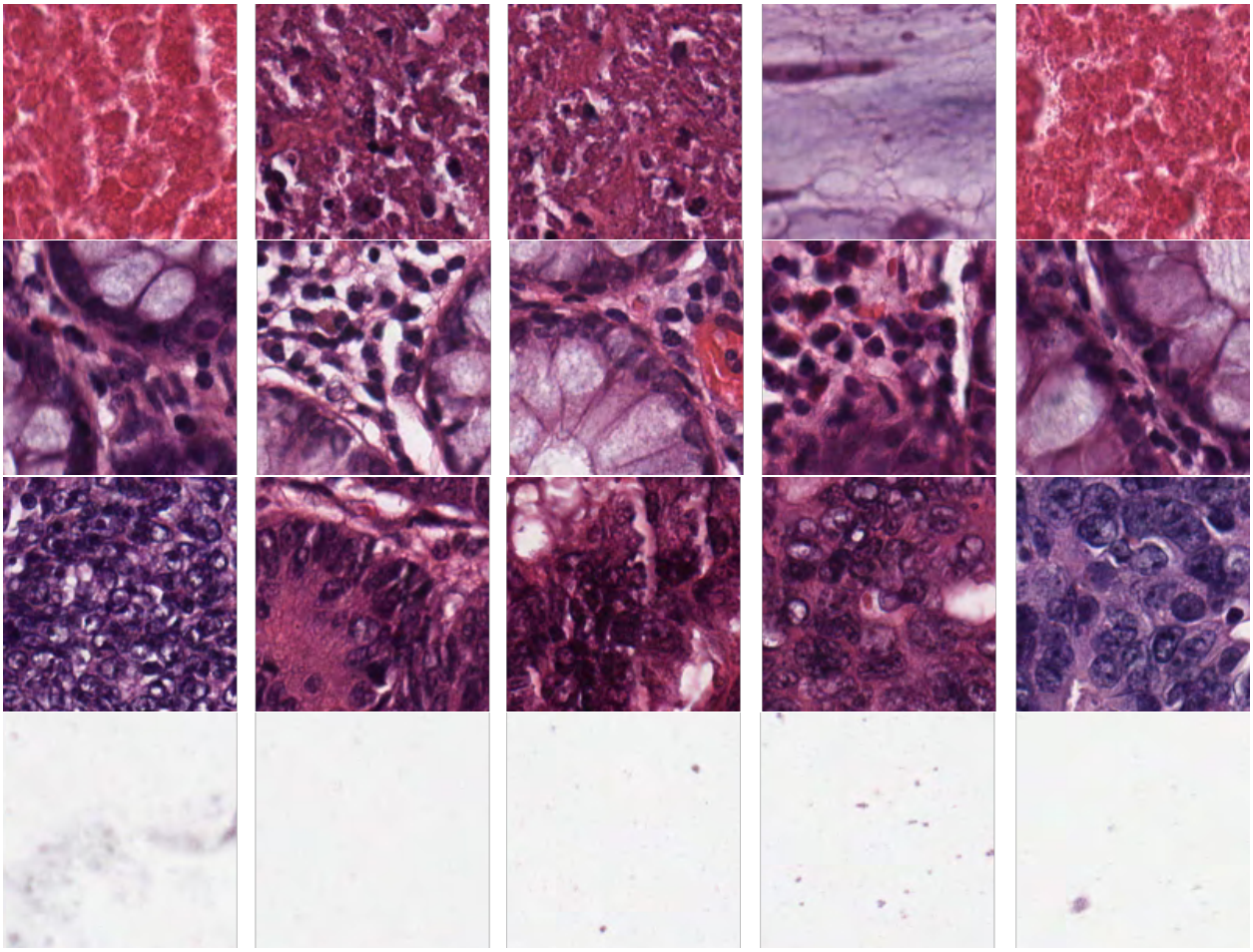

Figura 6: Mostres emprades per l'entrenament de la CNN.

En ordre segons apareixen en les figures 5 i 6 tenim:

1. Mostres d'estroma classificat com a stroma.
2. Mostres d'estroma complex classificades com a complex.
3. Mostres de teixit adipós classificades com a adipose.
4. Mostres de teixit del SI classificades com a lympho.
5. Mostres de mucosa classificades com a debris.
6. Mostres de glàndules mucoses classificades com a mucosa.
7. Mostres de teixit cancerós classificades com a tumor.
8. Mostres d'imatge sense teixit classificades com a empty.

Per tal de poder entrenar la CNN s'ha organitzat totes les imatges de 150 × 150 en "training set", "test set" i "validation set". El "training set" és el conjunt de dades utilitzat per entrenar pròpiament la CNN i ajustar-ne els pesos que li permeten prendre decisions. El "validation set" s'utilitza per provar l'algoritme amb dades que la CNN encara no hagi treballat i comprovar que no es doni "overfitting". I el "test set" que serveix per veure l'efectivitat de la CNN que s'ha construït. En l'apartat de descripció de l'algoritme s'aborda el funcionament de la CNN amb més detall. El contingut de la carpeta amb les imatges per fer l'entrenament conté un 60% de les dades totals, el "validation set" conté un 20% de les dades i el "test set" conté l'altre 20% de les dades. Dins de cada carpeta s'ha organitzat les imatges segons els 8 tipus de teixit per tal de que es pugui aprendre i classificar les imatges segons les etiquetes que se li proporcionen. Tot el repartiment de les imatges s'ha fet de manera aleatòria seguint les pautes indicades.

3.2.2 Whole Slide Images (WSI)

Les WSI s'han utilitzat en el nostre estudi per veure l'efectivitat de la CNN aplicada en un cas real. La utilització d'aquest tipus d'imatges en investigació mèdica està a l'ordre del dia (Pantanowitz, Farahani, and Parwani (2015)). Per això s'ha proposat seguir l'esquema de l'estudi de JN Kather (J. N. Kather et al. (2016)) i aplicar una nova versió de l'algoritme en 10 imatges que proporciona el propi article.

El que es vol és que l'algoritme classifiqui parts de la imatge per tal de que al reconstruir-la es pugui apreciar les regions segons els 8 tipus de teixit.

En la figura 7 es pot observar 9 de les 10 WSI emprades en l'estudi. S'ha escollit imatges que inclouen diferents regions amb teixit del tipus pel que s'ha entrenat a detectar l'algoritme, i totes elles tenen alguna regió cancerosa. La qualitat de les imatges s'ha reduït al mostrar-les per tal de facilitar el procés d'incorporació a l'article. La mida original era de 5000 × 5000píxels.

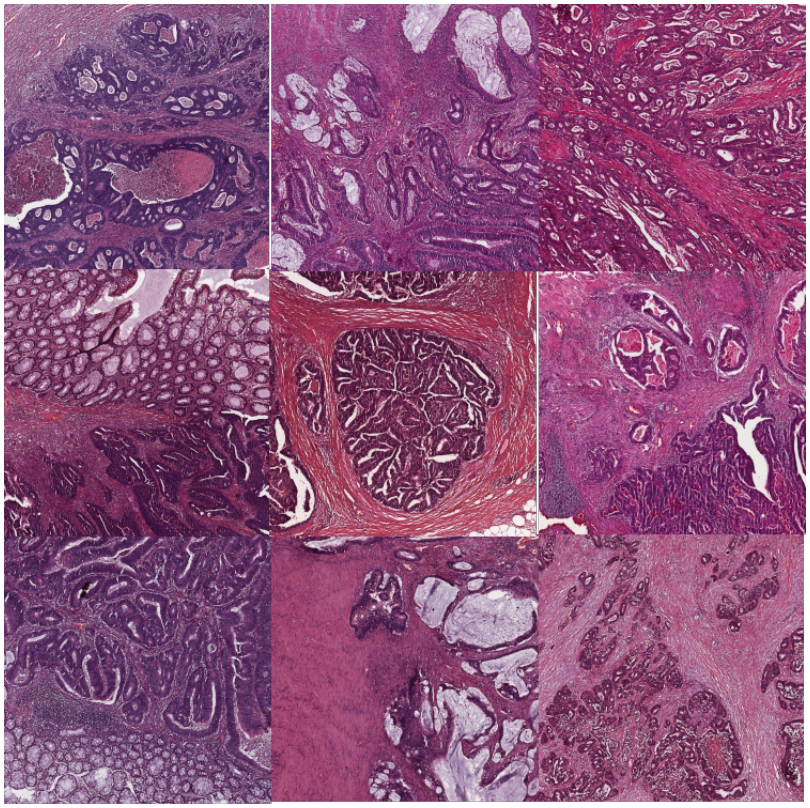

Figura 7: WSI emprades per provar l'efectivitat de la CNN en un cas real aplicat. S'ha omès una d'elles per practicitat a l'hora de mostrar les imatges.

3.3 Descripció de l'Algoritme

Durant tot el document s'ha anat fent comentaris sobre el funcionament de la CNN fent èmfasi als aspectes que es volien destacar pel que feia als apartats que s'estaven explicant. En aquest apartat es farà una explicació més àmplia de com s'ha fet per construir tota l'estructura del programa, processar i finalment determinar el tipus de teixit segons les regions de les WSI. El codi que s'ha utilitzat en llenguatge de programació s'incorpora a l'apartat d'annexos i es pot consultar per veure detalladament els passos que s'ha seguit. Molts dels passos que aquí es detallaran són una transcripció directe de la funció que s'executa amb codi de programació Python i en alguns casos Bash.

El primer pas ha estat configurar l'ordinador seguint els passos indicats en el seu apartat, s'ha fet el tractament de dades i és aleshores quan s'ha començat amb la configuració de la CNN mitjançant els paquets de Fastai, que estan muntats a dalt de tot de PyTorch. Quan es diu "muntats a dalt de tot" en realitat significa que són funcions que utilitzen el codi de PyTorch per crear les seves pròpies funcions. Amb els paquets de Fastai les funcions proporcionades garanteixen que s'estan complint certes bones pràctiques en la construcció de la CNN que a nivell d'usuari faciliten la feina per arribar a assolir bons resultats de classificació.

3.3.1 Exactitud i Loss Function

Un cop es tenen preparades totes les llibreries i programes instal · lats es procedeix a configurar la CNN fent el que s'anomena "Fine Tuning", que consisteix en entrenar la CNN provant diferents paràmetres de configuració per veure amb quin s'obtenen millors resultats. Les Imatges han estat repartides en 8 carpetes diferents. Nosaltres podem contrastar la correcte classificació d'aquestes gràcies a que quan l'algoritme faci la classificació de cada imatge podrà accedir a les carpetes d'on les ha tret i comprovar si ha fet la classificació

correctament. Amb aquest contrast s'extreu el valor de la "loss function".

Per valorar si s'està obtenint els resultats esperats s'utilitza dos valors per indicar al programador que la classificació avança en el bon sentit. El primer índex que s'utilitza durant l'entrenament de la CNN és el de la "loss function", que mesura la falta d'exactitud mitjançant l'equació 1. L'altre índex essencial per veure si s'està fent una classificació significativament correcte és el de l'exactitud, que calcula el percentatge total de classificacions correctes d'aquell procés de classificació seguint l'equació indicada a baix. En cada cicle d'entrenament de la CNN es mostra els valors d'aquests índex, i el número de "epochs" (èpoques) que és el número total de passades que fa el programa per totes les dades en cada cicle d'entrenament.

- Loss Function (Equació 1): $-(y \times log(p) + (1-y) \times log(1-p))$

  On "y" representa la classificació real de la imatge i "p" la probabilitat de que la classificació sigui correcte.

- Exactitud (Accuracy): nº Classificacions Correctes/nº de Classificacions Totals

L'exactitud cobrarà més importància al final de l'estudi, quan es vulgui evaluar el comportament final de l'algoritme.

```
def binary_loss(y, p):
    return np.mean(-(y * np.log(p) + (1-y)*np.log(1-p)))
```

Quadre de Codi 1: Equació Loss Function

La "loss function" també permet visualitzar si en algun dels passos de l'entrenament s'està donant "overfitting" o bé "underfitting". Aquests problemes fan referència al fet que les capes convolucionals estan o bé massa adaptades a les dades o bé massa poc adaptades respectivament, i per tant l'entrenament s'estaria donant de manera errònia. Se sap que hi ha "overfitting" quan el valor de la "loss function" és més baix en el resultat del "training set" que en el del "validation set", i el "underfitting" succeeix en el sentit contrari. És més comú el procés de "overfitting", ja que es dona si es sobreentrena la CNN o bé si no es disposa de prou dades per entrenar el model. En la taula 1 podem veure com es veuria un procés on s'estigui donant overfitting. Es pot observar la loss function disminuint de manera més pronunciada en el cas del training set.

Epoch	trn-loss	val-loss	accuracy
0	0.245918	0.264162	0.913177
1	0.213685	0.258654	0.921348
2	0.206906	0.25671	0.917263
3	0.193242	0.249137	0.919305
4	0.190431	0.261737	0.917263

Taula 1: Overfitted Batch process.

3.3.2 Gradient Descent

El "gradient descent" (GD) és un mètode algorítmic que permet trobar solucions òptimes a un conjunt de problemes complex, tal que es busca la solució òptima per la millor configuració de filtres de CNN (tenint en compte el resultat final) per determinar que una imatge efectivament és la classe que s'ha determinat. La idea general del "Gradient Descent" és que s'ha de provar paràmetres per minimitzar la "loss function" al màxim possible (Aurélien (2017)). Aquest mètode comença amb una configuració determinada dels paràmetres de l'algoritme, i agafant una direcció indicada per la disminució de la "loss function" va fent variacions dels paràmetres fins que s'arriba a un mínim de la funció, que vindria a ser la solució més òptima al problema. La mida d'aquestes variacions en la direcció de minimitzar el cost de la funció s'anomena "Learning Rate" (LR).

El “learning rate” determina la distància entre els valors que es proven en cada intent. Si bé la funció del GD és buscar la millor solució al problema, el LR determina el camí a seguir per arribar a aquesta solució. Si el LR és massa elevat és possible que l’amplitud dels intents faci passar d’un extrem a l’altre sense possibilitat d’arribar a la solució òptima, com cada problema té el seu perfil de comportament del cost es té que buscar el millor LR per cada problema. Per altre banda si és massa baix el temps de càlcul s’augmentarà fins a fer impossible l’aplicació de l’algoritme.

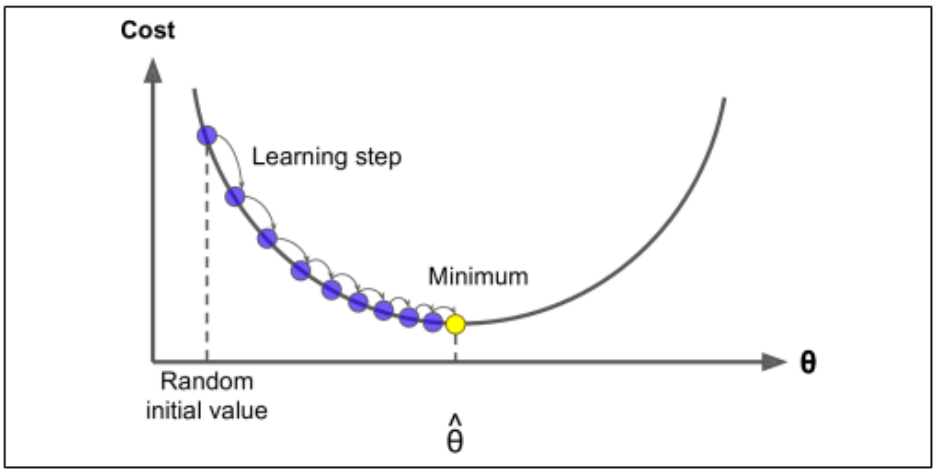

Figura 8: Els Learning Steps de la figura venen determinats pel LR. Figura extreta del llibre Hands-On Machine Learning with Scikit-Learn.

El tipus de “gradient descent” que s’ha fet servir en l’estudi és el “Stochastic Gradient Descent”, que consisteix en agafar valors d’una instància del “training set” de forma aleatòria i utilitzar-la per computar els gradients dels filtres.

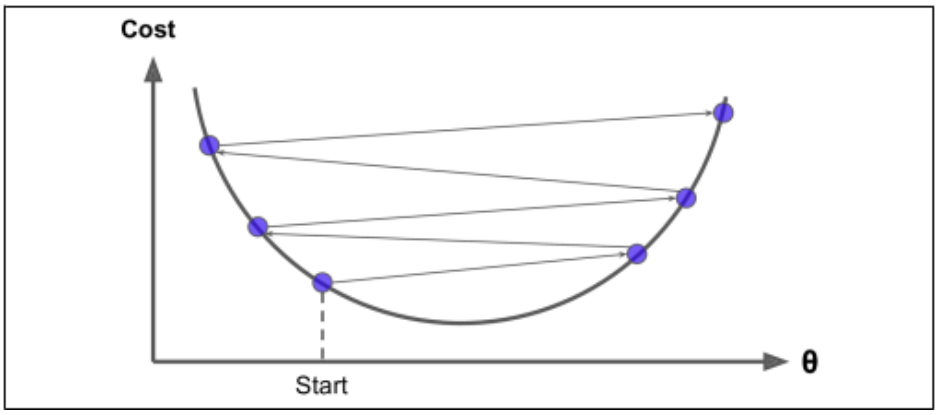

Figura 9: Figura que mostra què passa en un procés amb un LR massa elevat. Figura extreta del llibre Hands-On Machine Learning with Scikit-Learn.

3.3.3 Arquitectura i Entrenament de la CNN

En la introducció s’ha parlat de la facilitat de la transferència del coneixement en la tecnologia CNN, doncs és en aquesta part on es fa evident aquest fet. L’arquitectura de la CNN fa referència a l’estructura de filtres que s’adopta com a xarxa neuronal convolucional inicial. Les CNN necessiten moltes dades per ser entrenades, de manera que quan es vol entrenar una xarxa per donar solució a un problema concret des de zero es necessita una quantitat d’informació que moltes vegades els investigadors no disposen. Per altre banda els filtres que

s'apliquen en les classificacions estan capacitats per reconèixer estructures que no són només pròpies d'un tipus d'imatge, sinó que es poden utilitzar aquests filtres de les primeres capes convolucionals provinents d'una altre CNN i només entrenar els últims nivells per tal de donar solució al problema concret.

Una aplicació més entenedora d'aquest fet es pot veure a l'article de MD. Zeiler et al. 2013 (Matthew D. Zeiler (2013)), on podem veure detalladament la dissecció d'una CNN des de les primeres capes. Per poder entrenar les capes posteriors es pot fer servir la funció "learnmodel.unfreeze()" de la llibreria Fastai (veure exemple en Quadre de Codi 5), que permet modificar els filtres de tal manera que quan fem "learnmodel.freeze()" després d'entrenar la CNN els patrons de reconeixement de les ultimes capes quedin establerts per reconèixer els patrons propis del nou problema.

El model inicial que s'ha seleccionat per adaptar-lo al problema que es tracta és l'anomenat "resnet34". L'article original on es planteja aquesta arquitectura és obra de K. He et al. i data del 2015. Es pot trobar publicat a l'arxiu web de la Universitat de Cornell i es cita en la bibliografia del present estudi(K. He, X. Zhang, S. Ren (2015)). Aquesta CNN té diferents versions entrenades segons el número de capes convolucionals que es vulgui utilitzar, l'únic factor limitant és la capacitat computacional de que es disposi, i és per això que s'ha seleccionat la versió de 34 capes. S'ha seleccionat després d'intentar millorar el resultat amb diferents arquitectures com "VGG" fins a altres versions de "ResNet" més complexes com "ResNext" o bé "ResNet" amb més capes convolucionals. Es mostra la versió que millor exactitud ha donat després de fer tota l'optimització pertinent, i també s'ha tingut en compte la velocitat i capacitat computacional ja que la versió "resnet50" presenta resultats similars a "resnet34" però la segona és molt més ràpida d'entrenar.

El codi Python del model es pot trobar en aquest enllaç (última visita dia 26/12/2018):

Enllaç a resnet.py

3.3.4 Fine Tuning

Per poder executar l'algoritme de manera pertinent a partir d'una arquitectura preexistent s'ha d'optimitzar el model a partir dels paràmetres d'aquest; "Fine Tuning" en anglès es podria traduir com a "optimització" en català però aquest és el terme utilitzat per la comunitat científica.

Per dur a terme el "fine tuning" s'ha executat diferents mètodes que han permès obtenir el millor resultat en altres estudis de CNN. Els mètodes utilitzats són la tria del millor LR mitjançant la biblioteca Fastai, "Data Augmentation", "Differential Learning Rate", "Test Time Augmentation", i "Cycling LR".

3.3.4.1 Optimitzant el LR

Triar el millor LR per començar a entrenar la CNN és el primer mètode que s'ha aplicat per optimitzar l'algoritme. Aplicar aquest mètode és molt simple mitjançant la llibreria Fastai de PyTorch, es pot fer aplicant la funció "learnmodel.lr_find()" i després accedint al gràfic:

El primer gràfic (figura 10) mostra com es va augmentant el LR per cada passada per les dades. El segon és el gràfic que interessa, i mostra com disminueix la "loss function" a mesura que augmenta el LR. Es fa interessant el fet d'agafar un LR que estigui en un moment proper al mínim del gràfic quan la "loss function" encara esta disminuint, per això s'ha seleccionat el LR = 0.08 per aquest model.

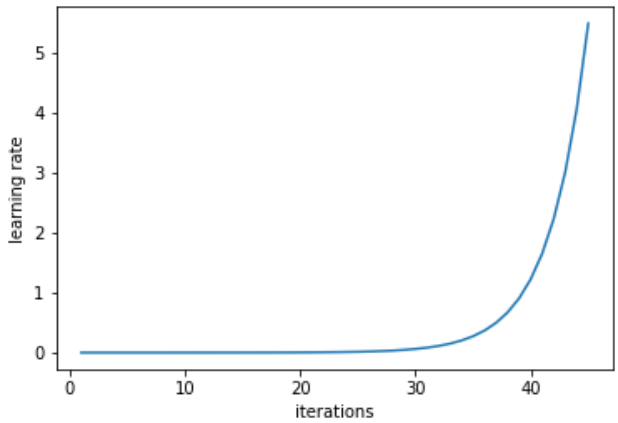

Figura 10: Gràfic on es pot veure com la funció lrfind va augmentant el LR en cada iteració.

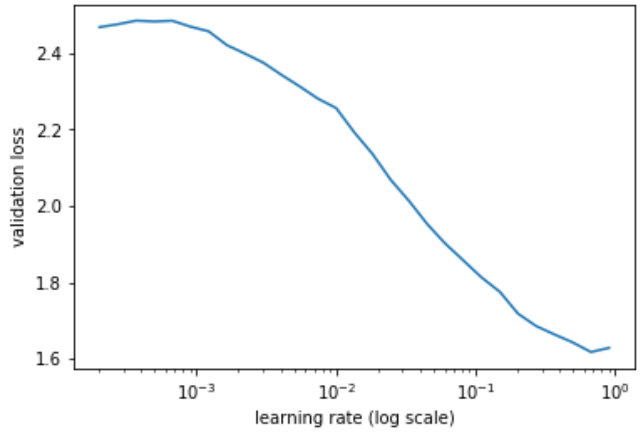

Figura 11: LR que permet obtenir un resultat de la loss function.

3.3.4.2 Data Augmentation

El següent pas ha estat el d'aplicar el concepte de "Data Augmentation" que consisteix en agafar cada imatge per entrenar la capa convolucional i fer diverses transformacions a cada imatge i el que s'aconsegueix aleshores és obtenir més d'una perspectiva de cada imatge, fet que disminueix el "overfitting" del model.

En l'exemple de la imatge extreta de les dades podem veure què passa amb la imatge quan s'aplica aquest mètode. Es pot observar la mateixa imatge en diferents versions en què s'ha invertit, s'ha rotat i s'ha ficat la imatge sota un augment de 110%.

La funció que ha permès aplicar aquest mètode la tenim escrita en Python en el quadre de codi 2.

```
tfms_from_model(arquitecturaCNN, midaImg, aug_tfms = transformacions, max_zoom = 1.1)
```

Quadre de Codi 2: Funció per aplicar Data Augmentation a les dades per entrenar el model.

Aquesta funció forma part de la llibreria de Fastai. Els primers arguments de la funció són irrellevants per fer "data augmentation", el que interessa és l'argument "aug_tfms" que permet indicar quina mena de

transformacions es volen aplicar a les imatges. I el "max_zoom" que correspon al "zoom out" que es pot aplicar a les imatges per intentar corregir-ne els marges.

Dins la categoria de procediments de "data augmentation" s'inclou el procediment de "test-time augmentation" o TTA, que és un procediment que consisteix a fer el mateix procediment fet a l'entrenament de la CNN però aquesta vegada amb la imatge que s'està classificant per veure si es pot millorar el resultat de la classificació, es dona l'etiqueta que sigui més freqüent en la classificació realitzada a cada imatge derivada de l'original. Aquest fet permet corregir un error molt comú en les aplicacions de DL en què els marges de les imatges tractades es veuen retallats deguts a la mecànica de processament de les imatges dels kernels, ja que l'última fila de píxels moltes vegades en queda afectada. També permet que la geometria de la imatge no interfereixi en la classificació, ja que disminueix l'impacte de la distribució dels elements dins la imatge.

El procediment TTA s'ha aplicat en l'últim moment de l'entrenament de la CNN, però no sempre optimitza el resultat final de l'exactitud del model.

```
log_preds,y = learnmodel.TTA()
```

Quadre de Codi 3: Funció per aplicar TTA amb la imatge que s'està classificant. Aquesta funció pertany a la llibreria de Fastai.

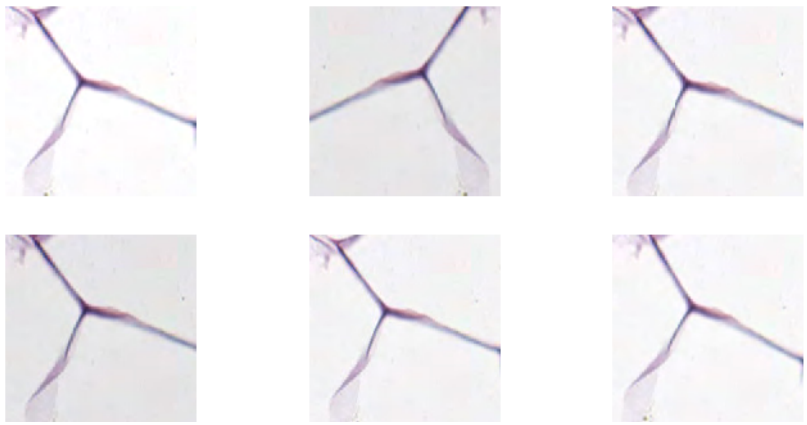

Figura 12: Transformacions realitzades durant el data augmentation amb una imatge de teixit adipós.

En l'exemple de la figura 12 es mostra una imatge de teixit adipós la qual se li ha aplicat el mètode de data augmentation. Podem observar les diferents transformacions que s'ha fet a la imatge. El procés de TTA consisteix en fer aquestes transformacions en el moment de la classificació de la imatge.

3.3.4.3 Cycling LR

Una vegada transformades les dades i entrenada la primera versió de la CNN es passa a fer el que s'anomena com LR cíclic (Cycling LR en anglès), que consisteix en disminuir el LR contínuament, i reiniciar-lo en el moment que es pugui haver trobat un mínim local de tal manera que es podrà saltar les barreres imposades per aquest mínim i augmentarà la possibilitat de trobar el mínim global. Podem veure una explicació més extensa del mètode en l'article de L. Smith de l'any 2017 (Smith (2017)).

Per aplicar aquest mètode es fa servir la funció de python que s'empra per començar a entrenar la CNN amb l'opció "cycle_len". Amb el paràmetre fixat en el quadre de codi 4 podem reiniciar el LR cada dues passades per les dades.

```
learn.fit(lr, 2, cycle_len = 2)
```

Quadre de Codi 4: Funció en python per començar a entrenar la CNN. Aquesta pertany a la llibreria de fastai. El lr és el Learning Rate, 2 fa referència al número d'èpoques o passades per les dades, i cycle-len fa referència al número d'èpoques que es vol incloure dins de cada cicle.

En la figura 13 podem veure l'exemple de gràfic del LR resultant del quadre de codi 4.

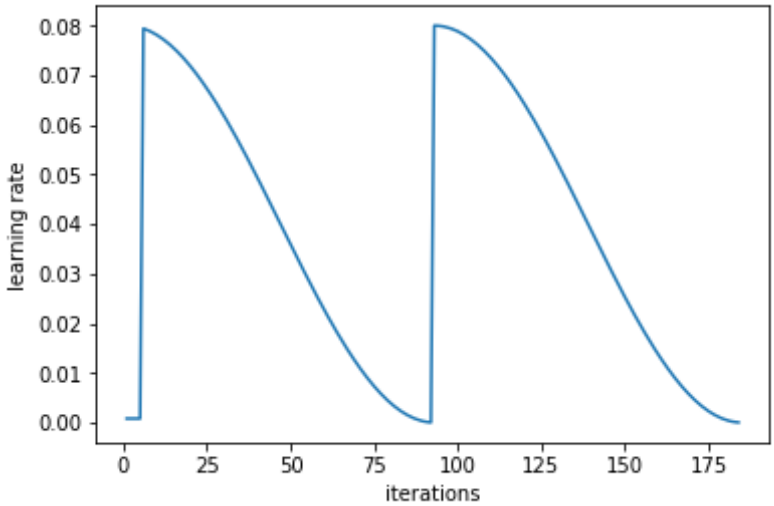

Figura 13: Figura il · lustrativa de la variació del LR durant l'aplicació del mètode de cycling LR.

3.3.4.4 Differential Learning Rate

Aquesta tècnica es proposa per un entrenament més eficient mitjançant la transferència del coneixement. La va proposar Jeremy Howard durant el transcurs del curs "Practical Deep Learning for coders, part 1, version 2" impartit al Data institute de la Universitat de San Francisco, contingut dins del conjunt de cursos anomenats Fastai.

Consisteix en aplicar un LR més baix en les primeres capes i augmentar-lo a mesura que anem entrenant les capes superiors ja que les primeres capes són més generals i requereixen menys aprenentatge, i a mesura que anem cap a les capes més profundes el coneixement ha de ser més intens i aquest fet es reflecteix aplicant aquest mètode. Aplicar aquesta funció és molt simple si s'empra la llibreria de Fastai per construir la CNN, ja que només li hem d'indicar més d'un LR i la funció automàticament busca la millor configuració per aplicar aquest setup.

```
learnmodel.unfreeze()
lrs = np.array([8e-4,8e-3,8e-2])
learnmodel.fit(lrs, 5, cycle_len = 1)
```

Quadre de Codi 5: Es comença per descongelar les capes de la CNN per poder-los modificar, es fixen 3 LR diferents i s'aplica la funció per començar a entrenar la CNN altre vegada però aquest cop aplicant la tècnica dels Differential Learning Rates.

Aquest seria un exemple d'aplicació de DLR amb codi python. Es comença descongelant les capes per poder modificar-ne la configuració i es fixen diferents LR en ordre ascendent dins un vector per aplicar-ho a l'entrenament. Finalment s'entrena el model mitjançant "learnmodel.fit()" incorporant el vector de LRs (figura 14).

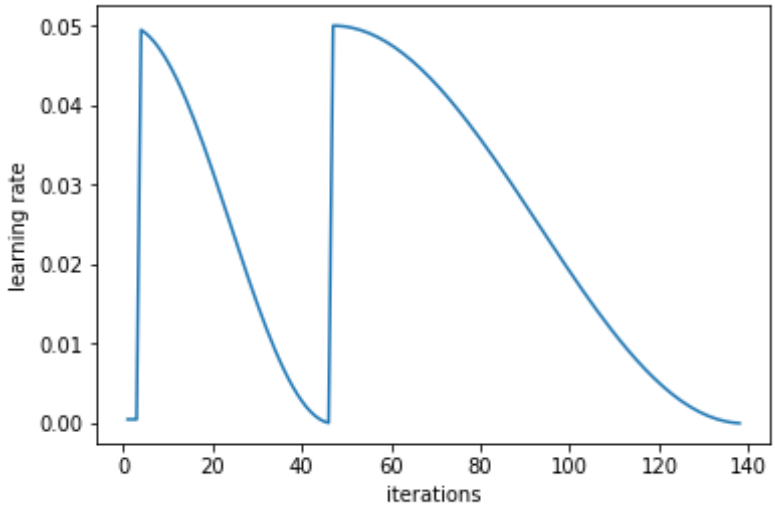

Figura 14: El gràfic del LR mostra com s'ha aplicat un LR diferent segons la epoch en que s'estigués treballant.

Fins aquí s'han recollit tots els procediments emprats per entrenar la CNN i millorar el paràmetre d'exactitud final de l'algoritme, que a continuació es farà servir per classificar parts de 150 × 150 píxels de WSI de 5000 × 5000 píxels. Primer s'ha provat diferents arquitectures de CNN que no han millorat el resultat, s'ha provat a entrenar-la començant amb imatges reformatejades a una mida més petita i tampoc s'ha millorat el paràmetre final, s'ha provat diferents mètodes de configuració del LR i també diferents mètodes per augmentar les dades i reduir el "overfitting". El resultat final queda una CNN amb una exactitud que oscil · la entre el 94%-95% d'exactitud amb el que ja podem passar a classificar les parts de les WSI.

3.4 Processament i classificació de WSI

Les WSI són imatges d'unes dimensions molt grans, arriben a pesar desenes de MB en 1 sola imatge i això es deu a la seva alta resolució. Aquestes imatges engloben una gran regió de teixit i tenen tota la informació enregistrada en 5000 × 5000píxels. Són cada vegada més utilitzades en aplicacions mèdiques, ja que són compatibles amb aplicacions digitals (Evans et al. (2014)) i permeten l'accés a la informació de formes molt innovadores (Pantanowitz, Farahani, and Parwani (2015)). Aquesta aplicació en concret té ús dins el camp de la histopatologia, que és el camp que tracta les malalties des de la perspectiva de la histologia.

En el present estudi es planteja l'ús d'aquestes imatges per buscar una aplicació de la CNN que s'ha entrenat dins el sector mèdic i es busca replicar els resultats de l'estudi de JN Kather (J. N. Kather et al. (2016)), a la vegada que s'aplica la classificació de les imatges amb una exactitud més acurada que la obtinguda en el moment de l'estudi l'any 2016. Per processar les imatges s'ha fet un procés de fragmentació per poder-les classificar mitjançant la CNN que s'ha entrenat per classificar imatges de 150 × 150 en 8 classes de teixit diferents.

Després de la fragmentació s'ha obtingut 1156 imatges de cada una de les 10 WSI que s'han col · locat en una carpeta per a cada WSI. Posteriorment s'ha classificat cada una de les imatges segons la informació continguda en el model de la CNN que s'ha entrenat al començament de l'estudi. S'ha guardat l'etiqueta de cada imatge i s'ha aplicat la funció que es mostra en el quadre de codi inferior per donar color a la imatge segons un canal RGB que li dona color segons el tipus d'etiqueta assignada a la imatge.

```
def pintar(width, height):
    negre = (0,0,0)
    gris = (120,120,120)
    blanc = (1000,1000,1000)
    lila = (150,20,147)
    blau = (23,140,190)
    groc = (240,220,20)
    verd = (0,210,10)
    tronja = (240,120,30)
    for x in range(width):
        for y in range(height):
            if prediccions[a] == 'empty': píxels[x, y] = negre
            if prediccions[a] == 'adipose': píxels[x, y] = gris
            if prediccions[a] == 'mucosa': píxels[x, y] = blanc
            if prediccions[a] == 'debris': píxels[x, y] = lila
            if prediccions[a] == 'complex': píxels[x, y] = groc
            if prediccions[a] == 'stroma': píxels[x, y] = verd
            if prediccions[a] == 'tumor': píxels[x, y] = tronja
            if prediccions[a] == 'lympho': píxels[x, y] = blau
```

Quadre de Codi 6: Funció definida per pintar els píxels de la imatge tractada segonsl'etiqueta assignada per l'algoritme. S'han definit 8 colors mitjançant un canal decolor RGB els quals s'assignen depenent de l'etiqueta emmagatzemada a "prediccions".

La funció "pintar" escrita en python s'ha utilitzat per donar color a cada part de la WSI que s'estava tractant. Es té en compte l'etiqueta assignada per la CNN per assignar un dels 8 colors:

- Estroma Simple - classificat com a "stroma" es pinta de color verd.
- Estroma Complexe - classificat com a "complex" es pinta de color groc.
- Teixit del SI - classificat com a "lympho" es pinta de color blau.
- Teixit Adipós - classificat com a "adipose" es pinta de color gris.
- Mucosa - classificat com a "debris" es pinta de color lila.
- Teixit Glandular - classificat com a "mucosa" es pinta de color blanc.
- Epiteli cancerós - classificat com a "tumor" es pinta de color tronja.
- Fons sense teixit - classificat com a "empty" es pinta de color negre.

S'ha volgut mantenir al màxim possible els colors utilitzats per l'article de JN Kather de manera que es pugui comparar els resultats amb més facilitat.

En el moment de fragmentar la WSI a cada part de la imatge s'ha assignat un nom en funció de la localització del fragment dins la imatge original en el format "altura_amplada.jpg" indicant la posició dels píxels (Quadre de Codi 7). Després de classificar els fragments de WSI i pintar els quadres del color pertinent s'ha reconstruït les WSI mitjançant el nom del fragment per recol · locar-lo en la posició que li correspon en la imatge original.

```
def image_slice(image_path, outdir, sliceHeight, sliceWidth):
    img = Image.open(image_path)
    imageWidth, imageHeight = img.size
    left = 0
    upper = 0
```

```
    while (left < imageWidth):
        while (upper < imageHeight):
            if (upper + sliceHeight > imageHeight and \
                left + sliceWidth > imageWidth):
                bbox = (left, upper, imageWidth, imageHeight)
            elif (left + sliceWidth > imageWidth):
                bbox = (left, upper, imageWidth, upper + sliceHeight)
            elif (upper + sliceHeight > imageHeight):
                bbox = (left, upper, left + sliceWidth, imageHeight)
            else:
                bbox = (left, upper, left + sliceWidth, upper + sliceHeight)
            working_slice = img.crop(bbox)
            working_slice.save(os.path.join(outdir, str(upper) + '_' + str(left) + '.jpg'))
            upper += sliceHeight
        left += sliceWidth
        upper = 0
```

Quadre de Codi 7: Es construeix la funció que permetrà fragmentar les WSI en imatges en fragments de la mida que s'escaigui, en aquest cas de 150 píxels quadrats.

```
if __name__ == '__main__':
    for subdir, dirs, files in os.walk(workingdir):
        for file in files:
            image_slice(subdir + '/' + file, subdir, 150, 150)
```

Quadre de Codi 8: S'aplica la funció que s'ha indicat en el quadre de codi 7 amb les mides necessàries i a tots els arxius que hi ha als directoris de les dades, és a dir, es fragmenten les 10 WSI en 1156 fragments cada una.

S'ha construït un script de python que s'ha emprat per reconstruir cada una de les WSI amb els quadres de color corresponents al tipus de teixit (Quadre de Codi 9). S'utilitzen dos bucles "while" que busquen davant i darrera del "_"del nom del fragment de les WSI i guarden les coordenades del fragment. Posteriorment s'empra la funció "imatge.paste()" per enganxar el quadre en la posició corresponent en format de píxel com tot el procediment.

```
for g in range(len(files_to_move)):
    imagen = Image.open(workingdir + files_to_move[g])
    slices = !ls {workingdir + str(g + 1) + '/'}
    for i in range(len(slices)):
        j = 0
        k = 1
        vertical = str()
        horitzontal = str()
        while slices[i][j] != "_":
            vertical = vertical + slices[i][j]
            j = j + 1
        while slices[i][len(vertical)+k] != ".":
            horitzontal = horitzontal+slices[i][len(vertical)+k]
            k = k + 1
        sliced = Image.open(workingdir + str(g + 1) + '/' + vertical + "_"+horitzontal + '.jpg')
        imagen.paste(sliced,(int(horitzontal),int(vertical)))
    imagen.save(workingdir + "class_" + files_to_move[g])
```

Quadre de Codi 9: Funció escrita en python per reconstruir les WSI a partir dels fragments classificats. S'utilitzen els bucles "for"per navegar per tots els fragments de les WSI i comprovant el nom de cada fragment s'incorpora cadascun a la posició que té enregistrada en el nom, respecte la imatge original.

Les WSI resultants tenen les regions de la imatge original del color que correspon a cadascun dels 8 tipus de teixit. Ara bé, les imatges resultants tenen entre un 6-5% d'errors i tenen una resolució molt baixa, ja que s'ha classificat regions de teixit de 150 píxels quadrats (apartat de resultats).

Per poder entendre les imatges generades s'ha seleccionat el mètode anomenat com a "alpha blending" per generar una imatge composta amb la original i la classificada (Quadre de Codi 10). Aquest mètode consisteix en agafar una imatge com a base d'una altre imatge (imatge alpha) per tal de generar una nova imatge composta. Aquest procediment permetrà veure les regions que han estat classificades per l'algoritme dins les imatges originals.

```
for i in range(len(files_to_move)):
    image_WSI = Image.open(workingdir + f'{files_to_move[i]}')
    image_WSI = image_WSI.convert("LA")
    image_WSI = image_WSI.convert("RGBA")
    image_base = Image.open(workingdir + "class_" + f'{files_to_move[i]}')
    image_base = image_base.convert("RGBA")
    alphaComposited = Image.alpha_composite(image_base, image_WSI)
    alphaBlended = Image.blend(image_WSI, image_base,.6)
    alphaBlended.save(workingdir + "alpha_blended_" + f'{files_to_move[i]}')
```

Quadre de Codi 10: Funció que obre cada parella d'imatges, la original i laclassificada, les converteix en el mateix tipus de colors i posteriormentles mescla mitjançant el procediment "alpha blending"posant la imatge classificadacom a base de l'original en blanc i negre.

4 Resultats

Aquest treball engloba dos grans blocs que defineixen tota la feina realitzada durant l'estudi. El primer és el bloc d'entrenament de la CNN, que finalitza amb la classificació del test set per comprovar l'efectivitat de l'algoritme amb unes dades que aquest no haurà tractat en cap moment. El segon bloc engloba tot el tractament i classificació d'imatges WSI.

4.1 Entrenament de la CNN

El bloc de l'entrenament de la CNN es vol fer més transparent mostrant tots els resultats que han resultat després de cada procediment indicat en la part de funcionament de l'algoritme mitjançant l'arquitectura resnet34. Aquesta ha estat la que ha donat millor resultat de totes les que s'ha assajat. Es mostraran els resultats de cada part amb imatges il · lustratives de tot el procés i com a part final es farà un resum del desenvolupament de diferents arquitectures. S'ha seguit els mateixos procediments per entrenar la CNN amb totes les arquitectures. Finalment s'ha escollit resnet34 com a punt de partida per entrenar l'algoritme degut a que és la que més "accuracy" té i més ràpid es pot entrenar.

```
data = ImageClassifierData.from_paths(PATH, tfms=tfms)
learn = ConvLearner.pretrained(arch, data)
```

Quadre de Codi 11: Es defineixen les variables learn i data que permetran guardar la informació de la CNN. learn fa referència als filtres de les capes convolucionals, i data fa referència a les dades amb les que s'estan entrenant els filtres.

Es comença fent una comprovació per veure el LR òptim per entrenar la CNN (figura 11). S'ha decidit que el paràmetre LR òptim per entrenar-la és de 0.08 ja que s'ha d'escollir un LR en un moment on la loss function encara estigui disminuint, fet que indica una millora en el resultat. Un cop s'ha decidit el LR s'ha començat a entrenar la CNN. Les variables "learn" i "data" són les que guardaran els filtres de les capes convolucionals i les dades amb les transformacions pertinents respectivament (Quadre de Codi 11)

```
lr = 0.08
learn.fit(lr, 5)
```

Quadre de Codi 12: Codi python del 1r Entrenament de la CNN. Es fixa el LRi el número d'epochs.

Epoch	trn-loss	val-loss	accuracy
0	0.721692	0.596212	0.843718
1	0.572319	0.383981	0.867211
2	0.488973	0.42201	0.87334
3	0.414638	0.32847	0.891726
4	0.374729	0.328249	0.88764

Taula 2: Procés realitzat en el 1r entrenament de la CNN mitjançant el codi del quadre de codi 12.

El primer pas en entrenar la CNN ha estat un procés de 5 epochs amb el $LR = 0.08$ (taula 2). Això vol dir que s'ha fet un total de 5 passades per les dades per entrenar els filtres de les capes convolucionals. S'ha escollit el paràmetre de 5 epochs ja que sempre es vol entrenar fins al punt anterior a que succeeixi el overfitting. El resultat d'aquest procés d'entrenament ha estat 0.88 d'exactitud sobre 1.

Tot seguit s'ha continuat amb l'entrenament i s'ha decidit realitzar el mètode de data augmentation degut a que es vol poder entrenar amb més passades per les dades sense arribar a fer overfitting. Per això es fixa el paràmetre "cycle_len = 2" i les transformacions a les dades es fixen amb la variable tfms (Quadre de Codi 13). El paràmetre "cycle_len" permet aplicar també el mètode del cycling LR en aquest mateix pas.

```
tfms = tfms_from_model(resnet34, sz, aug_tfms=transforms_side_on, max_zoom=1.1)
learn.fit(lr, 2, cycle_len = 2)
```

Quadre de Codi 13: tfms és la variable que guarda les transformacions que es volen fer a les dades, amb ella es fixen el tipus de data augmentation que es vol fer, ja que es fixa aug-tfms per decidir les variacions en les imatges i el max-zoom per indicar el zoom out màxim que es pot aplicar a la imatge.

Epoch	trn-loss	val-loss	accuracy
0	0.324609	0.305563	0.896834
1	0.272232	0.27528	0.907048
2	0.263446	0.283403	0.911134
3	0.250688	0.276794	0.911134

Taula 3: Procés realitzat en el 2n entrenament de la CNN mitjançant el codi del quadre de codi 13.

S'ha fet que es reinicii el LR 1 vegada conjuntament amb data augmentation (taula 3).

```
learn.unfreeze()
lrs = np.array([8e-4,8e-3,8e-2])
learn.fit(lrs, 3, cycle_len=1)
```

Quadre de Codi 14: Es comença descongelant els filtres guardats de les capes convolucionals per tenir la capacitat de modificar-los. La variable lrs guarda un vector que conté els 3 LRs que s'ha seleccionat, cada un manté una relació de 1/10 amb l'anterior.

Epoch	trn-loss	val-loss	accuracy
0	0.42692	0.24534	0.917263
1	0.293706	0.224261	0.92952
2	0.207025	0.174469	0.951992

Taula 4: Aplicació del Differential Learning Rate mitjançant el quadre de codi 14.

Amb l'últim mètode s'ha aconseguit arribar a una exactitud final de 0.9520. Però encara falta per aplicar un últim mètode a veure si permet millorar el paràmetre final. Es tracta del TTA, un mètode que serveix sobretot per evitar que la distribució dels elements dins la imatge afecti al resultat de la classificació. En el cas d'imatges histològiques és previsible que influeixi poc ja que els elements en una imatge d'aquestes característiques té un caos que permet que la distribució dels elements tingui poca influència en l'algoritme. El resultat pel que fa l'exactitud d'aquest mètode ha donat un resultat de 0.9418 que fa empitjorar el resultat del mètode anterior. Aquest fet confirma el que ja s'esperava. De totes maneres es creu que l'aplicació d'aquest mètode es fa interessant degut a que és difícil saber a priori si la distribució en la imatge afectarà la decisió final.

Després d'aplicar tot l'entrenament a la CNN s'ha elaborat una matriu de confusió per veure quines són les classes en què l'algoritme ha acumulat més errors. Les classes amb més errors de classificació han estat complex i stroma, seguides de lympho. Si s'observa atentament la graella podem veure com els errors sempre es presenten entre les mateixes classes, de manera que s'hauria de buscar mètodes per solucionar la confusió entre classes que segurament tenen morfologies tissulars similars (figura 15). El problema potser requereix més imatges per entrenar i permetre l'algoritme a reconèixer de manera eficaç aquestes classes. Això no és possible de desenvolupar en el present estudi degut a que no es disposa de la informació ni l'accés a la informació necessari per solucionar aquest problema.

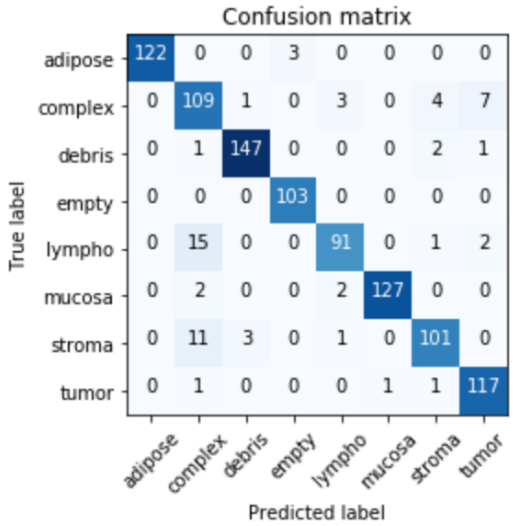

Figura 15: Matriu de confusió resultant de la classificació.

4.1.1 Confirmació de resultats: test set

Una vegada s'ha entrenat la CNN es fa necessari comprovar la seva efectivitat amb unes dades que encara no hagi vist. Per realitzar aquesta funció s'ha preparat un conjunt de dades anomenat test set, que va a part dels conjunts de dades utilitzats per entrenar l'algoritme. La configuració de les carpetes és la mateixa que en el validation test i en el training test, per tant es contrasta el nom de la carpeta amb l'etiqueta assignada a cada imatge per veure si s'està classificant correctament. El codi per executar aquest procés està recollit en el Quadre de Codi 15.

```
trn_tfms, val_tfms = tfms
error_count=0
total_files=0
for i in range(len(data.classes)):
    workingdir = "/home/deak/fastai/data/colorectal_cancer/test/"
    workingdir = workingdir + data.classes[i] + "/"
    files = os.listdir(workingdir)
    total_files = total_files + len(files)
    for a in range(len(files)):
        im = val_tfms(open_image(workingdir + files[a]))
        pred = learn.predict_array(im[None])
        if (learn.data.classes[np.argmax(pred)] == data.classes[i]) == False:
            error_count=error_count+1
total_acc= 1- error_count/total_files
print(total_acc)
```

Quadre de Codi 15: Codi Python emprat per calcular l'exactitud amb el conjunt de dades test set.

L'exactitud obtinguda d'aquest procés ha estat de 94.75%, molt semblant a l'obtinguda per l'algoritme amb el validation set. Després d'això es considera que l'algoritme ha estat correctament entrenat i és possible passar a classificar les regions de les WSI.

4.1.2 Resultats d'altres arquitectures

L'arquitectura resnet34 s'ha escollit com a punt de partida després de provar diferents arquitectures de CNN per intentar millorar el paràmetre d'exactitud. Les arquitectures que s'han assajat són diferents variants de VGG i també diferents variants de resnet i resnext, que és una versió millorada de resnet.

Amb la VGG16 s'ha arribat a un percentatge d'exactitud de 90.39%. No s'ha pogut arribar a aplicar el mètode de differential LR ja que la memòria de l'ordinador no ha permès anar més endavant amb el procediment. Aquest entrenament no ha estat tant llarg com els de les arquitectures que veurem a continuació, ha tingut una durada de 1 minut i mig sense aplicar el differential LR.

Després s'ha provat una altre versió de la VGG, la VGG16-BN, la qual directament no s'ha pogut entrenar degut a una falta de memòria. Probablement s'hauria d'ampliar la memòria de l'ordinador o accedir a una GPU externa per tal d'intentar aplicar una millora al mètode.

Com a última VGG s'ha provat la versió de VGG19, la qual ha reportat un resultat de 90.1% molt similar a la versió de VGG16. La diferència de temps en l'entrenament respecte la seva versió més simple no aporta una millora en els resultats finals.

Aleshores s'ha començat a provar les versions de resnet, amb resnet34 com a primera prova. El resultat ha estat el millor obtingut amb les diferents arquitectures que s'ha provat. S'ha pogut aplicar tots els mètodes de millora de l'exactitud final i tot en un temps rècord de 2 minuts i mig. El resultat final ha estat una exactitud d'entre 94 i 95%, ja que aquesta versió s'ha entrenat moltes vegades amb configuracions de les dades diferents per veure si els resultats eren reproduïbles.

La següent versió de resnet a ser entrenada ha estat la de resnet50, que també s'ha pogut entrenar fins al final. El temps d'entrenament d'aquesta versió ha estat de 4 minuts i mig, 2 minuts més que en la versió de 34 capes convolucionals. El resultat de l'exactitud final ha estat molt similar a l'obtingut amb resnet34, amb un valor del paràmetre de 94.5%.

L'última versió que s'ha incorporat a l'estudi ha estat la de 101 capes convolucionals. A aquesta no se li ha pogut aplicar el mètode del differential LR. El temps d'entrenament era molt superior ja que la versió de 50 capes s'ha trigat 2 minuts en entrenar fins al punt en què s'aplicava el differential LR i en aquest cas s'ha trigat uns 3 minuts en aplicar els mateixos procediments. En canvi l'exactitud en aquest punt era superior que en les dues versions entrenades de resnet, s'ha arribat al llindar de 92% d'exactitud sense haver aplicat el mètode de differential LR. Aquest últim fet indica com el fet d'augmentar el nombre de capes convolucionals amb resnet millora el resultat del paràmetre final, en canvi no succeeix el mateix amb l'arquitectura VGG.

4.2 Classificació de WSI

Una vegada entrenat l'algoritme que permet classificar els tipus de teixits continguts en les WSI s'ha procedit a fer el tractament i classificació de les imatges mitjançant la CNN que s'ha descrit fins ara. A continuació es mostra cada imatge conjuntament amb la seva versió original i la versió classificada. La qualitat de les imatges classificades és baixa degut a la falta d'un tractament posterior de les dades adient. No es disposa de les eines de manipulació d'imatges adients per obtenir un resultat més clar i per això s'ha presentat els resultats mitjançant l'alpha blending. S'ha aplicat un procés d'interpol · lació bicúbica i un suavitzat de les imatges mitjançant matrius de 3 * 3 però no s'ha aconseguit el resultat esperat, el codi python d'aquesta part es presenta a l'apartat d'annexos.

Es mostra primer la imatge original, després la classificada i per últim la que les mescla mitjançant el canal alpha.

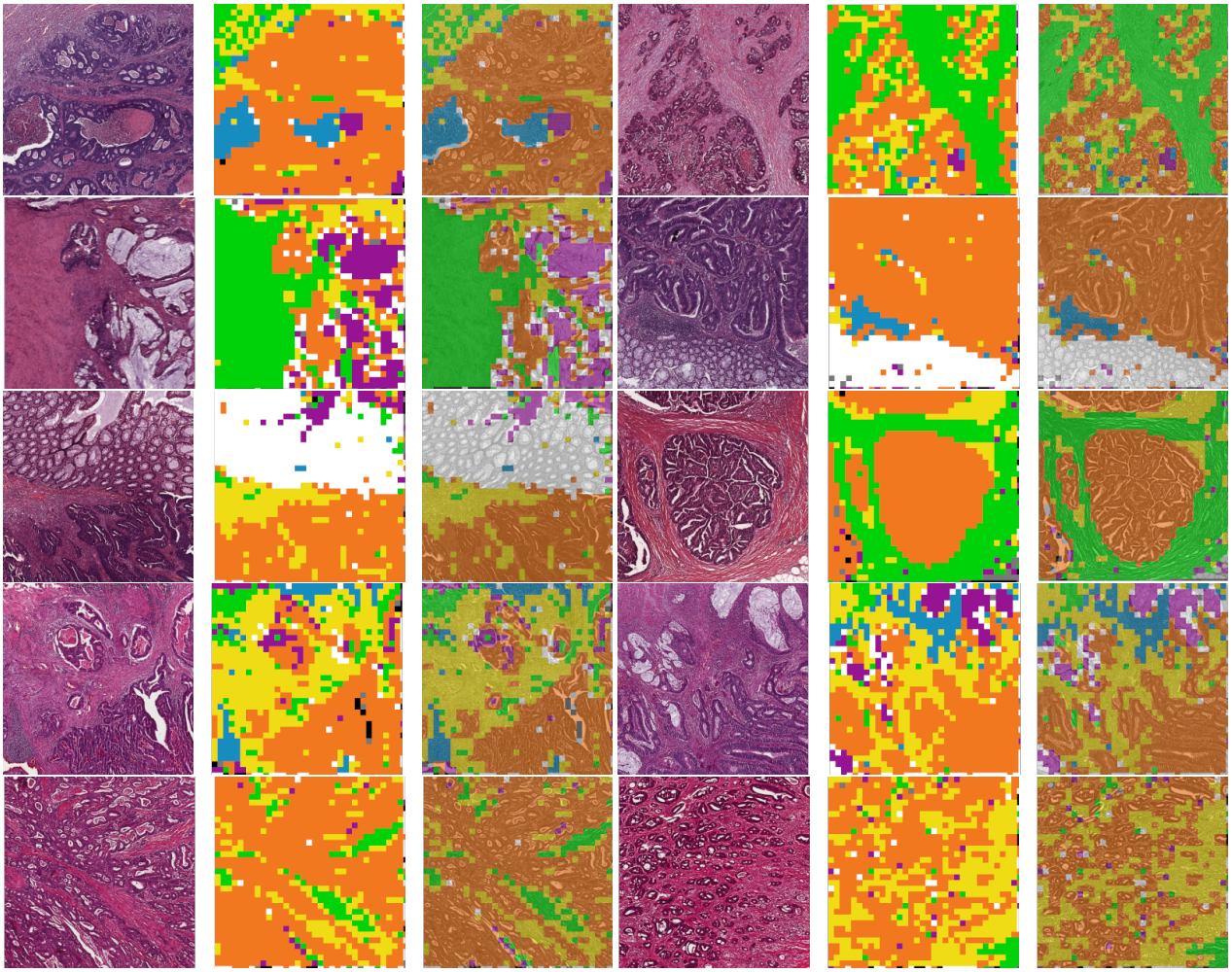

Figura 16: WSI classificades mitjançant una CNN entrenada per detectar 8 tipus diferents de teixit.

- Estroma Simple - classificat com a “stroma” es pinta de color verd.
- Estroma Complexe - classificat com a “complex” es pinta de color groc.
- Teixit del SI - classificat com a “lympho” es pinta de color blau.
- Teixit Adipós - classificat com a “adipose” es pinta de color gris.
- Mucosa - classificat com a “debris” es pinta de color lila.
- Teixit Glandular - classificat com a “mucosa” es pinta de color blanc.
- Epiteli cancerós - classificat com a “tumor” es pinta de color tronja.
- Fons sense teixit - classificat com a “empty” es pinta de color negre.

Enllaç a la versió completa de les imatges: drive.google.com

Les imatges s’han pujat en un repositori de Google Drive per fer-les accessibles en format complert per a tothom. En aquell repositori les imatges es troben tal com s’ha obtingut els resultats de l’estudi. Els resultats aquí mostrats s’han hagut de tractar per tal de poder-los mostrar en l’article.

Les imatges classificades poden contenir alguns errors, al voltant d’un 5-6%, però en general es considera que s’ha obtingut una classificació prou ajustada tenint en compte que els resultats obtinguts en l’article de JN

Kather (J. N. Kather et al. (2016)). En aquell article no es mostra tota la informació sobre el tractament de les imatges i per tant ha faltat millorar la qualitat de les WSI finals un cop han estat classificades. El fet de treballar amb molts fragments de la imatge permet extrapolar alguns errors, ja que l'error de classificació està relacionat amb la imatge original i no és només un càlcul puntual sobre una decisió que no es pot contrastar.

5 Conclusions

El resultat final de l'estudi ha estat obtenir una CNN entrenada capaç de classificar 8 tipus de teixits en WSI obtingudes de biòpsies de pacients amb casos reals de CRC. Ja existien casos com el que s'ha entrenat però en el present estudi s'ha millorat el resultat obtingut fins ara en cap altre estudi que s'hagi explorat per comparar resultats. En l'estudi de JN Kather el resultat final era d'un 89% d'exactitud aproximadament (J. N. Kather et al. (2016)), i en l'estudi realitzat per l'exestudiant del máster en Bioinformàtica i Estadística de la UOC John Marturet Rodrigo l'any 2018, on s'arribava fins a un 93% en el cas més complex i sense classificació de WSI. Nosaltres hem millorat el resultat final de l'exactitud de l'algoritme i hem esbrinat camins que poden millorar el desenvolupament de la CNN més enllà del resultat que hem obtingut.

Els tractaments del càncer colorrectal actualment consisteixen en fer una observació de les imatges histològiques a mans d'un metge especialitzat. La visió d'un especialista no deixa de ser una interpretació basada en les dades que proporcionen les imatges la qual pot ometre informació important i augmentar el número de falços negatius en la detecció del càncer en les imatges. El nostre mètode proporciona una fiabilitat per sobre de les capacitats de l'ull humà ja que detecta directament patrons pròpis de cada teixit basant-se en les relacions entre píxels, en una escala de detecció més presisa. Aquest algoritme també facilita l'accés a la informació degut al seu procés d'automatització ja que la seva aplicació només depèn de tenir l'equipament i configuració adients, pot ser una eina molt eficaç per ajudar als metges especialitzats més que un substitut del tractament actual.

La potència de computació és determinant del resultat final obtingut en entrenar la CNN, i es considera la utilització d'ordinadors més potents com a eina imprescindible per aplicar una solució com la que aquí s'està tractant. En cas de disposar d'ordinadors més potents es considera l'opció de l'arquitectura resnet101 com la més factible, ja que en aplicacions mèdiques l'exactitud és un element clau per la viabilitat en l'aplicació d'un nou tractament i el temps d'entrenament de la CNN no és un factor clau a tenir en compte.

És possible automatitzar tractaments mèdics mitjançant eines d'intel · ligència artificial. L'eficàcia d'aquests dependrà de la capacitat de computació que tingui el centre on es vulgui aplicar el tractament. Fins i tot es planteja la creació d'una plataforma web per proporcionar aquesta mena de serveis a tots els hospitals que es puguin beneficiar de la tecnologia. La tecnologia serà més eficaç com més dedicació es fiqui en augmentar la qualitat de les imatges finals, per exemple entrenant la CNN per poder classificar fragments més petits de les WSI.

El resultat final de la classificació de les WSI ha estat un èxit. L'algoritme ha estat capaç de classificar les regions delimitades en la imatge inicial segons els 8 tipus de teixit pels que s'ha entrenat l'algoritme. Per tant es conclou l'estudi amb uns resultats satisfactoris i amb expectativa d'ampliació de la tecnologia de cara al futur cap a mètodes amb un marge d'error més petit. Si bé els tractaments mèdics sempre depenen de sensors que detectin un quadre clínic, ja sigui el metge o un aparell, l'automatització del tractament de les dades obra la porta a desenvolupar noves tecnologies en l'àmbit de la medicina personalitzada. És important buscar noves aplicacions del Machine Learning en el sector mèdic per tal de potenciar aquest fet.

6 Glossari

- Càncer Colorrectal (CRC): Malaltia producte de l'envelliment que afecta el colon i el recte. Les cèl · lules es divideixen de manera descontrolada invadint teixits i funcions fisiològiques.
- Whole-Slided Images (WSI): Tipus d'imatges d'alta resolució. Les d'aquest estudi són imatges de biòpsies de pacients reals amb càncer colorrectal.
- Machine Learning (ML): Ciència que estudia els mètodes d'aprenentatge de les màquines.
- Deep Learning (DL): Tecnologia que s'engloba dins el Machine Learning i que comprèn tots aquells mètodes elaborats a partir de xarxes neuronals profundes.
- Xarxa Neuronal Convolucional (CNN en anglès): Aplicació del Deep Learning en què s'entrena una xarxa neuronal per processar la informació mitjançant capes anomenades capes convolucionals.
- Natural Lenguage processing (NLP): Ciència inclosa dins el Machine Learning que consisteix en l'estudi dels mètodes que permetrien a les màquines entendre el llenguatge natural humà amb les seves pròpies eines.
- Overfitting: Procés que es dona durant la fase d'entrenament de la CNN en el que aquesta està adaptant els seus filtres de manera massa ajustada amb les dades del training set.
- Underfitting: És el procés invers a l'anterior. L'algoritme no ha adaptat els seus filtres suficientment a les dades i no són viables per solucionar el problema.
- Accuracy (Acc): Paràmetre que s'empra per mesurar el desenvolupament final de l'algoritme. Es dedueix a partir dels errors totals comesos per la CNN.
- Loss Function (LF): Equació emprada per esbrinar el grau de desajustament entre la decisió presa per l'algoritme i la classe real de quelcom s'estigui analitzant. Aquesta forma part de les equacions de cost function.
- Epoch: 1 passada per totes les dades.
- Learning Rate (LR): Paràmetre que estableix la distància entre els passos realitzats durant l'entrenament de la CNN.
- Gradient Descent (GD): Mètode en què es busca el mínim valor d'una cost function. S'empra en l'entrenament d'una CNN per buscar els valors òptims dels filtres de les capes convolucionals.
- Stochastic Gradient Descent (SGD): Variació del GD en què s'utilitza una sentència aleatòria de les dades com a punt de partida. Aquest mètode permet estalviar memòria a l'ordinador i disminuir el temps d'entrenament de la CNN.
- Differential Learning Rate (DLR): Mètode en què es proven diferents LR de més petit a més gran en un mateix procés d'entrenament.
- Data Augmentation (DA): Mètode que permet obtenir diferents versions d'una mateixa imatge. És útil en l'entrenament de les CNNs ja que disminueix el grau d'overfitting.
- Kernel: Matriu que permet processar les dades d'un punt concret a partir de la interacció amb els punts del seu entorn. S'utilitzen per processar els valors de les capes convolucionals.
- Sistema Immune (SI): Sistema cel · lular que garanteix l'estabilitat de l'organisme a llarg plaç. Està compost per més d'un tipus cel · lular i impedeix que agents exògens invadeixin l'organisme.
- Fine Tuning: Procés mitjançant el qual es busca la millor configuració de la CNN per desenvolupar una tasca concreta.
- Graffic Unit Processing (GPU): Unitat de hardware d'un ordinador que permet processar dades d'imatges de manera molt més eficient que una CPU.

- Red - Green - Blue (RGB): Configuració de color que permet accedir a una gama de colors determinada. És un estandar en la tecnologia del color. Àmpliament utilitzat en el processament d'imatges.

7 Annex

En aquesta secció s'inclou tot aquell codi de python que no s'ha pogut incloure en el cos de la memòria per raó de practicitat. Es mostra tot el codi executat per ordre amb un petit títol de la tasca que aquest desenvolupa.

7.1 Part introductòria

En aquesta part es fixen els paràmetres inicials amb què es treballarà. S'importen tots els paquets necessaris i es fixa el directori de treball. També es fixa les transformacions a realitzar a les dades i l'arquitectura que s'empra com a punt de partida.

```
%reload_ext autoreload
%autoreload 2
%matplotlib inline
from fastai.imports import *
  from fastai.transforms import *
from fastai.conv_learner import *
from fastai.model import *
from fastai.dataset import *
from fastai.sgdr import *
from fastai.plots import *
from PIL import Image
import os
import cv2
import numpy as np
```

```
PATH_TO_DATA="/home/deak/fastai/data/"
PATH = PATH_TO_DATA+"colorectal_cancer/"
```

```
def get_data(sz):
    tfms = tfms_from_model(resnet34, sz, aug_tfms=transforms_side_on, max_zoom=1.1)
    return ImageClassifierData.from_paths(PATH, tfms=tfms_from_model(arch, sz))
```

```
sz=224
arch=resnet34
data = get_data(sz)
data.classes
```

7.2 Escollint un LR

```
learn = ConvLearner.pretrained(arch, data)
lrf=learn.lr_find()
```

```
learn.sched.plot_lr()
LR= learn.sched.plot()
```

7.3 Comencem a entrenar la CNN i apliquem DA

```
tfms = tfms_from_model(resnet34, sz, aug_tfms=transforms_side_on, max_zoom=1.1)
```

```
data = ImageClassifierData.from_paths(PATH, tfms=tfms)
learn = ConvLearner.pretrained(arch, data)
```

```
lr=0.08
learn.fit(lr, 5)
```

```
learn.precompute=False #Això permet aplicar DA
```

```
learn.fit(lr, 2, cycle_len=2)
learn.sched.plot_lr()
learn.save('224_lastlayer')
learn.load('224_lastlayer')
```

7.4 S'aplica Differential LR

```
learn.unfreeze()
lrs=np.array([8e-4,8e-3,8e-2])
learn.fit(lrs, 5, cycle_len=1)
learn.sched.plot_lr()
```

7.5 S'aplica TTA

```
log_preds,y = learn.TTA()
probs = np.mean(np.exp(log_preds),0)
accuracy_np(probs, y)
```

7.6 Es prova el test set

```
trn_tfms, val_tfms = tfms
error_count=0
total_files=0
for i in range(len(data.classes)):
    workingdir = "/home/deak/fastai/data/colorectal_cancer/test/"
    workingdir = workingdir + data.classes[i] + "/"
    files = os.listdir(workingdir)
    total_files = total_files + len(files)
    for a in range(len(files)):
        im = val_tfms(open_image(workingdir + files[a]))
        pred = learn.predict_array(im[None])
        if (learn.data.classes[np.argmax(pred)] == data.classes[i]) == False:
            error_count=error_count+1
total_acc= 1- error_count/total_files
print(total_acc)
```

7.7 Analitzant els resultats

```
preds = np.argmax(probs, axis=1)
probs = probs[:,1]
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y, preds)
```

```
plot_confusion_matrix(cm, data.classes)
```

7.8 Fragmentar WSI

```
def image_slice(image_path, outdir, sliceHeight, sliceWidth):
    img = Image.open(image_path)
    imageWidth, imageHeight = img.size
    left = 0
    upper = 0
    while (left < imageWidth):
        while (upper < imageHeight):
            if (upper + sliceHeight > imageHeight and \
                left + sliceWidth > imageWidth):
                bbox = (left, upper, imageWidth, imageHeight)
            elif (left + sliceWidth > imageWidth):
                bbox = (left, upper, imageWidth, upper + sliceHeight)
            elif (upper + sliceHeight > imageHeight):
                bbox = (left, upper, left + sliceWidth, imageHeight)
            else:
                bbox = (left, upper, left + sliceWidth, upper + sliceHeight)
            working_slice = img.crop(bbox)
            working_slice.save(os.path.join(outdir, str(upper) + '_' + str(left) + '.jpg'))
            upper += sliceHeight
        left += sliceWidth
        upper = 0
```

```
files_to_move=!ls {workingdir} | grep CRC
```

```
for i in range(1,11):
    !mkdir {workingdir+str(i)}
    !mv {workingdir+files_to_move[i-1]} {workingdir+str(i)+'/'+files_to_move[i-1]}
```

```
if __name__ == '__main__':
    for subdir, dirs, files in os.walk(workingdir):
        for file in files:
            image_slice(subdir + '/' + file, subdir, 150, 150)
```

```
for i in range(1,11):
    !mv {workingdir+str(i)+'/'+files_to_move[i-1]} {workingdir+files_to_move[i-1]}
```

7.9 Predicció de les regions de les WSI

```
for i in range(1,11):
        if i==1: files1 = os.listdir(f'{workingdir}'+f'{i}')
```

```
        if i==2: files2 = os.listdir(f'{workingdir}'+f'{i}')
        if i==3: files3 = os.listdir(f'{workingdir}'+f'{i}')
        if i==4: files4 = os.listdir(f'{workingdir}'+f'{i}')
        if i==5: files5 = os.listdir(f'{workingdir}'+f'{i}')
        if i==6: files6 = os.listdir(f'{workingdir}'+f'{i}')
        if i==7: files7 = os.listdir(f'{workingdir}'+f'{i}')
        if i==8: files8 = os.listdir(f'{workingdir}'+f'{i}')
        if i==9: files9 = os.listdir(f'{workingdir}'+f'{i}')
        if i==10: files10 = os.listdir(f'{workingdir}'+f'{i}')
```

```
def pintar(width, height):
    negre = (0,0,0)
    gris = (120,120,120)
    blanc = (1000,1000,1000)
    lila = (150,20,147)
    blau = (23,140,190)
    groc = (240,220,20)
    verd = (0,210,10)
    tronja = (240,120,30)
    for x in range(width):
        for y in range(height):
            if prediccions[a]== 'empty': pixels[x, y] = negre
            if prediccions[a]== 'adipose': pixels[x, y] = gris
            if prediccions[a]== 'mucosa': pixels[x, y] = blanc
            if prediccions[a]== 'debris': pixels[x, y] = lila
            if prediccions[a]== 'complex': pixels[x, y] = groc
            if prediccions[a]== 'stroma': pixels[x, y] = verd
            if prediccions[a]== 'tumor': pixels[x, y] = tronja
            if prediccions[a]== 'lympho': pixels[x, y] = blau
```

```
for i in range(1,11):
    prediccions = list()
    PATH_LARGE_SUBDIR=f'{workingdir}'+f'{i}'+"/"
    if i==1:
        for a in range(len(files1)):
            im = val_tfms(open_image(PATH_LARGE_SUBDIR + files1[a]))
            preds1 = learn.predict_array(im[None])
            prediccions.append(learn.data.classes[np.argmax(preds1)])
            imagen = Image.open(PATH_LARGE_SUBDIR+files1[a])
            pixels = imagen.load()
            width, height = imagen.size
            pintar(width,height)
            imagen.save(PATH_LARGE_SUBDIR + files1[a], 'JPEG')
    if i==2:
        for a in range(len(files2)):
            im = val_tfms(open_image(PATH_LARGE_SUBDIR + files2[a]))
            preds2 = learn.predict_array(im[None])
            prediccions.append(learn.data.classes[np.argmax(preds2)])
            imagen = Image.open(PATH_LARGE_SUBDIR+files2[a])
            pixels = imagen.load()
            width, height = imagen.size
            pintar(width,height)
            imagen.save(PATH_LARGE_SUBDIR + files2[a], 'JPEG')
    if i==3:
```

```
        for a in range(len(files3)):
            im = val_tfms(open_image(PATH_LARGE_SUBDIR + files3[a]))
            preds3 = learn.predict_array(im[None])
            prediccions.append(learn.data.classes[np.argmax(preds3)])
            imagen = Image.open(PATH_LARGE_SUBDIR+files3[a])
            pixels = imagen.load()
            width, height = imagen.size
            pintar(width,height)
            imagen.save(PATH_LARGE_SUBDIR + files3[a], 'JPEG')
    if i==4:
        for a in range(len(files4)):
            im = val_tfms(open_image(PATH_LARGE_SUBDIR + files4[a]))
            preds4 = learn.predict_array(im[None])
            prediccions.append(learn.data.classes[np.argmax(preds4)])
            imagen = Image.open(PATH_LARGE_SUBDIR+files4[a])
            pixels = imagen.load()
            width, height = imagen.size
            pintar(width,height)
            imagen.save(PATH_LARGE_SUBDIR + files4[a], 'JPEG')
    if i==5:
        for a in range(len(files5)):
            im = val_tfms(open_image(PATH_LARGE_SUBDIR + files5[a]))
            preds5 = learn.predict_array(im[None])
            prediccions.append(learn.data.classes[np.argmax(preds5)])
            imagen = Image.open(PATH_LARGE_SUBDIR+files5[a])
            pixels = imagen.load()
            width, height = imagen.size
            pintar(width,height)
            imagen.save(PATH_LARGE_SUBDIR + files5[a], 'JPEG')
    if i==6:
        for a in range(len(files6)):
            im = val_tfms(open_image(PATH_LARGE_SUBDIR + files6[a]))
            preds6 = learn.predict_array(im[None])
            prediccions.append(learn.data.classes[np.argmax(preds6)])
            imagen = Image.open(PATH_LARGE_SUBDIR+files6[a])
            pixels = imagen.load()
            width, height = imagen.size
            pintar(width,height)
            imagen.save(PATH_LARGE_SUBDIR + files6[a], 'JPEG')
    if i==7:
        for a in range(len(files7)):
            im = val_tfms(open_image(PATH_LARGE_SUBDIR + files7[a]))
            preds7 = learn.predict_array(im[None])
            prediccions.append(learn.data.classes[np.argmax(preds7)])
            imagen = Image.open(PATH_LARGE_SUBDIR+files7[a])
            pixels = imagen.load()
            width, height = imagen.size
            pintar(width,height)
            imagen.save(PATH_LARGE_SUBDIR + files7[a], 'JPEG')
    if i==8:
        for a in range(len(files8)):
            im = val_tfms(open_image(PATH_LARGE_SUBDIR + files8[a]))
            preds8 = learn.predict_array(im[None])
```

```
            prediccions.append(learn.data.classes[np.argmax(preds8)])
            imagen = Image.open(PATH_LARGE_SUBDIR+files8[a])
            pixels = imagen.load()
            width, height = imagen.size
            pintar(width,height)
            imagen.save(PATH_LARGE_SUBDIR + files8[a], 'JPEG')
    if i==9:
        for a in range(len(files9)):
            im = val_tfms(open_image(PATH_LARGE_SUBDIR + files9[a]))
            preds9 = learn.predict_array(im[None])
            prediccions.append(learn.data.classes[np.argmax(preds9)])
            imagen = Image.open(PATH_LARGE_SUBDIR+files9[a])
            pixels = imagen.load()
            width, height = imagen.size
            pintar(width,height)
            imagen.save(PATH_LARGE_SUBDIR + files9[a], 'JPEG')
    if i==10:
        for a in range(len(files10)):
            im = val_tfms(open_image(PATH_LARGE_SUBDIR + files9[a]))
            preds10 = learn.predict_array(im[None])
            prediccions.append(learn.data.classes[np.argmax(preds10)])
            imagen = Image.open(PATH_LARGE_SUBDIR+files10[a])
            pixels = imagen.load()
            width, height = imagen.size
            pintar(width,height)
            imagen.save(PATH_LARGE_SUBDIR + files10[a], 'JPEG')
```

7.10 Reconstruir WSI ja classificades

```
for g in range(len(files_to_move)):
    imagen = Image.open(workingdir+files_to_move[g])
    slices = !ls {workingdir+str(g+1)+'/'}
    for i in range(len(slices)):
        j=0
        k=1
        vertical= str()
        horitzontal= str()
        while slices[i][j]!="_":
            vertical= vertical+slices[i][j]
            j=j+1
        while slices[i][len(vertical)+k]!=".":
            horitzontal= horitzontal+slices[i][len(vertical)+k]
            k=k+1
        sliced = Image.open(workingdir+str(g+1)+'/'+vertical+"_"+horitzontal+'.jpg')
        imagen.paste(sliced,(int(horitzontal),int(vertical)))
    imagen.save(workingdir+ "class_" +f'{files_to_move[g]}')
```

7.11 Alpha Blending

```
for i in range(len(files_to_move)):
    image_WSI = Image.open(workingdir + f'{files_to_move[i]}')
    image_WSI = image_WSI.convert("LA")
    image_WSI = image_WSI.convert("RGBA")
    image_base = Image.open(workingdir + "class_" + f'{files_to_move[i]}')
    image_base = image_base.convert("RGBA")
    alphaComposited = Image.alpha_composite(image_WSI, image_base)
    alphaBlended = Image.blend(image_base, image_WSI,.4)
    alphaBlended.save(workingdir + "alpha_blended_" + f'{files_to_move[i]}')
```

7.12 Bicubic interpolation

```
import cv2

img = cv2.imread('/home/deak/imatge',  cv2.IMREAD_UNCHANGED)

print('Original Dimensions : ',img.shape)

scale_percent = 50 # percent of original size
width = int(img.shape[1] * scale_percent / 100)
height = int(img.shape[0] * scale_percent / 100)
dim = (width, height)
resized = cv2.resize(img, dsize=dim, interpolation = cv2.INTER_CUBIC)

print('Resized Dimensions : ',resized.shape)
cv2.imwrite('/home/deak/imatge',resized)
```

7.13 Median Filtering

```
median= cv2.medianBlur(resized, 9)
cv2.imwrite('/home/deak/imatge',median)
```

Bibliografia

Aurélien, Géron. 2017. *Hands-On Machine Learning with Scikit-Learn.* doi:10.3389/fninf.2014.00014.

B, Silvia Cascianelli, Raquel Bello-cerezo, Francesco Bianconi, Mario L Fravolini, Mehdi Belal, Barbara Palumbo, and Jakob N Kather. 2018. "Intelligent Interactive Multimedia Systems and Services 2017" 76. doi:10.1007/978-3-319-59480-4.

B, Tuan D Pham. 2017. "Advances in Neural Networks - ISNN 2017" 10262: 524–32. doi:10.1007/978-3-319-59081-3.

Evans, AndrewJ, ElizabethA Krupinski, Liron Pantanowitz, and RonaldS Weinstein. 2014. "2014 American Telemedicine Association clinical guidelines for telepathology: Another important step in support of increased adoption of telepathology for patient care." *Journal of Pathology Informatics* 6 (1): 13. doi:10.4103/2153-3539.153906.

Goodfellow, I, Y Bengio, and A Courville. 2016. "Deep learning." *Healthcare Informatics Research* 22 (4): 351–54. doi:10.1038/nature14539.

ICO. 2010. "El càncer colorectal." *Institut Català d'Oncologia.*

K. He, X. Zhang, S. Ren, J. Sun. 2015. "Deep Residual Learning for Image Recognition." *Cornell University Arxiv.* doi:10.1016/0141-0229(95)00188-3.

Kather, Jakob Nikolas, Cleo-Aron Weis, Francesco Bianconi, Susanne M. Melchers, Lothar R. Schad, Timo Gaiser, Alexander Marx, and Frank Gerrit Zöllner. 2016. "Multi-class texture analysis in colorectal cancer histology." *Scientific Reports* 6 (1). Nature Publishing Group: 27988. doi:10.1038/srep27988.

Matthew D. Zeiler, Rob Fergus. 2013. "Visualizing and Understanding Convolutional Networks." *Cornell University Arxiv.* doi:10.1111/j.1475-4932.1954.tb03086.x.

Pantanowitz, Liron, Navid Farahani, and Anil Parwani. 2015. "Whole slide imaging in pathology: advantages, limitations, and emerging perspectives." *Pathology and Laboratory Medicine International*, 23. doi:10.2147/PLMI.S59826.

Pollheimer, Marion J., Peter Kornprat, Richard A. Lindtner, Lars Harbaum, Andrea Schlemmer, Peter Rehak, and Cord Langner. 2010. "Tumor necrosis is a new promising prognostic factor in colorectal cancer." *Human Pathology* 41 (12). Elsevier Inc.: 1749–57. doi:10.1016/j.humpath.2010.04.018.

Romeny, Haar, International Conference, and David Hutchison. 2004. *Image Analysis and Recognition.* Vol. 3212. doi:10.1007/b100438.

Smith, Leslie N. 2017. "Cyclical learning rates for training neural networks." *Proceedings - 2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017*, no. April: 464–72. doi:10.1109/WACV.2017.58.

Spanhol, Fabio A., Luiz S. Oliveira, Caroline Petitjean, and Laurent Heutte. 2015. "A Dataset for Breast Cancer Histopathological Image Classification." *IEEE Transactions on Biomedical Engineering* 63 (7): 1455–62. doi:10.1109/TBME.2015.2496264.

Spanhol, Fabio Alexandre, Luiz S Oliveira, Caroline Petitjean, and Laurent Heutte. 2016. "Breast cancer histopathological image classification using Convolutional Neural Networks." *International Joint Conference on Neural Networks (IJCNN)*, 2560–7. doi:10.1109/IJCNN.2016.7727519.

Stenkvist, Björm, Siahild Westman-Naeser, Jan Holmquist, Bo Nordin, Ewert Bengtsson, Jan Veaelius, Olle Eriksson, and Cecil H. Fox. 1978. "Computerized Nuclear Morphometry as an Objective Method for Characterizing Human Cancer Cell Populations." *Cancer Research* 38 (12): 4688–97. doi:10.1371/journal.pone.0016208.