Classificació i segmentació de tumors cerebrals
Programació informàtica i les seves aplicacions

Antoni Lopera Barril
Oriol Parera Cuscó

December 26, 2024

1 Introducció

El *Deep Learning* ha revolucionat el món de la imatge mèdica, permetent una anàlisi precisa i automatitzada de dades complexes. L'objectiu d'aquest projecte és la implementació de dos models DL per analitzar la ressonància magnètica cerebral: una classificació i un model de segmentació. En tots els casos, es treballarà amb imatges obtingudes per ressonància magnètica (MRI).

En la primera fase de classificació, es determina si una imatge d'entrada conté un tumor i, si hi és present, el categoritza en un dels tres tipus inclosos entre les dades: pituitari, glioma, o meningioma.

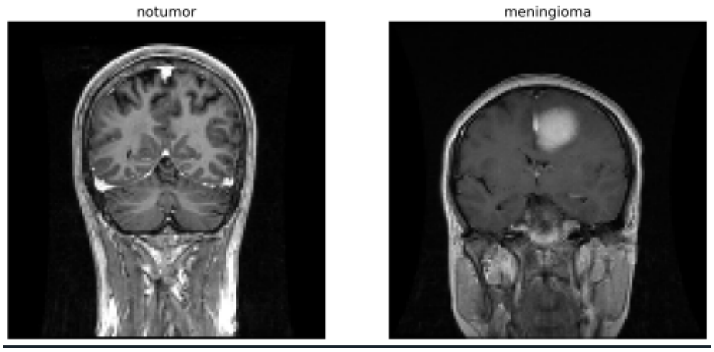


Figure 1: Tumor vs No Tumor

Per a imatges identificades com a contingents de tumors, a la segona etapa, corresponent al model de segmentació, s'aïlla i es delinea la regió tumoral, proporcionant informació crucial sobre la mida i la ubicació amb finalitats diagnòstiques i terapèutiques.

Ambdós models són actualment entrenats per separat, i com a futura implementació es proposa un entrenament en paral·lel utilitzant arquitectures de Python. Aquest informe detalla les eines i conjunts de dades utilitzades, els processos d'implementació, els resultats, l'ús de recursos, i conclou amb una anàlisi de les limitacions i les possibles millores.

Un esquema d'aquest plantejament és el mostrat a la figura 2

2 Experiments

2.1 Models i eines utilitzades

2.1.1 Model de classificació

En aquest apartat del treball es presenta un enfocament per a la classificació d'imatges utilitzant una xarxa neuronal convolucional profunda basada en DenseNet201, una arquitectura prèviament entrenada sobre el conjunt d'imatges ImageNet. Aquesta arquitectura permet aprofitar coneixements generals sobre característiques d'imatges que han estat prèviament apreses, millorant així la capacitat de classificació en noves tasques. [2]

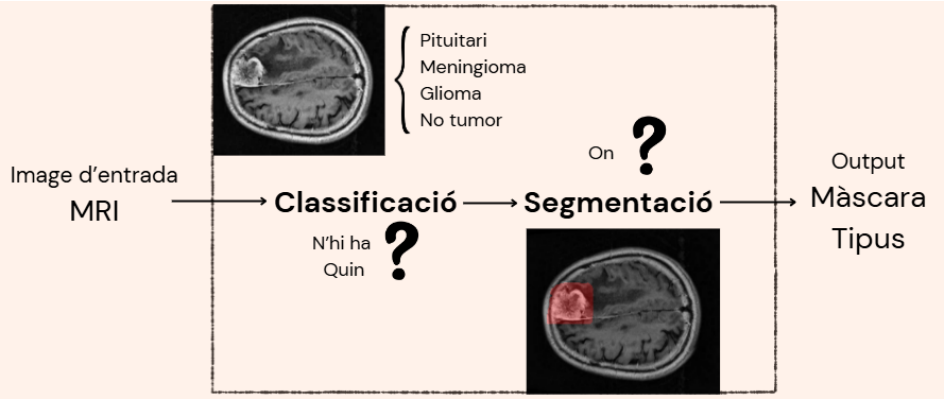


Figure 2: Esquema del model compost

El procés de preparació dels conjunts de dades comença amb la utilització de `ImageDataGenerator` per carregar i augmentar les imatges d'entrenament. S'ha de destacar que s'entren dues carpetes, amb el nom de Training i Testing, on cadascuna d'aquesta conté 4 carpetes més, cada una amb el nom del tumor o amb el no tumor. A més, aquesta classe permet aplicar transformacions com desplaçament horitzontal, rotacions i zoom per augmentar la diversitat de les imatges i evitar el sobreajustament (overfitting). Les imatges es redimensionen i es normalitzen perquè els valors dels píxels es situïn entre 0 i 1. Per a les dades d'entrenament, es realitzen augmentacions amb diferents transformacions per simular variacions que podrien ocórrer en un conjunt de dades real.

Per a la validació, les imatges es normalitzen sense aplicar augmentacions, per mantenir la coherència del conjunt de dades. I finalment, es fa un conjunt de prediccions per assegurar que el model és consistent pels diferents estudis que es volen realitzar, amb el posterior anàlisis de resultats.

2.1.2 Model de segmentació

L'objectiu principal d'aquest model és identificar de manera precisa la localització i els límits de tumors dins imatges mèdiques classificades prèviament com a contingents de tumor. Per dur a terme aquesta tasca, s'ha escollit l'arquitectura *Mask R-CNN*, un model de DL que combina la detecció d'objectes i segmentació d'aquests en visió computacional.

Per començar, s'inicialitza el model amb pesos pre-entrenats[1], prenent avantatge de la transferència d'aprenentatge per adaptar el model eficientment al domini de la imatge mèdica. En quant a la sortida del model, aquesta és una màscara binària amb píxels corresponents a tumor amb valor 1 i píxels sense tumors amb 0.

El *backbone* utilitzat en aquest cas, és a dir, la part responsable per l'extracció d'informació valuosa de les imatges d'entrada, es tracta de *ResNet-50*. Aquest serà l'encarregat d'extreure patrons, contorns, textures i més detalls d'alt nivell. A més, aquesta base s'emparella amb una *Feature Pyramid Network*, que agafa els mapes de característiques de *ResNet* i els processa barrejant diferents resolucions de les imatges per assegurar que es pot processar qualsevol tipus de qualitat.

Pel que fa a com treballa el model, aquest accepta les imatges d'entrada i en genera sobre aquestes regions d'interes o ROIs (*region of interests*) on és probable que l'objecte a detectar s'hi trobi. Finalment, una xarxa convolucional completa o FCN (*Fully convolutinal network*) prediu la màscara binària per cada ROI.

No obstant, aquest model base no proporciona uns resultats sòlids, ja que està preparat per la detecció d'objectes a gran escala, i no se centra en la detecció de cossos petits o detallats, com poden ser els tumors en les MRI. Per tant, és obvi que cal dur a terme un entrenament del model amb un *dataset* dissenyat específicament per a segmentació de tumors cerebrals. En aquesta part del procés, s'estudiaran diferents configuracions dels paràmetres: nombre d'èpoques o *epochs* (cops que el model entrena amb el conjunt d'imatges complet), el nombre de lots o *batches* (paquets en què es divideixen les èpoques), l'optimitzador, la mètrica de pèrdues i de precisió, etc.

[1] Aquests pesos pre-entrenats són obtinguts del *dataset* COCO (*Common Objects in Context*), especificats dins la llibreria *torchvision*.

Malgrat i que aquest model finalment ha acabat donant resultats presentables segons els nostres objectius inicials, no és el primer tipus de model que s'ha provat. Segmentar tumors, al ser una tasca complicada, requereix d'un model ben preparat anteriorment. Per arribar a aquesta conclusió, prèviament es van realitzar execucions, entrenaments i proves amb models més senzills que no calia carregar i que podien ser programats usant llibreries de DL ofertes per Python. A continuació, es descriurà breument el model proposat inicialment i que va ser descartat per males mètriques.

Aquesta esmentada primera versió es basava en un model U-Net, caracteritzat principalment per la seva arquitectura dividida en un bloc de codificació i un altre de decodificació. El primer usa capes convolucionals i de reducció de dimensions per extraure característiques rellvants de cada imatge, i el segon bloc usa capes per reconstruir la resolució de la imatge original combinant patrons extrets anteriorment mitjançant concatenacions d'imatges processades. En aquí, es permet llibertat sobre quantes convolucions i reduccions dur a terme, millorant els resultats a mesura que s'augmenta aquesta quantitat, però fent molt més exigent la computació. El model s'entrenava amb les mateixes imatges que el que s'usa actualment i la sortida era idèntica: una màscara binària per indicar la regió del tumor. Tot i que aquest model permetia una personalització major i conté una estructura més clara del procés global, els resultats que obteníem no eren gens satisfactoris. El motiu que podem atribuir-li és el fet que sense carregar uns pesos inicials pre-entrenats, el model requeria de una quantitat molt més gran de passos per arribar al mateix punt de precisió. Degut a això, la decisió va ser buscar una alternativa que ens estalvies aquest primer procés d'ajustament dels pesos i no haver d'iniciar el procés de zero amb una xarxa neuronal sense coneixement previ.

2.2 Descripció de les dades

Per ambdues fases del projecte, les imatges que composen els *datasets* independents són imatges de ressonància magnètica (MRI) i són prèviament processades per adaptar-les als models en cada cas. A part de les imatges, s'han de tenir en compte altres tipus de dades incloses en els conjunts d'entrenament concrets, tal i com s'indica en les següents seccions. Pel processament d'aquestes imatges, en ambdós casos, al ser cada imatge individual independent de les altres, es proposa l'ús d'un plantejament paral·lel per millorar l'eficiència en quant a ús de recursos disponibles de la nostra màquina i un significatiu estalvi de temps.

2.2.1 Dades de classificació

Per a la classificació, disposem de les imatges d'entrenament prèviament classificades segons la presència de tumor i, en cas de tenir-ne, segons el tipus de tumor. En concret, el dataset ha estat obtingut de Kaggle, on es distribueixen les imatges en dues carpetes principals: una per a l'entrenament i una altra per a les proves (test). Cada una d'aquestes carpetes es divideix en subcarpetes que contenen les imatges corresponents als diferents tipus de tumors i a les imatges sense tumor. Les subcarpetes són:

- **Meningioma**: Conté les imatges que mostren tumors meningiomatosos.
- **Glioma**: Conté les imatges que mostren tumors gliomatosos.
- **Pituitari**: Conté les imatges que mostren tumors pituitaris.
- **No Tumor**: Conté les imatges que no presenten cap tipus de tumor.

Aquest dataset estructurat ens permet entrenar i avaluar el model de classificació mitjançant les imatges etiquetades en cada una de les subcarpetes. Així, es pot mesurar el percentatge d'encerts del model en la tasca de classificar imatges segons la presència i el tipus de tumor. Els resultats obtinguts es poden comparar amb les etiquetes prèviament assignades a les imatges, permetent una avaluació precisa del rendiment del model. D'aquesta manera, es pot medir el percentatge d'encerts del model.

2.2.2 Dades de segmentació

En quant a la segmentació, el procés és diferent, ja que no ens val amb una etiqueta per cada imatge, sinó que necessitem etiquetar cada píxel binàriament segons si pertany a la zona del tumor (1) o no (0). El *dataset* que usem en aquesa fase ha estat extret de Kaggle ([1]). Aquesta informació està inclosa

en un document .json (*JavaScript Object Notation*), en què cada cada imatge està identificada com a un objecte i un dels seus atributs es tracta de la posició dels píxels que delimiten el tumor.

Es disposen de 1502 imatges, de les quals degut a limitacions físiques de hardware, s'ha reduït a 250 perquè el cost computacional i recursos requerits sobrepassen els disponibles. Aquestes imatges tenen una resolució de 640x640 píxels, i pels mateixos motius especificats anteriorment, són redimensionalitzades a 320x320, reduïnt en un factor de 4 els *arrays* a processar, malgrat i alhora estar disminuint lleugerament la capacitat d'extracció de patrons.

Les imatges són carregades i processades de la següent manera. En primer lloc, se seleccionen 250 objectes aleatoris de l'arxiu d'anotacions on hi ha la informació de les imatges i se'n construeix la màscara en forma d'*array* binari dibuixant el polígon delimitat dels píxels indicats a l'arxiu de text. Seguidament, les corresponents imatges també es converteixen a *array* i posteriorment es normalitzen entre 0 i 1 no binàriament tenint en compte que són imatges de 8 bits. Per últim, tant la màscara com la imatge es reescalen a 320x320 píxels.

Un cop totes les imatges i màscares són processades i emmagatzemades, es pot iniciar el procés d'entrenament i posterior validació.

2.3 Configuració experimental

Un cop descrits els models a usar i el tipus de dades que manipularan, tant d'entrada com de sortida, per poder reproduir el procés que s'ha dut a terme cal indicar quina configuració s'ha utilitzat.

S'han dut a terme diferents execucions de la fase d'entrenament dels models per poder trobar quin conjunt de paràmetres resulta l'òptim i, atenent a les limitacions de processament que tenim i repartiment de tasques, els dos models s'han posat a prova per separat i al final són ajuntats en un sol *script* per la implementació última.

Abans d'entrar en detall, es comentaran breument els trets generals. S'utilitza Python com a llenguatge de programació i les seves llibreries que més endavant es comentaran. Pel que fa al hardware, s'ha utilitzat una maquinària amb un processador *11th Gen Intel(R) Core(TM) i5*, 4 nuclis, amb 8 GB de memòria RAM en un sistema operatiu Windows 11. Tot i no ser l'entorn més òptim, no s'ha pogut aconseguir millors especificacions.

2.3.1 Configuració experimental per la classificació

Per entrenar el model, es defineixen les següents configuracions:

- **Funció de pèrdua:** `categorical_crossentropy`, adequada per problemes de classificació multiclasse.
- **Optimitzador:** `Adam`, amb una taxa d'aprenentatge inicial de 0.0001.
- **Mètriques:** Es fa servir la precisió (`accuracy`) per monitoritzar el rendiment del model.

A més, es defineixen diverses estratègies per millorar el procés d'entrenament:

- **ModelCheckpoint:** Desa el model amb la pèrdua de validació més baixa.
- **EarlyStopping:** Atura l'entrenament si la pèrdua de validació no millora després de 10 èpoques consecutives.
- **ReduceLROnPlateau:** Redueix la taxa d'aprenentatge si la pèrdua de validació no millora durant 5 èpoques.

2.3.2 Configuració experimental per la segmentació

Els paràmetres d'experimentació que s'han anat variant i l'evolució dels quals s'ha estudiat són el nombre d'èpoques i l'optimitzador, mantenint en tots els casos el ritme d'aprenentatge inicial (*learning rate* = 1) i el nombre de lots (*batch size* = 4) constants. També cal mencionar que la mètrica de pèrdues (*loss function*) està implícitament definida dins el model *Mask R-CNN* i treballa internament, per tant no ens cal definir-la.

La mètrica que utilitzarem per mesurar la precisió del nostre model i comprovar si millora o empitjora a mesura que s'entrena és la mètrica d'intersecció sobre unió o IoU (*intersection over union*).

Aquesta és la més usada en quant a mesurar la precisió d'una segmentació predita comparada amb la considerada correcte, pren un valor entre 0 i 1, i és calculada com la divisió entre l'àrea superposada de la predicció amb la màscara real i l'àrea total coberta per les dues màscares.

Un cop realitzat l'estudi, el conjunt òptim de paràmetres que s'ha trobat mitjançant l'experimentació és de 4 èpoques amb l'optimitzador AdamW (*Adaptative Moment Estimation with Weight Decay*). El que fa aquest optimitzador és millorar l'Adam, el qual canvia el ritme d'aprenentatge de cada paràmetre en funció d'estimacions dels gradients, mitjançant la reducció automàtica dels pesos al càlcul dels gradients, assegurant millor regularització i generalització.

3 Descripció detallada de la implementació

En aquesta secció, el que es farà serà presentar de quina manera s'ha posat en pràctica tot el que s'ha mencionat anteriorment: des del processat d'imatges fins a l'obtenció de resultats passant per l'entrenament dels models. S'analitzarà pas per pas què realitza cada funció del codi per poder arribar a l'objectiu final del projecte.

3.1 Implementació de la classificació

3.1.1 Preparació del conjunt de dades

La funció `prepare_the_datasets` és la responsable de pre-processar les dades d'entrenament i validació per adaptar-les al model de classificació. Aquesta funció utilitza `ImageDataGenerator` per aplicar diverses transformacions a les imatges amb l'objectiu d'ampliar el conjunt de dades i millorar la robustesa del model. Entre aquestes transformacions es troben el reescalat dels valors dels píxels, rotacions, desplaçaments i reflexions.

A més, es defineixen dos generadors de dades:

- **Generador d'entrenament:** Aplica augmentació de dades amb canvis com ara el desplaçament horitzontal i vertical, zoom, reflexions i altres variacions.
- **Generador de validació:** Només aplica el reescalat dels valors dels píxels per normalitzar les imatges.

Cada generador carrega les dades d'un directori especificat, les reorganitza en lots segons la mida (`batch_size`) i les adapta a les dimensions esperades pel model (en aquest cas, 256x256).

3.1.2 Definició del model

El model utilitza la xarxa pre-entrenada `DenseNet201`, carregada amb els pesos d'`ImageNet`. Es desactiva l'entrenament de les capes del model base per preservar el coneixement adquirit durant l'entrenament previ. A sobre del model base es defineix una capçalera (*top model*) personalitzada, que consta de les següents capes:

- `Flatten:` Converteix les sortides del model base en un vector unidimensional.
- `Dense:` Una capa densa amb activació ReLU i 64 neurones.
- `Dropout:` Una capa que aplica un percentatge de desconnexió del 20% per prevenir el sobreajustament.
- `Dense:` Una capa final amb activació softmax, corresponent al nombre de classes (`num_classes`).

El model complet s'instancia combinant la sortida del model `DenseNet201` amb aquesta capçalera.

3.1.3 Configuració de l'entrenament

Per entrenar el model, es defineixen les següents configuracions, explicades anteriorment:

- **Funció de pèrdua categorical_crossentropy**
- **Optimitzador Adam**

- **Accuracy**
- **ModelCheckpoint**
- **EarlyStopping**
- **ReduceLROnPlateau**

3.1.4 Entrenament i avaluació del model

El model s'entrena durant 50 èpoques amb una mida de lot (*batch size*) de 64, utilitzant els conjunts d'entrenament i validació preparats. El nombre de passos per època es calcula dividint el nombre total d'imatges pel *batch size*. Un cop completat l'entrenament, el model es guarda en un fitxer amb extensió `.keras`, i es fa una avaluació final sobre les dades de validació per determinar la pèrdua i la precisió.

3.2 Implementació de la segmentació

3.2.1 Preparació del conjunt de dades

La primera part del codi consta de la creació d'un objecte anomenat `BrainTumorDataset`, amb l'objectiu de pre-processar les dades i preparar-les pel format que el model s'espera com a entrada.

S'hi defineixen diferents mètodes: `__init__(self,images,masks)`, per inicialitzar la classe amb atributs `images` i `masks`; `__len__(self)`, retorna el nombre d'imatges que es tenen per obtenir el tamany del *dataset*; `__getitem__(self,idx)`, que retorna informació de la mostra a la posició `idx`. Aquest últim mètode comença obtenint la imatge i la corresponent màscara, n'extreu un identificador (ID), exclou el fons de la màscara (valors assignats a 0) i crea una màscara binària per cada ID d'objecte, en computa les coordenades del rectangle que delimita els marges de la màscara, la converteix a un tensor i finalment inclou tota la informació en un diccionari del format esperat pel model *Mask R-CNN*.

3.2.2 Processat d'anotacions

El processat de l'arxiu d'anotacions es duu a terme amb les funcions `load_annotations_parallel` i `process_single_annotation`, presentant una manera organitzada de preparar imatges i crear-ne màscares per utilitzar-les en l'entrenament.

La primera funció comença llegint l'arxiu .json que conté la *metadata* i informació referent a la segmentació de cada imatge específica indentificada pel seu ID únic. Per optimitzar la tasca, un cop llegides les anotacions, s'utilitza un plantejament paral·lel amb `ThreadPoolExecutor` per tal de processar tasques de manera molt més eficient. Cada *thread* és aleshores responsable d'un cert nombre de lots. Per cada imatge dins el lot es crida a la funció `process_single_annotation` i el corresponent *output* obtingut s'afegeix a dues llistes: `images` i `masks`.

Pel que fa a la segona funció, aquesta processa cada imatge de manera individual, tal i com s'indica al seu nom. Primer de tot, extreu la informació de l'arxiu d'anotacions corresponent a l'argument d'entrada i busca la imatge a la corresponent carpeta on es troba i converteix l'arxiu a format RGB. Seguidament crea la màscara binària (*true mask*) que remarca la zona on hi ha el tumor i dibuixa el polígon resultant sobre una imatge en blanc. Aleshores té lloc el reescalat de dimensions 640x640 a 320x320, es normalitzen els valors dels píxels de l'*array* de la imatge per què vagin de 0 a 1 en comptes de 0 a 255, ja que treballem amb imatges de 8 bits, i es converteix la màscara a format binari. Finalment, es retornen els *arrays* o matrius corresponents a la parella imatge-màscara i també s'emmagatzemen en format imatge visualitzable a directoris indicats per possibles comprovacions.

Un exemple d'imatge processada i construcció de la corresponent *true mask* és el mostrat en la figura 3.

3.2.3 Computació de la mètrica IoU

La funció dedicada a això és `compute_iou`, que rep dos *arrays* d'entrada corresponents a la màscara predita pel model (`pred_mask`) i la correcte (`true_mask`), ambdues binàries. Amb operadors lògics es calcula la intersecció (`logical_and`) i la unió (`logical_or`) i es retorna el ratio entre aquestes dues magnituds (fig. 4).

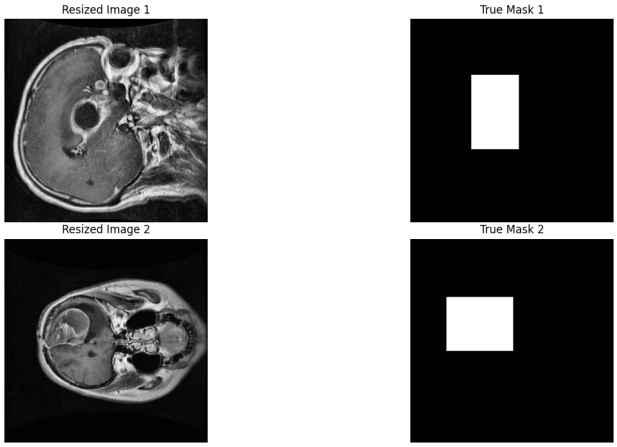


Figure 3: Imatges i corresponents màscares

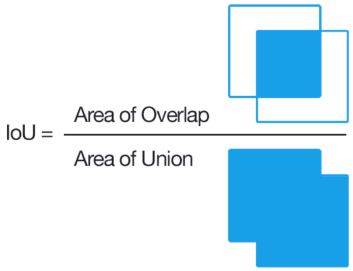


Figure 4: IoU formula and visualization

3.2.4 Entrenament del model

Aquesta és probablement la funció més important per la implementació d'aquest model amb una precisió suficientment bona i dins els nostres criteris de qualitat. `train_model` rep els següents elements com arguments d'entrada: model pre-entrentat, dades d'entrenament i validació, nombre d'èpoques, ritme d'aprenentatge, optimitzador, tamany de lots i rutes d'accés als directoris on emmagatzemar el model i les mètriques resultants.

Es comença inicialitzant l'optimitzador, creant un diccionari per emmagatzemar les mètriques de cada època i configurant el model en mode d'entrenament. Seguidament, s'entra en un bucle per cada una de les èpoques d'entrenament que s'han de realitzar. Iterant per tot el set d'imatges i màscares d'entrenament, aquest es divideix en els lots. Per cada lot, es mouen les imatges al dispositiu que l'ha de processar (CPU o GPU, si n'hi ha), reinicia els gradients, realitza un pas endavant en els valors dels pesos i són actualitzats.

Un cop acabada la fase d'entrenament, té lloc la validació. Del *dataset* complet, un 20% és reservat a validació, i el 80% restant és el que ha estat utilitzat per entrenament. En aquest cas, com el conjunt d'imatges és de 250 per limitacions computacionals, 200 imatges són destinades a entrenament i 50 a validació. En aquesta nova fase, es canvia el model a mode evaluació, s'itera sobre les imatges reservades i es generen les prediccions de màscares usant el model ja entrenat, calculant-ne la mètrica IoU i després es mostra per pantalla.

A part, per cada època, es fa un seguiment de les mètriques d'interès: l'*epoch loss*, el valor IoU mig, el temps d'execució i el % d'ús de CPU o GPU si escau.

Al final de tot, es guarda el model com a *checkpoint*, per si es vol entrenar sobre aquest de nou o directament evaluar imatges individuals i obtenir-ne la màscara de predicció a la sortida.

3.3 Implementació del model compost

3.3.1 Carregament dels models

El primer pas a dur a terme per l'aplicació composta és carregar els model de classificació i segmentació prèviament entrenats independentment. El primer està basat amb `keras`, es realitza pujant directament l'arxiu corresponent al model, i el segon es carrega usant *Mask R-CNN* sense pesos i aquests són actualitzats més tard carregant l'últim *checkpoint*.

3.3.2 Predicció del tumor

Aquesta funció `predict_tumor_type` serà l'encarregada d'usar el model de classificació i determinar a quina de les 4 classes pertany la imatge que entrem al model compost amb la funció `model.predict(x)`, on `x` és la imatge convertida a RGB, reescalada i normalitzada. Retorna tant el tipus de tumor com la imatge.

3.3.3 Segmentació del tumor

La funció `segment_image` és l'encarregada de, quan calgui, delinar la regió tumoral sobre una màscara binària. S'entra la imatge, es converteix en tensor (`img_tensor`) i s'emmagatzema al dispositiu (CPU o GPU), es posa el model en mode evaluació (`model.eval()`) i s'envia a la xarxa neuronal (`model(img_tensor)[0]`). Finalment, si la màscara és no nul·la, es retorna la màscara predita pel model.

3.3.4 Funció principal

El pilar del programa ara és molt senzill d'explicar. Després de carregar els dos models, crida la funció `predict_tumor_type` i si la sortida és diferent a "No Tumor", procedeix a enviar la imatge al segon model de segmentació. En cas negatiu, el programa acaba en aquest punt.

Si escau usar la segmentació, es crida a la funció `segment_image` i finalment es visualitza el resultat, indicant la regió tumoral sobre la imatge i el tipus de tumor detectat.

4 Resultats

4.1 Presentació i anàlisi dels resultats de classificació

Tot i haver dissenyat i implementat un codi específicament orientat a la distinció entre els tres tipus de tumors cerebrals (glioma, meningioma i pituitari), no hem aconseguit obtenir una classificació precisa pels gliomes. Els resultats obtinguts en aquesta part de l'estudi han mostrat limitacions, probablement relacionades amb la complexitat de les característiques visuals de les imatges o amb la capacitat del model per diferenciar entre les subtils variacions entre els tipus de tumors. No obstant això, hem obtingut un bon rendiment en la tasca de diferenciar les imatges que contenen tumors de les que no en contenen. El model ha aconseguit identificar amb força precisió les imatges que mostren la presència de tumors, la qual cosa ens ha permès establir una primera línia de separació entre les imatges amb tumor i les que no en tenen. Això no ha suposat un problema per la segona etapa, vist que és indiferent el tipus de tumor.

A continuació, es mostren diversos exemples que il·lustren el funcionament de la distinció entre imatges amb tumor i sense tumor (sense especificar tipus de tumor). Per dur a terme aquesta tasca, s'han utilitzat les quatre carpetes creades (no-tumor, glioma, meningioma, pituitari) i s'han sotmès al model de classificació. El model ha generat quatre histogrames on es mesura el nombre d'imatges classificades com tumor i com no tumor. A partir d'aquests resultats, es pot observar que, en general, el model funciona correctament en la tasca de distingir entre imatges amb presència de tumor i imatges sense tumor, tot i que la distinció entre els diferents tipus de tumors pot necessitar més ajustos.

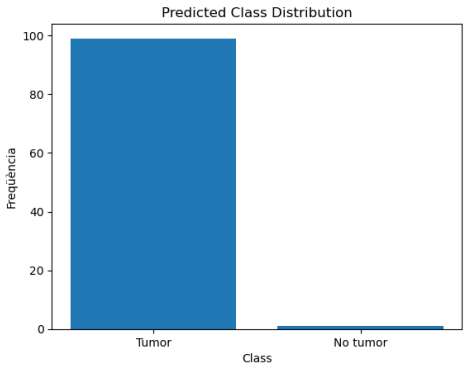


Figure 5: Distribució de tumors i no tumors, a partir de 100 imatges de la carpeta de Testing de gliomes.

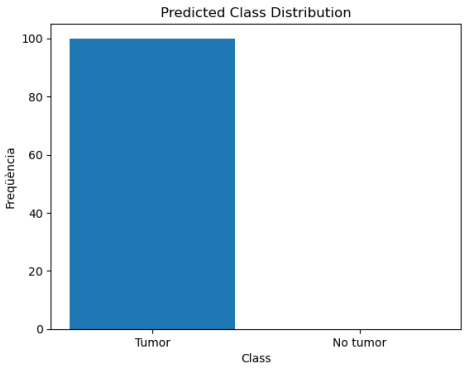


Figure 6: Distribució de tumors i no tumors, a partir de 100 imatges de la carpeta de Testing de Piuritary.

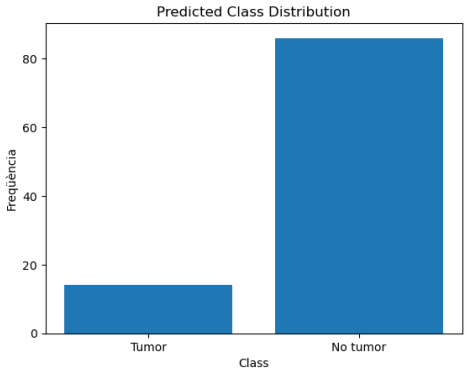


Figure 7: Distribució de tumors i no tumors, a partir de 100 imatges de la carpeta de Testing de No Tumor.

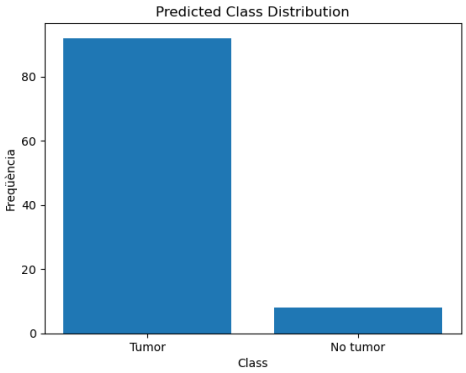


Figure 8: Distribució de tumors i no tumors, a partir de 100 imatges de la carpeta de Testing de Meningioma.

Un cop presentada la distinció entre imatges amb tumor i sense tumor, també s'inclouran els resultats obtinguts per a la classificació dels diferents tipus de càncer. Cal destacar que, tot i haver intentat millorar la precisió del model en aquesta classificació, els resultats obtinguts no són prou precisos i requerixen un model més avançat, que escapa dels nostres coneixements actuals, per poder abordar els petits detalls que justifiquen les diferències entre els diversos tipus de tumors. No obstant això, la limitació més evident és el cas del glioma, ja que es pot observar que és l'unic no predomiant en predir 80 imatges d'una carpeta de validació concreta per a cada figura.

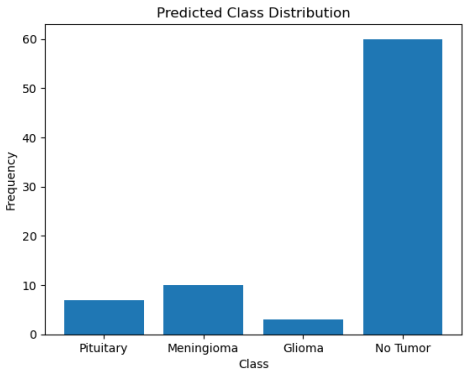


Figure 9: Predicció de 80 no tumors

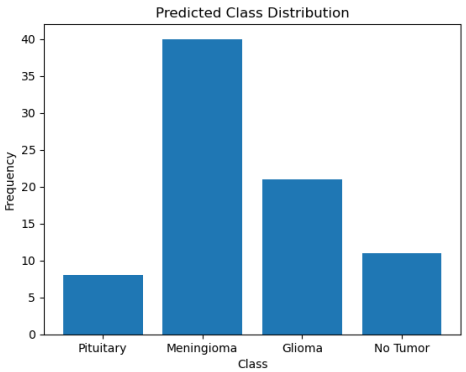


Figure 10: Predicció de 80 glioma

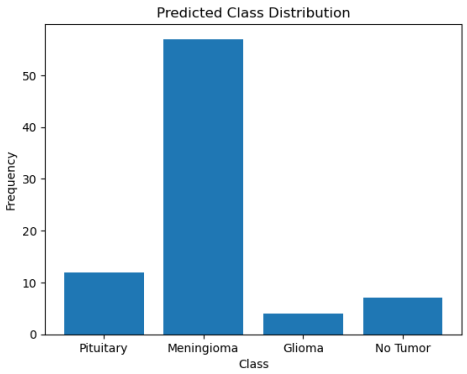


Figure 11: Predicció de 80 meningioma

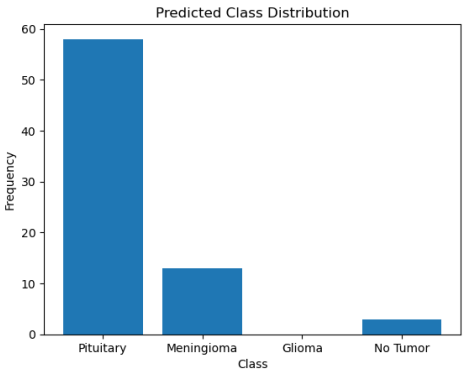
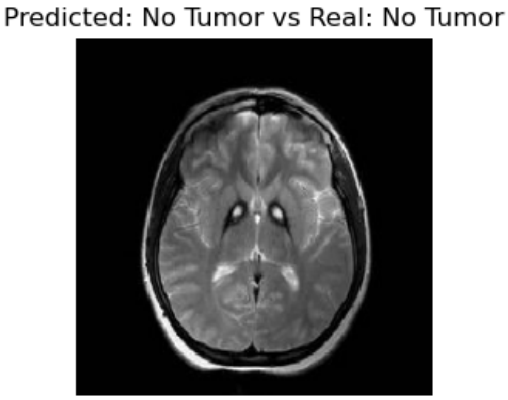


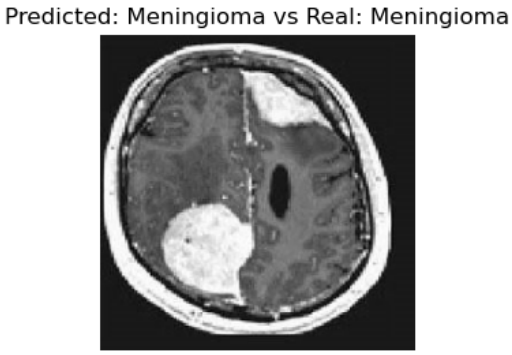
Figure 12: Predicció de 80 pituitary

A continuació, es mostra la sortida correcta de quatre gràfics que il·lustren les prediccions realitzades pel model per a diferents tipus de tumors i casos sense tumor.

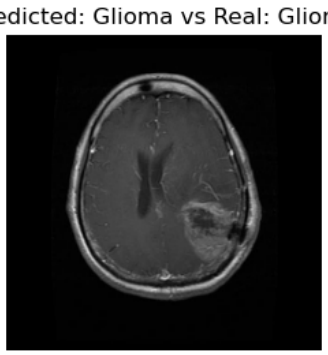
Predicció correcta del "Pituitary"



Predicció correcta de "No Tumor"



Predicció correcta de "Meningioma"



Predicció correcta de "Glioma"

Figure 13: Prediccions correctes per cada classe

4.1.1 Visualització de les mètriques d'entrenament

La funció `plot_training_curves` genera gràfics per visualitzar l'evolució de la pèrdua i la precisió durant l'entrenament i la validació. Aquests gràfics permeten analitzar el comportament del model al llarg de les èpoques i identificar possibles problemes com ara sobreajustament o subentrenament. Al final del procés d'entrenament, el millor model es recupera mitjançant els callbacks configurats durant la definició de la xarxa neuronal, permetent recuperar-los pels llocs on s'ha localitzat més precisió de

validació.

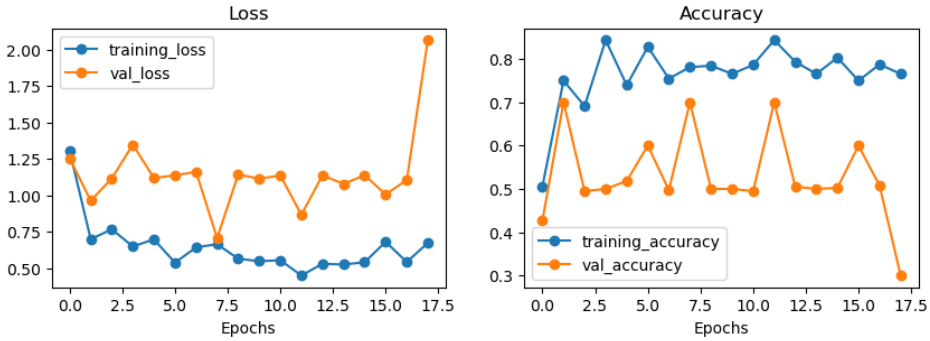


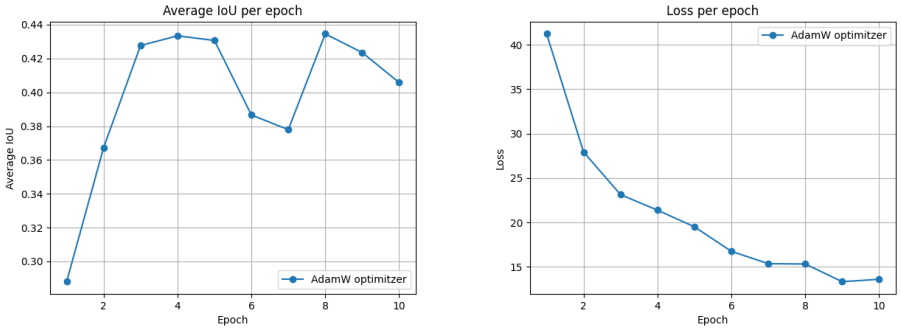
Figure 14: Evolució de la pèrdua i la precisió durant l'entrenament

4.2 Presentació i anàlisi dels resultats de segmentació

La continuació natural de tota la implementació, plantejament i disseny dels models, és analitzar-ne tots els resultats obtinguts. Ens caldrà tenir en compte dos factors: la sortida del model compost quan avaluem una imatge d'entrada i també tots els resultats que ens permeten saber quin conjunt de paràmetres és millor pel nostre interès, ja que ens aporten la sortida més precisa al final.

El procés que se segueix quan hi ha una imatge d'entrada és el següent. El primer model l'analitza i informa sobre si no hi ha tumor o si n'hi ha. En el segon cas, informa del tipus de tumor i posteriorment s'envia la imatge al segon model, el qual prediu amb una màscara on es troba aquest tumor, el marca i ho mostra com a figura a la pantalla.

Amb el set d'entrenament de segmentació, pels diferents experiments que s'han realitzat i mantenint *learning rate* = 1 i *batch size* = 4, obtenim les mètriques d'encert i pèrdues en cada època. Com es veu a les gràfiques de la figura 15, amb l'optimitzador AdamW, trobem que a mesura que avancen les èpoques augmenta significativament el ratio d'intersecció sobre unió, tret d'alguna irregularitat, i disminueix la pèrdua. Malgrat això, arriba un punt on la precisió se satura i no augmenta més. Aleshores, el que cal fer és trobar un punt mig que ens serveixi per assegurar una bona precisió amb poques pèrdues, un consum balançejat de recursos i per últim, evitar un sobre-entrenament del model a les dades d'entrenament, cosa que s'aconsegueix mantenint un nombre d'èpoques no excessivament elevat.

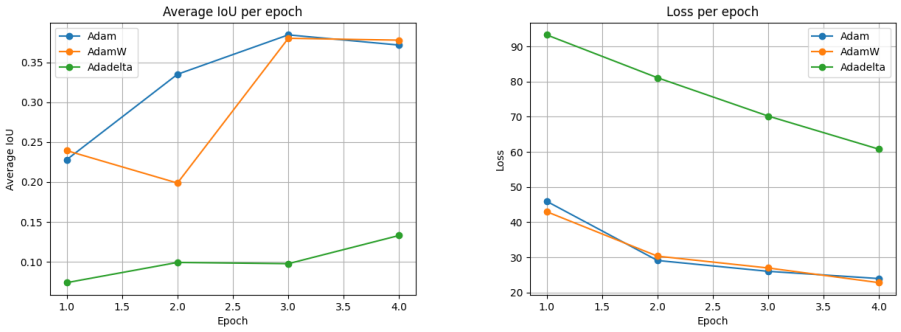


(a) Mètrica IoU

(b) Funció de pèrdua

Figure 15: Mètriques d'encert en fase de segmentació

Inicialment, d'aquest entrenament de 10 èpoques, extraiem la conclusió que no és encertat un entrenament tan llarg, sinó que ens cal reduir-lo. De les dades obtingudes, concluïm que l'ideal seria al voltant de 4 èpoques, amb un IoU mig de 43%. Després d'aquesta fita, es pot realitzar algun experiment més variant l'optimitzador utilitzat i fixant el nombre d'èpoques a 4. Precisament, aquesta proposta s'ha dut a terme en la figura 16, pels optimitzadors Adam, AdamW i Adadelta, que han estat escollits degut a la seva semblança i eficàcia demostrada en aplicacions del mateix àmbit.



(a) Mètrica IoU (b) Funció de pèrdua

Figure 16: Mètriques d'encert en fase de segmentació per diferents optimitzadors

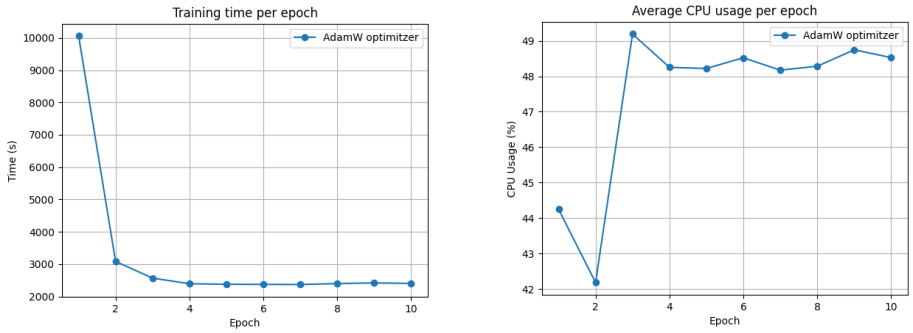
A la llum dels resultats obtinguts, tot i que les corbes són semblants, ens decantarem lleguerament per un model amb els paràmetres de la taula 1.

Parameter	**Value**
Epochs	4
Batch size	4
Learning rate	0.001
Optimizer	AdamW

Table 1: Paràmetres del model

4.2.1 Mètriques computacionals

Un altre camp molt important a l'hora de desenvolupar un projecte d'aquest estil i magnitud, és estudiar-ne les mètriques computacionals resultants de totes les nostres execucions. L'avaluació final de les imatges, on simplement s'han de carregar els models ja entrenats, no mostren un consum excessiu, i per tant no les estudiarem en profunditat. En canvi, per la fase d'entrenament, el consum de temps i recursos és altament significatiu.



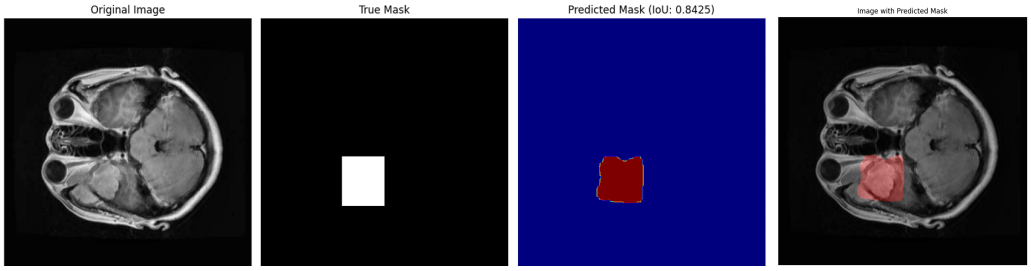
(a) Temps d'entrenament (b) Ús de la CPU

Figure 17: Mètriques computacionals en fase de segmentació

Sense tenir en compte el comportament anormal de la primera època per motius externs, veiem un temps d'entrenament i % d'ús de la CPU (*central processing unit*) constants, al voltant de 2400 segons per època i 48.5% respectivament.

4.2.2 Aplicació

Després d'haver dut a terme l'explicació detallada de tota la implementació i l'anàlisi de mètriques d'encert i computacionals, cal presentar el funcionament del model quan s'enfronta a una aplicació real. A mode d'exemple, es presentaran el cas d'un MRI pertanyent al *dataset* de la qual en tenim la màscara real per comparar (fig. 18).



(a) Comparació de màscares (b) Màscara predita

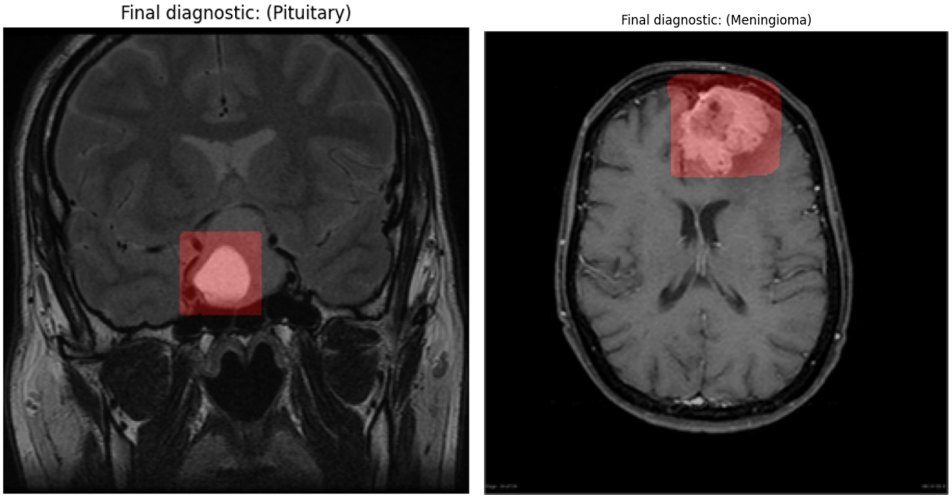
Figure 18: Exemple d'aplicació del model de segmentació

El que es mostra a les figures és un clar exemple de com treballa el model. Rep una imatge MRI d'entrada d'un pacient, la processa i analitza i retorna una màscara binària, la qual pot ser solapada amb menys opacitat sobre la imatge inicial i s'acaba obtenint la predicció clara de la localització del tumor.

4.3 Resultats del model compost

Finalment, ha arribat el moment de presentar el producte final d'aquest projecte: el model compost format pels dos models aplicats en sèrie. Un model que realitza seqüencialment una classificació de la imatge de ressonància magnètica i posteriorment, en segmenta la regió tumoral.

A les figures de continuació (19), es mostren dos exemples diferents de l'aplicació en sèrie del model.



(a) Exemple de pituitari (b) Exemple de meningioma

Figure 19: Exemples d'execució del model compost

5 Limitacions i possibles millores

En quant a limitacions, destaquem la impossibilitat de treballar amb el dataset sencer i amb les imatges amb la qualitat més alta que se'ns ofereix, ja que ara mateix ens veiem obligats a reduïr-ne tant la qualitat com la quantitat.

Ja havent realitzat tot el desenvolupament d'ambdós models i assegurant-nos bons resultats independents, el que ens ha quedat pendent és poder realitar els dos entrenaments dels models de manera paral·lela aprofitant els diferents nuclis dels que disposem. No obstant, dur això a terme ha presentat moltes complicacions inesperades, com bolcaments de memòria o pròpies limitacions computacionals dels dispositius utilitzats, bloquejos, sobrecalentaments o temps massa elevats degut a l'elevada càrrega de feina.

En conseqüència, una clara i viable millora seria la futura implementació d'un plantejament paral·lel general aprofitant la independència dels processos d'entrenament, amb funcions de llibreries semblants a les ja utiltizades pel processat d'imatges com *ProcessPoolExecutor*. Un cop ambdós models fossin entrenats i guardats per poder carregar-los posteriorment, el procediment seria el mateix que actualment: carregar-los alhora i fer passar una imatge pels dos models en sèrie per obtenir-ne el tipus de tumor i la localització sobre la MRI.

6 Conclusions

Finalment, es pot observar com la superposició dels dos models ha estat un èxit, permetent crear un model bastant robust que ens permet classificar i segmentar els tumors amb una precisió bastant alta. No obstant això, certes millores podrien ser incorporades per tal d'acabar de millorar el temps d'execució dels models, com podria ser l'aplicació del paral·lelisme o l'aplicació de tècniques més sofisitcades per tal d'acabar d'augmentar la precisió del model, especialment en el cas del tumor glioma.

References

[1] Parisa Karimi Darabi. Brain tumor image dataset : Semantic segmentation. https://www.kaggle.com/datasets/pkdarabi/brain-tumor-image-dataset-semantic-segmentation/data, 2023.

[2] Tridib Raj. Brain tumer detection densenet-201, and resnet50. https://www.kaggle.com/code/tridibraj/brain-tumer-detection-densenet-201-and-resnet50/notebook, 2024.