APLICACIÓ DE MODELS DE DEEP LEARNING EN SEGMENTACIÓ D'IMATGES

TREBALL FI DE GRAU

Autor: Enric Sena Alvarez

Supervisor: Toni Lozano Bagen

Grau en Estadística Aplicada

Curs 2023-2024

Bellatera, 14 de Juny de 2024

Resum

Aquest estudi explora l'impacte que actualment està causant la intel · ligència artificial especialment en l'indústria de l'automòbil, aconseguint així mitjançant algoritmes i el correcte tractament de dades proporcionar certa autonomia al vehicle per tal de reduir la sinistralitat. En concret s'han entrenat models de xarxes neuronals amb diverses profunditats de capes amb la intenció de trobar-ne un que tingui un rendiment adequat i sigui capaç de segmentar imatges de trànsit de manera que discrimini de la imatge quines parts són carretera i quines són altres vehicles, vorals o bé altres elements adjacents. Per tal de millorar la predicció del model, s'ha utilitzat una tècnica de visió per computador de manera que s'han obtingut resultats amb una millora significativa de la qualitat de les imatges.

Agraïments

Agraeixo al meu tutor, Toni Lozano, pel suport i l'acompanyament durant el desenvolupament del treball. La seva tutoria ha estat fonamental, guiant-me i ajudant-me a resoldre i entendre els problemes que anaven sorgint a mesura que anava avançant la investigació.

Voldria mencionar també el meu agraïment a totes aquelles persones que m'han donat suport durant aquest temps i en especial a la meva família i amics. Moltes gràcies!

Índex

1 Introducció

En els últims anys s'ha viscut un augment exponencial de la intel · ligència artificial tant a escala mediàtica com en l'àmbit tècnic. En aquest treball es tractarà una tipologia concreta de models que hi ha darrere d'aquesta nova eina la qual permet tractar amb una tipologia de dades diferent de les tradicionals.

Fins no fa massa temps, les dades que es podien fer servir per desenvolupar models estadístics eren dades estructurades com, per exemple, les bases de dades relacionals. El principal avantatge que aporten els models d'aprenentatge profund és que permeten tractar amb dades no estructurades com poden ser fitxers d'àudio, textos o imatges que és el cas d'estudi d'aquest treball. En aquest treball es tractarà de desenvolupar 3 models diferents amb diverses profunditats de capes i complexitat, dels quals s'estudiarà les seves mètriques d'entrenament i validació amb les que es compararan i s'escollirà el model que millor funcioni.

La finalitat d'aquest treball és explicar com s'utilitzen en l'actualitat els models d'aprenentatge automàtic a la indústria i, particularment, es tractarà un problema que no és trivial com és l'aplicació d'aquests models a la indústria de l'automòbil i com aquests models han incrementat la seguretat viària, ja que són capaços de mitjançant una càmera situada al davant del vehicle, diferenciar la carretera respecte d'altres vehicles i de l'entorn. Aquest fet permet en part, eliminar els accidents derivats de la fatiga del conductor perquè el propi vehicle és capaç de mantenir-se a la calçada.

En aquest informe s'explicarà l'evolució de les xarxes neuronals partint dels exemples i estructures més senzilles fins als models de xarxes neuronals convolucionals utilitzats per resoldre el problema anteriorment mencionat. Finalment, es mostraran els resultats obtinguts i es mostrarà quin és l'ús a la indústria d'aquest tipus d'arquitectures en àmbits diferents dels de l'automoció.

2 Metodologia

2.1 Introducció a les xarxes neuronals

Avui en dia es parla molt d'aprenentatge automàtic i sobre el potencial que té per millorar segons quins processos o, fins i tot, simplificar-ne d'altres i així millorar la qualitat de vida de les persones. Potser, les persones amb més curiositat sobre aquest àmbit i que han fet recerca sobre quins processos hi ha darrere d'aquesta popular idea d'intel · ligència artificial (d'ara endavant, IA) i s'han trobat que hi ha diferents arquitectures les quals serveixen per resoldre diferents tipologies de problemes. Tanmateix, és possible que algú amb certa curiositat sobre l'àmbit, tingui coneixement de les tècniques que hi ha darrere de cada arquitectura que són les xarxes neuronals. Doncs, en aquest apartat s'explicarà quin ha estat el procés de desenvolupament d'aquesta tecnologia partint des de la seva unitat més senzilla: la neurona.

La base del funcionament d'una neurona artificial és intentar aproximar-se al màxim al funcionament d'una neurona humana (7). De fet, aquesta idea de Frank Rosenblatt del 1958 recollia les investigacions de Warren McCulloch (matemàtic) i Walter Pitts (neuropsicòleg) del 1943 on en una publicació (8) van descriure el funcionament de les neurones del cervell i van modelar una primera xarxa neuronal fent servir circuits elèctrics.

Tornant al descobriment de Rosenblatt, la seva diferència significativa va ser el fet d'introduir pesos a les entrades de la neurona, de manera que aquesta era capaç de generar tantes respostes com entrades tenia. Per facilitar la comprensió d'aquest apartat clau es pot observar a la Figura 1.

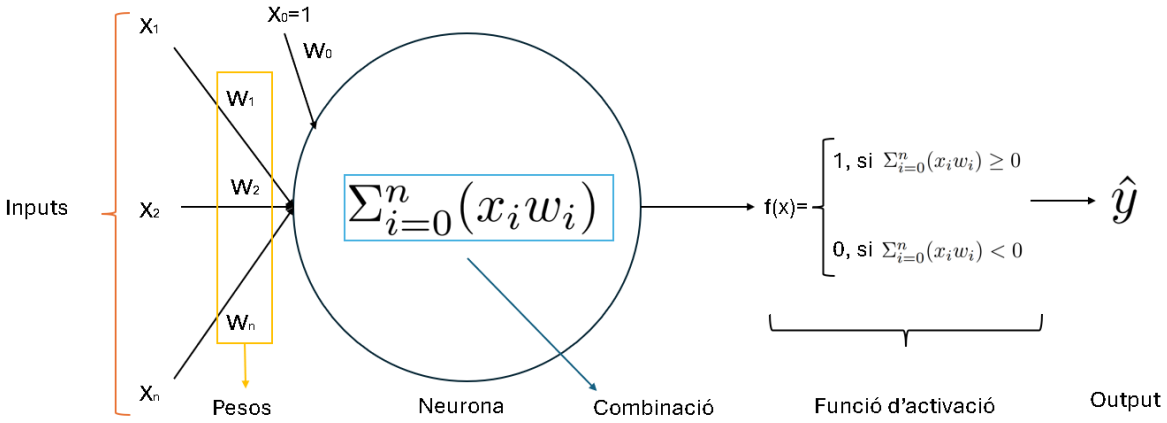

Figura 1: Gràfic d'una neurona simple

Es pot observar les diferents parts que formen la neurona. D'esquerra a dreta, primer element són els inputs que són les dades d'entrada de la neurona. A part dels inputs hi ha uns valors que no es poden modificar i que són intrínsecs de cada neurona i s'anomena biaix. Val a dir, que el format dels valors d'entrada sempre és numèric.

Seguidament, hi ha els pesos: aquests valors permeten a la neurona assignar certa importància a cada dada d'entrada. Tot seguit, dintre de la neurona es troba la combinació, aquesta és una operació matemàtica. Per exemple, els possibles càlculs podrien ser els que figuren a la Taula 1:

Combinació	**Expressió Matemàtica**	**Descripció**
Suma Ponderada	$z(x) = \sum_{i=1}^{n} w_i x_i$	La suma ponderada dels inputs amb pesos.
Màxim	$z(x) = \max(w_1 x_1, ..., w_n, x_n)$	El valor màxim dels inputs.
Mínim	$z(x) = \min(w_1 x_1, ..., w_n, x_n)$	El valor mínim dels inputs.
AND	$z(x) = w_1 x_1 \wedge ... \wedge w_n x_n$	Operadors AND ($\wedge$) entre els inputs.
OR	$z(x) = w_1 x_1 \vee ... \vee w_n x_n$	Operadors OR($\vee$) entre els inputs.

Taula 1: Principals combinacions d'inputs d'una neurona.

Tornant al gràfic de la neurona (Figura 1), el següent element que s'observa és la funció d'activació. Aquesta funció és clau en el procés, ja que és la que determinarà quin format tindrà la sortida de la neurona. Les principals funcions d'activació es poden veure a la Taula 2.

Funció d'Activació	**Expresió Matemàtica**	**Descripció**
Step Function	$f(x) = \begin{cases} 1 & \text{si } x \geq 0 \\ 0 & \text{si } x < 0 \end{cases}$	Activa la neurona si l'entrada es major o igual a zero, sinó, la desactiva.
Linear	$f(x) = x$	Funció lineal on la sortida és proporcional a l'entrada.
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$	Funció sigmoide que transforma l'entrada en un valor entre 0 y 1.
Tangent Hiperbòlica	$f(x) = \tanh(x) = \frac{sin(x)}{cos(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	Transforma l'entrada en un valor entre -1 i 1.
ReLU (Rectified Linear Unit)	$f(x) = \max(0, x)$	Només activa la neurona si l'entrada és positiva, del contrari val 0.
Softmax	$f(x_i) = \frac{e^{x_i}}{\sum_j^n e^{x_j}}$	Converteix els valors d'entrada en una distribució de probabilitat.

Taula 2: Funcions d'activació més comuns en xarxes neuronals.

Un cop s'ha exposat el naixement i funcionament de les neurones, es pot començar a intuir la versatilitat d'aquesta tecnologia. El següent pas en l'evolució de la IA va ser el fet de construir una arquitectura que combinés diferents neurones de manera que van sorgir els conceptes com:

1. **Capes d'entrada**: Capa que rep els inputs de les dades.
2. **Capes ocultes**: Capes internes del model.
3. **Capa de sortida**: Capa que retorna els outputs de la xarxa.

En aquest cas, tècnicament no es podria nomenar xarxa neuronal sinó, que va sorgir com a perceptró multicapa (Figura 2).

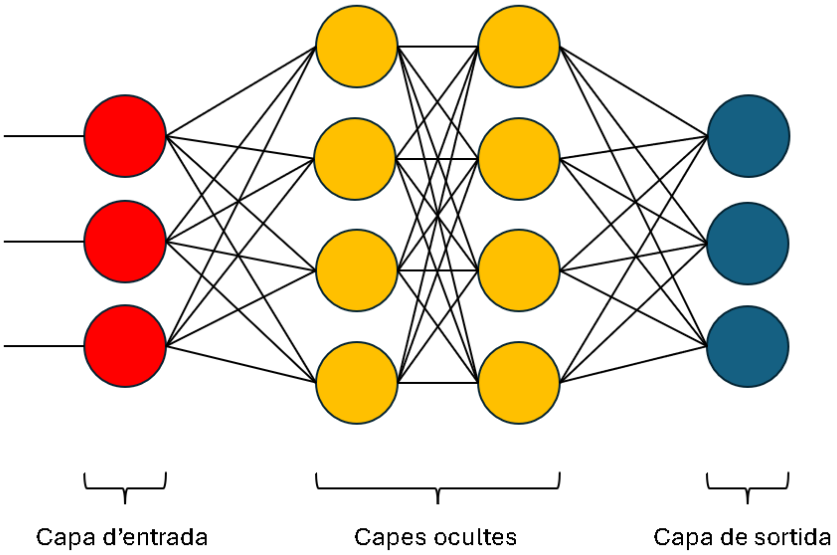

Figura 2: Perceptró multicapa

Ara que ja s'ha vist quina és la unitat bàsica de funcionament d'una xarxa neuronal, cal comentar quins són els diferents tipus d'estructures o arquitectures que es poden utilitzar, ja que cadascuna d'elles permet resoldre una tipologia de problemes diferent segons quin resultat es volgui obtenir (Figura 34).

Primerament, cal destacar que al Perceptró multicapa és quasi idèntic a l'estructura més bàsica d'una xarxa neuronal, les anomenades "Fully Connected". L'únic fet que les diferencia és el mètode d'entrenament i és que aquestes últimes, fan servir el mètode de retropropagació (backpropagation). Igualment en ambdós casos totes les neurones de l'arquitectura estan connectades amb les de la següent capa.

Els principals problemes que poden resoldre les xarxes neuronals Fully Connected són, per exemple: Classificació (Binària, Multi-classe, Multi-etiqueta), Regressió; Reconeixement de patrons; Series temporals i Aproximació de funcions matemàtiques. Aquest tipus d'arquitectura funciona molt eficientment quan les dades d'entrada tenen format de vector, ja que degut a les múltiples connexions entre les neurones, si l'entrada és un altre format de dades com un tensor, s'observarà un augment significatiu en el temps d'execució. Més endavant, es comentarà quina és l'arquitectura més adient per treballar amb tensors.

Una altra tipologia d'arquitectura d'un model són les xarxes neuronals recurrents. Aquestes es poden classificar en 3 subcategories: les Vainilla, Gated Recurrent Unit (GRU) i Long Short Term Memory Unit (LSTM). Tant la variant GRU com LSTM no s'explicaràn en profunditat, ja que no són l'objecte d'aquest treball, però és necessari comentar-les, ja que permeten millorar les prediccions de series temporals, per exemple, donant més importància a les dades més recents a l'hora de predir els valors futurs.

El gran avantatge que ha suposat les xarxes neuronals recurrents és en l'àmbit del processament del llenguatge natural pel fet que permeten establir certa relació entre cada dada d'entrada (en aquest cas, les lletres de les paraules) i a partir d'aquí són capaces d'"entendre"textos.

És necessari destacar que arrel dels avenços tecnològics dels últims anys, han sorgit infinitat de noves arquitectures cada una més complexa que l'anterior, les quals permeten resoldre una mena de tasques cada vegada més complexes. Un bon exemple d'aquestes xarxes són les transformers, les quals han permés desenvolupar grans models de llenguatge capaços de, mitjançant certes expressions i estructures de frases, fer creure a un lector no expert que entenen sobre el tema que estan escrivint.

Pel que fa a les xarxes neuronals especialitzades a treballar amb tensors com a dades d'entrada, aquestes s'anomenen Xarxes Neuronals Convolucionals. Aquests models són claus perquè permeten treballar amb imatges gràcies al fet que aquestes se li poden passar a les neurones d'entrada com a tensors. Per exemple, imaginem una imatge qualsevol a color: En realitat el que estem veient són combinacions de 3 colors, vermell, verd i blau (RGB en anglès) i, per tant, podem pensar la imatge com 3 matrius amb la mateixa mida sobreposades formant la imatge. Aquesta tipologia de dades s'anomena tensor.

2.2 Entrenament d'una Xarxa Neuronal

Un cop s'ha explicat com sorgeixen i quins grans grups de models hi ha i en concret quina tipologia s'utilitzarà per resoldre el problema plantejat en aquest treball, el següent pas és explicar com s'entrena un model.

La fase d'aprenentage d'un model és a grans trets conceptualment, igual en la majoria de models, des d'una regressió lineal fins un model de classificació d'imatges, passant per un model de seguiment d'infecció d'una població.

Primerament, cal assegurar-se de tenir unes dades consistents, netes i de qualitat, ja que, sense cap d'aquests tres aspectes, el model "aprendrà"de manera errònia. En el cas d'aquest estudi, les dades compleixen aquestes condicions ja que pertanyen a un dataset publicat a Kaggle.

Les dades utilitzades en aquest estudi són imatges de trànsit de vehicles obtingudes a partir d'una càmera instal · lada al front del cotxe.

En quant a les dades de resposta que se li entren al model, es tracten de màscares de cada imatge de manera que la xarxa neuronal rep cada imatge amb la seva màscara corresponent i intenta trobar-ne patrons sobre com es relacionen.

Es pot veure un exemple a la Figura 5

Figura 3: Exemple d'imatge

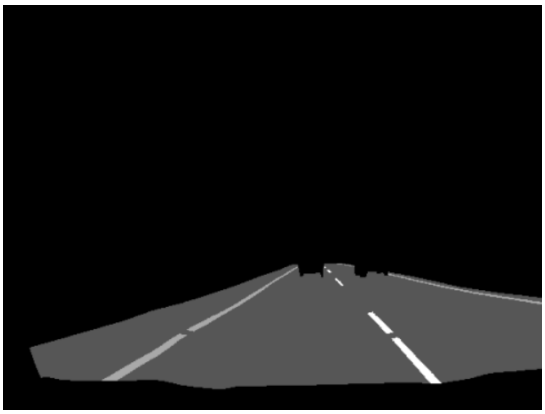

Figura 4: Exemple de màscara

Figura 5: Mostra de les dades

D'ara endavant, per referir-se a les imatges, es parlarà de conjunt X, ja que són les dades d'entrada, i en contraposició, per fer referència a les màscares, s'utilitzarà conjunt Y, ja que és la resposta del model a la que es vol arribar. El nombre total d'imatges de les que es disposen són 374 amb les seves màscares corresponents.

2.2.1 Preprocès de les dades

Aquest apartat és clau per tal de garantir un correcte aprenentatge per part del model, perquè com s'ha vist anteriorment, les funcions d'activació funcionen de manera òptima amb valors al voltant del zero (Com a questió prèvia, s'ha de dir que s'entén com a funcionament òptim, una correcta convergència de les neurones de la xarxa).

Hi ha una part del preprocés que és comú en ambdós conjunts i és el fet d'assegurar que tant les imatges com les màscares, tenen les mateixes dimensions (entenent que les imatges tenen codificació de color RGB i les màscares estan codificades com a escala de grisos). Un cop s'ha harmonitzat la mida de les dades, cal assegurar que els valors dels píxels de les màscares siguin valors entre zero i u, per tant, cal normalitzar els valors. Finalment, després d'aquest pas, s'obté que les imatges tenen dimensió (444,590,3); és a dir, tenen 444 píxels d'alçada, 590 d'amplada i 3 canals de color (RGB).

Si es comproven les dimensions de les màscares, aquestes tenen mida (444,590,1). Com s'ha explicat, s'observa que les dimensions d'alçada i amplada són idèntiques a les de les imatges i, en canvi, només tenen un únic canal de color perquè estan codificades en escala de grisos.

Un cop les dades d'entrada estan preparades en el format adequat per passar-les pel model, cal definir quina estratègia se seguirà per entrenar-lo.

2.2.2 Partició de les dades

Per entrenar els diferents models que s'han valorat en aquest estudi, s'ha optat per definir 2 subconjunts de dades i que inclouen tant el conjunt X com l'Y ja que són dades aparellades). S'ha fet una partició d'entrenament amb el 80% de les dades i el 20% restant s'ha destinat a validar el model. Així doncs, el model no "veurà" les imatges ni les màscares i realitzarà una predicció que serà comparada amb la màscara real de manera que permetrà saber com de bé funciona aquest).

2.2.3 Hiperparàmetres del model

En un model hi ha una sèrie de valors que controlen i garanteixen en certa mesura el correcte funcionament d'aquest. Aquests valors s'anomenen hiperparàmetres i tenen aquesta nomenclatura degut a que el model no és capaç de modificar-los i per tant venen donats per l'investigador.

Abans de continuar, cal introduïr el primer concepte d'hiperparàmetre en quant a la validació del funcionament del model ja que és l'indicador que ens permet saber realment quin és el rendiment del model real. Això es deu a que utilitza el subconjunt de validació mencionat anteriorment i retorna l'evolució d'aquest indicador. Els indicadors més coneguts els podem observar a la Taula4 (Veure Annex)

En el cas d'estudi d'aquest treball, no s'ha utilitzat cap de les mètriques anteriors tot i ser les més populars, ja que el que retorna la Xarxa Neuronal són imatges (les màscares) i per tant, cal utilitzar una mètrica capaç de recollir la informació i comparar-la respecte els valors reals.

La mètrica utilitzada s'anomena Dice Coefficient(10) i a la Taula 3 es poden observar les seves característiques.

Mètrica	Descripció
DICE Coefficient	El coeficient DICE, també conegut com a coeficient de Sørensen-Dice, és una mètrica utilitzada en problemes de segmentació d'imatges i en la validació de resultats de la segmentació. Mesura la similitud entre dues mostres i s'utilitza principalment per avaluar la superposició o la concordança entre la segmentació automàtica i la segmentació de referència. Es calcula mitjançant la fórmula: $$DICE = \frac{2 \times \lvert A \cap B \rvert}{\lvert A \rvert + \lvert B \rvert}$$ on A és el conjunt de pixels de la segmentació automàtica i B és el conjunt de pixels de la segmentació de referència. El coeficient DICE proporciona un valor entre 0 i 1, on 1 indica una concordança perfecta entre les dues segmentacions. Quan el valor s'apropa a 0, indica una falta de concordança.

Taula 3: Descripció del coeficient DICE

A continuació a la Figura 6 es pot observar de manera visual el funcionament d'aquesta mètrica.

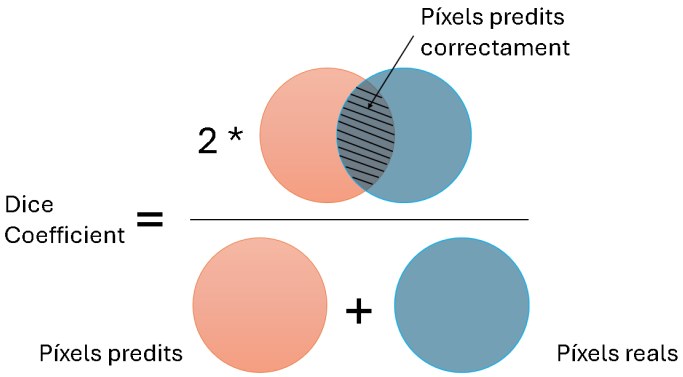

Figura 6: Dice Coefficient

És important destacar que en aquesta mètrica s'ha establert un llindar arbitrari mitjançant el qual es converteixen tant les màscares predites com les reals a imatges en colors blanc o negre (0 o 1).

Més endavant es demostrarà que aquest llindar no afecta a l'entrenament del model i és només per conèixer com rendeix el model durant l'entrenament.

Un altre hiperparàmetre que se li passa al model és quin optimitzador ha d'utilitzar. Un optimitzador serà l'algorisme que s'encarregarà d'indicar-li al model com de bé esta "aprenent"de les dades. En altres paraules, és l'encarregat de la convergència de la Xarxa Neuronal a cada iteració.

Particularment, en aquest estudi s'ha utilitzat l'Adam (Adaptative Moment Estimation)(11) el qual combina les millors característiques de l'algorisme Momentum (12) i del RMSprop (13).Aquests algorismes tenen la finalitat minimitzar la funció de pèrdua (Loss Function en anglés); més endavant s'aprofundirà més sobre el seu funcionament.

La idea principal de l'Adam és que és capaç d'ajustar la taxa d'aprenentatge segons convingui a cada iteració. És un gran avenç, ja que permet seguir el procés del descens del gradient de la forma més òptima i minimitzar l'error de quedar-se "estancat"en un mínim local de la funció.

L'Adam està format per diferents etapes que s'esmenen a continuació:

1. **Inicialització:** Inicialitza els moments de primer ordre m_0, de segon ordre v_0 i de temps (t) a zero mentre que els valors β_1, $\beta_2 \in [0,1)$ ja que són les taxes de decreixement exponencial per les estimacions dels moments. Finalment cal indicar que la funció $\mathcal{L}$ és la funció objectiu o de pèrdua.

2. **Càlcul del Gradient:** Calcula el gradient dels paràmetres del model respecte a la funció de pèrdua g_t.
$$g_t = \nabla\mathcal{L} \cdot (\theta_{t-1})$$

3. **Actualització del Moment de Primer Ordre (m):**
$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$$

4. **Actualització del Moment de Segon Ordre (v):**
$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$$

5. **Correcció del Biaix del Moment de Primer Ordre ($\hat{m}_t$):**
$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

6. **Correcció del Biaix del Moment de Segon Ordre ($\hat{v}_t$):**
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

7. **Actualització dels Paràmetres del Model:**
$$\theta_t = \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

8. **Repetir:** Repeteix els passos 2-7 per a cada iteració fins que el model convergeixi.

Com es pot observar, al ser un procés iteratiu aquest es va executant fins que el model convergeixi. És per això que cal trobar l'equilibri entre un optimitzador que sigui ràpid i alhora capaç d'adaptar-se a la "superfície"de la funció de pèrdua per trobar la direcció de màxim descens del gradient i arribar tan aviat com sigui possible a una solució. Per entendre el perquè és tan important utilitzar un bon optimitzador, a continuació, a la Figura 7, es pot observar un exemple de funció de pèrdua.

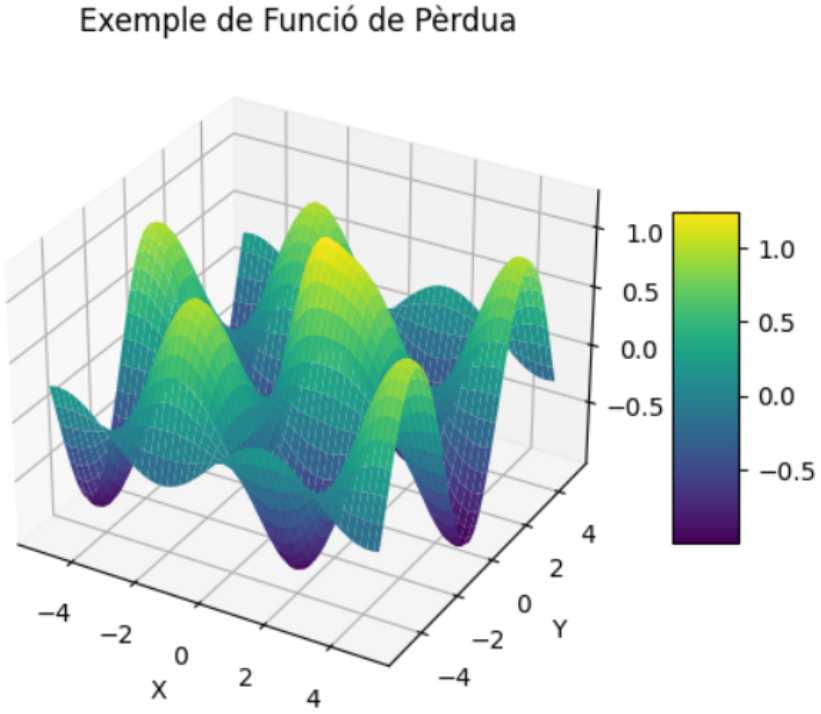

Figura 7: Exemple de Funció de Pèrdua

Seguint l'explicació de l'optimitzador on s'ha mencionat en diverses vegades la funció de pèrdua, ara s'exposarà en què consisteix aquesta.

Quan es parla de funció de pèrdua, també coneguda com funció objectiu o en anglès, loss function. Hom es refereix a una funció que mesura la diferència entre les prediccions del model i els valors reals. Aquesta informació l'utilitza la Xarxa Neuronal per veure si està convergint correctament i, per tant, està "aprenent" correctament els patrons i relacions entre les imatges i les seves màscares corresponents.

L'objectiu principal del model és trobar aquells "pesos"que minimitzen la funció de pèrdua, el que es tradueix en una millora de les prediccions que aquest produeix.

En el cas d'aquest estudi, s'ha utilitzat una funció prou coneguda dintre de l'àmbit de les Xarxes Neuronals el nom de la qual en anglès és Binary Crossentroy (en català es podria traduir com Entropia Creuada Binària) i està dissenyada per problemes on el retorn són valors entre 0 i 1, per tant, és l'adequada per utilitzar en aquest estudi.

La funció es defineix amb la següent fórmula:

$$\mathcal{L}(y, \hat{y}) = -\left(y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})\right)$$

Es pot observar que la fórmula anterior, té dues variables, y i $\hat{y}$ on y és l'etiqueta verdadera de la màscara i $\hat{y}$ és el valor predit pel model.

Aquesta Funció de Pèrdua, penalitza significativament les prediccions incorrectes (les que tenen una major distància entre el valor real i el predit). En canvi, quan la predicció s'apropa al valor real, resulta que el valor de la funció tendeix a zero i, per tant, indica que el model s'està entrenant correctament.

Un dels grans problemes que poden sorgir alhora d'entrenar un model qualsevol (no sols succeeix amb les Xarxes Neuronals), és el fet que aquest s'adapti massa bé a les dades d'entrenament i, per tant, en el moment de comprovar el seu rendiment amb el conjunt de validació, s'observi que no funciona correctament.

Per tal d'impedir aquest sobreajustament s'ha utilitzat un paràmetre nomenat en anglès Early Stopping el qual s'encarrega de monitorar l'evolució d'alguna mètrica de les que proporciona el model a cada iteració d'aquest. Particularment s'ha utilitzat com a mètrica a monitorar, la Funció de Pèrdua en el conjunt de validació i s'ha definit de la següent manera: Si després de 5 iteracions (epochs en termes de Xarxes Neuronals), la Funció de Cost no millora (en altres termes, si la diferència entre les prediccions i els valors reals no es veu reduïda), s'ha ordenat al model que s'aturi. El motiu és que s'entén que no té prou capacitat per a captar més informació de les dades i, per tant, la seva capacitat predictiva a partir d'aquest punt si continués entrenant-se només disminuiria.

Uns altres hiperparàmetres importants en el moment d'entrenar un model són les Epochs i el Batch Size. En el cas d'una Epoch, es refereix a una iteració completa del model sobre les imatges d'entrenament, de manera que "observa"totes les imatges. Després de cada Epoch és quan el model reajusta els pesos segons el valor de la funció de pèrdua obtinguda.

Quant al Batch Size és un paràmetre que defineix el nombre de mostres d'entrenament de manera que en comptes de passar per tot el conjunt de dades d'entrenament abans d'actualitzar els pesos segons la funció de pèrdua, aquest selecciona un subconjunt de les dades a fi de que el model s'actualitzi quan acaba cada Batch i no només quan acaba cada Epoch.

És important destacar que per defecte, els 3 models s'han entrenat establint el mateix nombre d'Epochs en 50 (per si l'Early Stopping no ha actuat perquè l'entrenament, s'estava duent a terme de manera correcta) i d'igual manera s'ha fet amb el Batch Size assignant-li un valor de 8 perquè per motius computacionals, en augmentar-lo, no es disposava de suficient memòria i capacitat de computació per a carregar més imatges simultàniament. Això passa perquè les imatges tenen unes dimensions que a priori poden semblar petites, però computacionalment requereixen molta capacitat de processament de càlcul. També cal destacar que s'ha hagut d'utilitzar l'editor de Kaggle, perquè és el que més capacitat de càlcul podia aportar aprofitant que les dades també eren de la plataforma.

2.3 Models utilitzats

En aquesta secció s'explicaràn els diferents models plantejats per resoldre el problema d'aquest estudi. Primerament cal destacar que, com s'ha mencionat anteriorment, els 3 models són Xarxes Neuronals Convolucionals i tenen variacions en la profunditat de capes i complexitat d'aquestes. Més endavant es detallaràn les diferències entre aquests.
És important remarcar que tal i com s'ha comentat anteriorment, els models plantejats tenen tots els mateixos hiperparàmetres vistos en l'apartat2.2.3 i en cas contrari s'indicarà quina modificació s'ha dut a terme. Hi ha una sèrie de conceptes clau que cal mencionar per tal de garantir l'enteniment de les estructures dels models ja que en major o menor mesura els 3 tenen aquestes tipologies de capes.

1. **Padding**: S'afegeixen píxels extra (normalment zeros) al voltant de les imatges per assegurar que les convolucions mantinguin la mida de l'input.

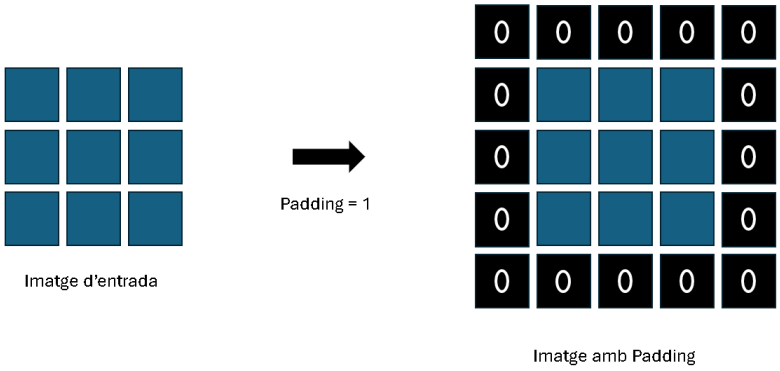

Figura 8: Exemple de Padding

2. **Max Pooling**: Redueix la resolució de la imatge seleccionant el valor màxim en regions no solapades d'aquesta.

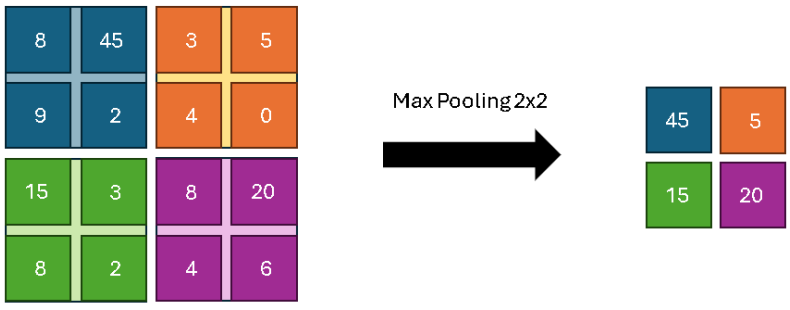

Figura 9: Exemple de Max Pooling 2x2

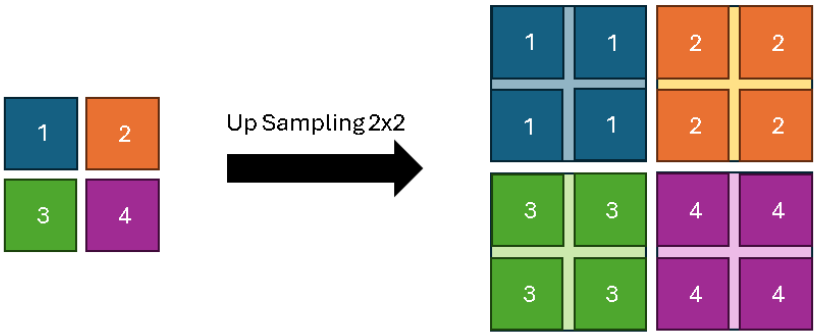

Figura 10: Exemple d'Up Sampling 2x2

3. **Up Sampling**: Augmenta la resolució de les imatges duplicant els píxels.
4. **Batch Normalization**: Millora la velocitat i estabilitat de l'entrenament normalitzant les activacions de les capes intermèdies.
5. **ReLU Activation**: Introdueix no linealitat en el model, establint a zero tots els valors negatius.

2.3.1 Model 1

El primer Model, igual que els altres 2 te arquitectura d'autoencoder, això significa que consta de dues parts diferenciables. Una primera part nomenada Encoder, s'encarrega de reduir la imatge per quedar-se només amb les dades "essencialsï l'altre part de la Xarxa Neuronal nomenada Decoder s'encarrega de tornar al tamany original la imatge i fer les prediccions corresponents.

L'estructura d'aquest model consta de les següents capes:

1. **Dimensions de les imatges (input_shape)**: Les dimensions de les imatges d'entrada són de 444x590 píxels amb 3 canals de color (RGB).
2. **Número de Filtres (num_filters)**: El nombre de filtres utilitzats en les capes convolucionals és de 32.
3. **Encoder**
   (a) **Primera Capa Convolucional (Conv2D)**
      i. Filtres: 32
      ii. Mida del Nucli: 3x3
      iii. Activació: ReLU (Veure Taula 2)
      iv. Padding: Same (per mantenir les dimensions de la imatge)
      v. Dimensions de les imatges: (444, 590, 3)
   (b) **BatchNormalization**: Aquesta capa normalitza les dades de sortida de les funcions d'activació per millorar la velocitat i estabilitat de l'entrenament.
   (c) **Activació ReLU**: ReLU s'utilitza per introduir no linealitat en el model. (Veure Taula 2)
   (d) **MaxPooling2D**
      i. Mida: 2x2
      ii. Stride: 2x2
      iii. Aquesta capa redueix la resolució de la imatge seleccionant el valor màxim en regions de 2x2 píxels. (Veure Figura 9)
4. **Decoder**
   (a) **Segona Capa Convolucional (Conv2D)**
      i. Filtres: 32
      ii. Mida del Nucli: 3x3
      iii. Activació: ReLU (Veure Taula 2)
      iv. Padding: Same (per mantenir les dimensions de la imatge)
   (b) **BatchNormalization**: Igual que anteriorment, normalitza les dades de sortida de les funcions d'activació.
   (c) **Activació ReLU**: Introducció de no linealitat.
   (d) **UpSampling2D**

i. Mida: 2x2
ii. Aquesta capa augmenta la resolució de la imatge duplicant els píxels en cada dimensió. (Veure Figura 10)

(e) **Capa Final Convolucional (Conv2D)**
i. Filtres: 1
ii. Mida del Nucli: 3x3
iii. Activació: Sigmoid (Veure Taula 2)
iv. Padding: Same (per mantenir les dimensions de la imatge)
v. Aquesta capa genera la imatge de sortida (màscara) i sols conté valors entre 0 i 1.

5. **Compilació del Model**

(a) Optimitzador: Adam 11 (Veure apartat 2.2.3)
(b) Funció de Pèrdua: Binary Crossentropy (Veure apartat 2.2.3)
(c) Mètrica: Dice Coefficient. (Veure Figura 10)

Per entendre de manera més visual quina és l'estructura del model podem observar la Figura 11.

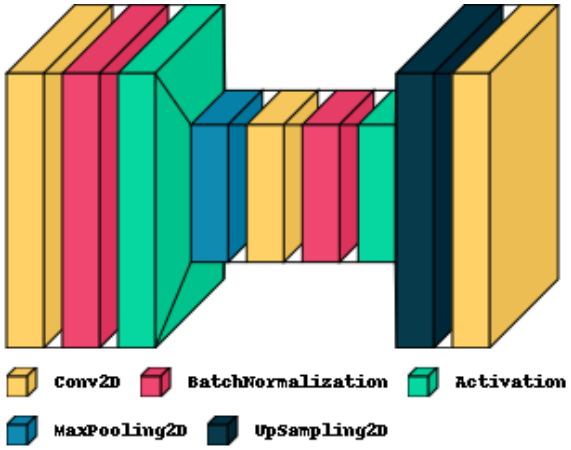

Figura 11: Visualització de les capes del Model 1

Si es vol veure amb més detall les dimensions de cada capa del model, es pot consultar la Figura35 a l'Annex (5). Com es pot veure a la Figura 11, i al contrari del que pot semblar per l'explicació detallada el model és bastant simple en comparació als altres que s'explicaran a continuació.

2.3.2 Model 2

En quant al model 2, segueix tenint arquitectura d'autoencoder i és una versió més complexa (amb més capes) que l'anterior. A continuació s'explica la seva estructura:

1. **Dimensions de les imatges (input_shape)**: Les dimensions de les imatges d'entrada és de 444x590 píxels amb 3 canals de color (RGB).
2. **Número de Filtres (num_filters)**: El nombre de filtres utilitzats en les capes convolucionals és de 32.
3. **Encoder**
   (a) **Primera Capa Convolucional (Conv2D)**
      i. Filtres: 32
      ii. Mida del Nucli: 3x3
      iii. Activació: ReLU (Veure Taula 2)
      iv. Padding: Same (per mantenir les dimensions de la imatge)
      v. Dimensions de les imatges: (444, 590, 3)
   (b) **BatchNormalization**: Aquesta capa normalitza les dades de sortida de les funcions d'activació per millorar la velocitat i estabilitat de l'entrenament.
   (c) **Activació ReLU**: ReLU s'utilitza per introduir no linealitat en el model.
   (d) **MaxPooling2D**
      i. Mida: 2x2
      ii. Stride: 2x2
      iii. Aquesta capa redueix la resolució de la imatge seleccionant el valor màxim en regions de 2x2 píxels.
   (e) **Capa Convolucional Addicional (Conv2D)**
      i. Filtres: 64 (2*num_filters)
      ii. Mida del Nucli: 3x3
      iii. Activació: ReLU
      iv. Padding: Same (per mantenir la mida espacial de la imatge)
   (f) **BatchNormalization**: Igual que anteriorment, normalitza les dades de sortida de les funcions d'activació.
   (g) **Activació ReLU**: Introducció de no linealitat.
4. **Decoder**
   (a) **Primera Capa Convolucional (Conv2D)**
      i. Filtres: 64 (2*num_filters)
      ii. Mida del Nucli: 3x3
      iii. Activació: ReLU
      iv. Padding: Same (per mantenir la mida espacial de la imatge)
   (b) **BatchNormalization**: Igual que anteriorment, normalitza les dades de sortida de les funcions d'activació.

(c) **Activació ReLU**: Introducció de no linealitat.

(d) **Capa Convolucional Transposada (Conv2DTranspose)**

i. Filtres: 32

ii. Mida del Nucli: 3x3

iii. Activació: ReLU

iv. Padding: Same (per mantenir la mida espacial de la imatge)

(e) **BatchNormalization**: Igual que anteriorment, normalitza les dades de sortida de les funcions d'activació.

(f) **Activació ReLU**: Introducció de no linealitat.

(g) **UpSampling2D**

i. Mida: 2x2

ii. Aquesta capa augmenta la resolució de la imatge duplicant els píxels en cada dimensió. (Veure Figura 10)

(h) **Capa Final Convolucional (Conv2D)**

i. Filtres: 1

ii. Mida del Nucli: 3x3

iii. Activació: Sigmoid (Veure Taula 2)

iv. Padding: Same (per mantenir la mida espacial de la imatge)

v. Aquesta capa genera la imatge de sortida amb una sola banda de valors entre 0 i 1.

5. **Compilació del Model**

(a) Optimitzador: Adam 11 (Veure apartat 2.2.3)

(b) Funció de Pèrdua: Binary Crossentropy (Veure apartat 2.2.3)

(c) Mètrica: Dice Coefficient. (Veure Figura 10)

A continuació es pot observar una imatge representativa de l'estructura del model (Veure Figura 12). En cas de voler un diagrama més detallat, consultar Figura 36 a l'Annex (5).

Figura 12: Visualització de les capes del Model 2

2.3.3 Model 3

Finalment s'ha entrenat el model 3 que és el més complex dels anteriors ja que s'ha augmentat considerablement el nombre de capes per tal d'observar quin era el rendiment d'aquest model amb les dades del estudi. Més endavant s'explicarà amb més detall però aquest model servirà per demostrar que no sempre el model més gran i que més triga en entrenar-se és el millor.

A continuació s'esmenten les capes d'aquest model:

1. **Dimensions de les imatges (input_shape)**: Les dimensions de les imatges d'entrada són de 444x590 píxels amb 3 canals de color (RGB).
2. **Número de Filtres (num_filters)**: El nombre de filtres utilitzats en les capes convolucionals és de 64.
3. **Encoder**
   (a) **Primera Capa Convolucional (Conv2D)**
      i. Filtres: 64
      ii. Mida del Nucli: 3x3
      iii. Activació: ReLU (Veure Taula 2)
      iv. Padding: Same (per mantenir les dimensions de la imatge)
      v. Dimensions de les imatges: (444, 590, 3)
   (b) **BatchNormalization**: Aquesta capa normalitza les dades de sortida de les funcions d'activació per millorar la velocitat i estabilitat de l'entrenament.
   (c) **Activació ReLU**: ReLU s'utilitza per introduir no linealitat en el model.
   (d) **Segona Capa Convolucional (Conv2D)**
      i. Filtres: 64
      ii. Mida del Nucli: 3x3
      iii. Activació: ReLU
      iv. Padding: Same (per mantenir les dimensions de la imatge)
   (e) **BatchNormalization**: Igual que anteriorment, normalitza les dades de sortida de les funcions d'activació.
   (f) **Activació ReLU**: Introducció de no linealitat.
   (g) **MaxPooling2D**
      i. Mida: 2x2
      ii. Stride: 2x2
      iii. Aquesta capa redueix la resolució de la imatge seleccionant el valor màxim en regions de 2x2 píxels.
   (h) **Capa Convolucional Addicional (Conv2D)**
      i. Filtres: 128 (2*num_filters)
      ii. Mida del Nucli: 3x3
      iii. Activació: ReLU
      iv. Padding: Same (per mantenir la mida espacial de la imatge)

(i) **BatchNormalization**: Igual que anteriorment, normalitza les dades de sortida de les funcions d'activació.

(j) **Activació ReLU**: Introducció de no linealitat.

(k) **Capa Convolucional Addicional (Conv2D)**

i. Filtres: 128 (2*num_filters)
ii. Mida del Nucli: 3x3
iii. Activació: ReLU
iv. Padding: Same (per mantenir la mida espacial de la imatge)

(l) **BatchNormalization**: Igual que anteriorment, normalitza les dades de sortida de les funcions d'activació.

(m) **Activació ReLU**: Introducció de no linealitat.

4. **Decoder**

(a) **Primera Capa Convolucional (Conv2D)**

i. Filtres: 128 (2*num_filters)
ii. Mida del Nucli: 3x3
iii. Activació: ReLU
iv. Padding: Same (per mantenir la mida espacial de la imatge)

(b) **BatchNormalization**: Igual que anteriorment, normalitza les dades de sortida de les funcions d'activació.

(c) **Activació ReLU**: Introducció de no linealitat.

(d) **Capa Convolucional Transposada (Conv2DTranspose)**

i. Filtres: 64
ii. Mida del Nucli: 3x3
iii. Activació: ReLU
iv. Padding: Same (per mantenir la mida espacial de la imatge)

(e) **BatchNormalization**: Igual que anteriorment, normalitza les dades de sortida de les funcions d'activació.

(f) **Activació ReLU**: Introducció de no linealitat.

(g) **UpSampling2D**

i. Mida: 2x2
ii. Aquesta capa augmenta la resolució de la imatge duplicant els píxels en cada dimensió. (Veure Figura 10)

(h) **Capa Convolucional Transposada (Conv2DTranspose)**

i. Filtres: 64
ii. Mida del Nucli: 3x3
iii. Activació: ReLU
iv. Padding: Same (per mantenir la mida espacial de la imatge)

(i) **BatchNormalization**: Igual que anteriorment, normalitza les dades de sortida de les funcions d'activació.

(j) **Activació ReLU**: Introducció de no linealitat.

(k) **Capa Final Convolucional (Conv2D)**

i. Filtres: 1

ii. Mida del Nucli: 3x3

iii. Activació: Sigmoid (Veure Taula 2)

iv. Padding: Same (per mantenir la mida espacial de la imatge)

v. Aquesta capa genera la imatge de sortida amb una sola banda de valors entre 0 i 1.

5. **Compilació del Model**

(a) Optimitzador: Adam (Veure apartat 2.2.3)

(b) Funció de Pèrdua: Binary Crossentropy (Veure apartat 2.2.3)

(c) Mètrica: Dice Coefficient. (Veure Figura 10)

A diferència dels models vistos anteriorment en aquest model s'ha realitzat una lleugera modificació dels hiperparàmetres, ja que al contrari del que s'havia fet fins ara, s'han definit 64 filtres a la capa d'entrada del model. Això podria permetre que el model fos capaç de capturar més detalls de les imatges respecte dels models amb 32 filtres. Alhora també és un problema perquè és més lent d'entrenar i hi ha un major risc que tingui sobre ajust.

Es pot veure una representació gràfica del model per facilitar l'enteniment d'aquest si s'observa la Figura 13 i en cas de voler més informació, es pot consultar la Figura

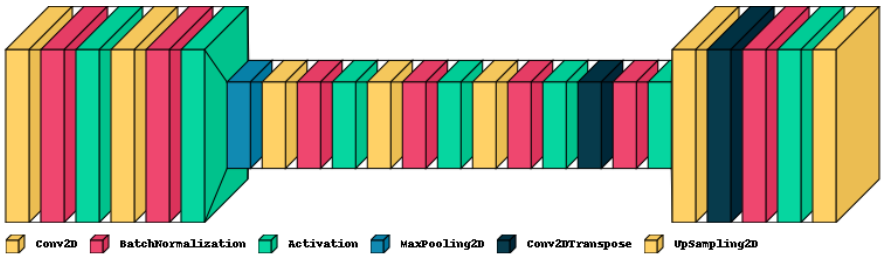

Figura 13: Visualització de les capes del Model 3

3 Resultats

En aquesta secció es presenten els resultats obtinguts arran d'entrenar els diferents models mencionats anteriorment. Principalment, s'avaluaran les dues mètriques mencionades anteriorment, la Funció de Pèrdua i el Dice Coefficient per tal de trobar quin és el millor model per resoldre el problema plantejat. Per tal de justificar quins models es descartés s'utilitzaran les gràfiques on es mostra l'evolució d'aquestes.

A continuació es mostren les mètriques de rendiment dels 3 models:

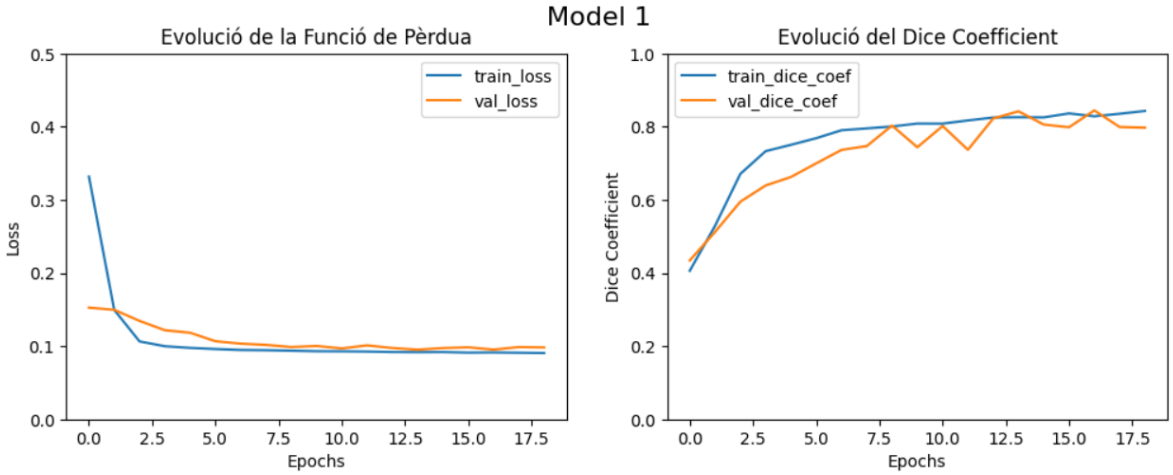

Figura 14: Mètriques del model 1

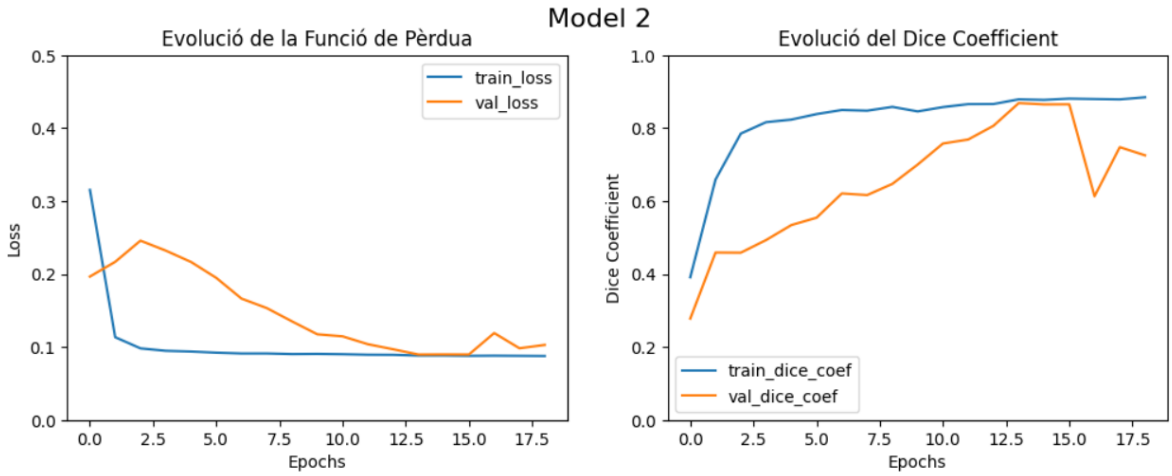

Figura 15: Mètriques del model 2

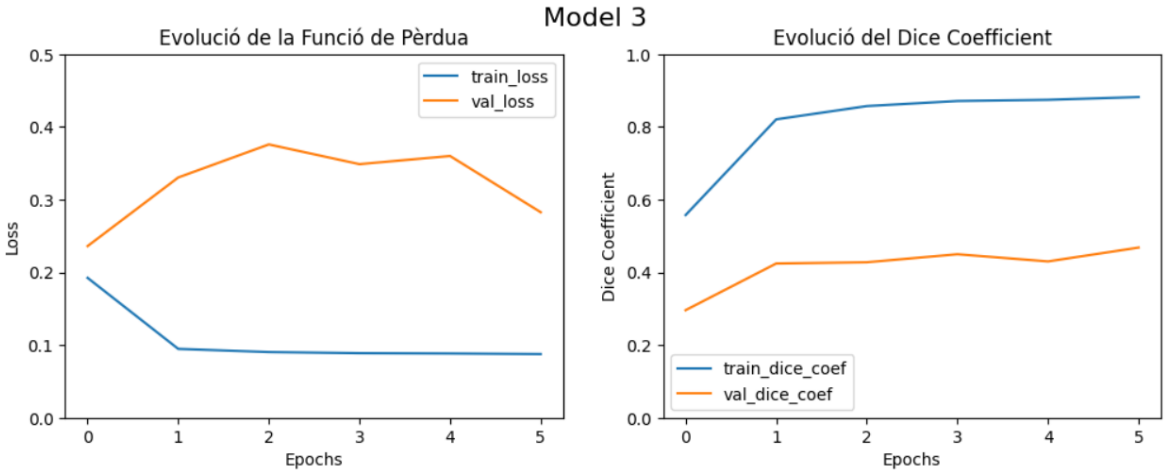

Figura 16: Mètriques del model 3

És important destacar que l'escala de les gràfiques de l'Evolució de la Funció de Pèrdua no és comparable a l'escala de les gràfiques d'evolució del Dice Coefficient. Cal destacar que als gràfics de cada mètrica s'observen els valors obtinguts sobre les dades d'entrenament (train_) i de les dades de validació (val_).

Primerament, es compararan els models segons les seves gràfiques d'evolució de la Funció de Pèrdua. El que es considera una evolució normal seria el que es pot observar a les figures 14 i 15 on tant el valor en dades d'entrenament i de validació els valors van disminuint i, per tant, la distància entre les màscares reals i les predites pel model, va disminuint i, com a resultat, es pot entendre que el model va captant correctament els patrons i relacions entre les dades. En canvi, si s'observa la Figura 16 corresponent al model 3, es pot apreciar que primerament la Loss de les dades d'entrenament sí que disminueix mentre que l'evolució de la Funció en les dades de validació el comportament s'allunya de l'esperat i, per tant, pot dur a interpretar que no s'està entrenant correctament i que en cas de voler prendre'l com a model definitiu caldria investigar que està succeint. Principalment, cal destacar que gràcies a l'Early Stopping el model 3 s'ha aturat a les 5 epochs pel fet que el mateix model ha detectat que no estava evolucionant correctament.

Respecte a l'evolució de la Funció de Pèrdua dels models 1 i 2, s'aprecien lleugeres diferències entre l'evolució d'aquestes. L'evolució de la Loss del model 2 en les dades de validació podria semblar que té un comportament perfectament compatible amb un model amb un rendiment correcte. Cap a les últimes epochs s'observa que la distància entre la Funció en les dades de test i de validació podria semblar que augmenta i, per tant, indicaria que el model està tenint sobre ajustament.

Com a conclusió de l'avaluació de la Funció de Pèrdua, es descarta el model 3 i es manté com a possibles models vàlids els models 2 i 1 tot i que com s'ha esmentat cal veure l'evolució del Dice Coefficient per acabar de seleccionar el model adequat per resoldre el problema.

Passant a l'altra mètrica d'avaluació del model observem el Dice Coefficient. Tal com s'ha comentat respecte de la Funció de Pèrdua, el model 3 es descartarà com a possible model vàlid perquè sembla un cas típic de sobre ajustament de les dades (overfitting) i a part, el coeficient assoleix un valor prop de 0.5. Aquesta conclusió porta a pensar que no sempre el model més complex és el millor, ja que depèn molt de la tipologia de dades i de com sigui de gran el volum d'aquestes.

Observant l'evolució del rendiment dels models 1 i 2 s'observen majors diferències de comportaments a través de les epochs. S'observa que el model 2 en algun punt arriba a un valor del coeficient proper a 0.8 (indica que el 80% dels píxels predits té el color correcte) però cap a les últimes iteracions, s'observa que aquest valor decreix. Aquesta conclusió juntament amb el que s'ha comentat sobre la Funció de Pèrdua d'aquest model, fa sospitar que el model està efectivament començant a realitzar sobre ajustament. Tot i això, si s'aturés l'entrenament abans d'arribar a aquesta situació, es podria considerar com a model bo, però, tot i això, i fent servir el principi de parsimònia o navalla d'Occam (14) en igualtat de condicions i resultats respecte del model 1, es descartaria perquè és un model més complex, amb més temps de computació i assoleix uns resultats equivalents al del model 1 que és més simple.

Pel que fa al model 1 s'observa que l'evolució del Dice Coefficient assoleix ràpidament valors entorn del 0.7 i continua augmentant fins a estabilitzar-se prop del 0.8 i, per tant, es considera un rendiment molt bo i, com a resultat, serà el model escollit per resoldre el problema.

A continuació a la Figura 17, es mostra una mostra de les imatges, les màscares reals i les prediccions del model 1.

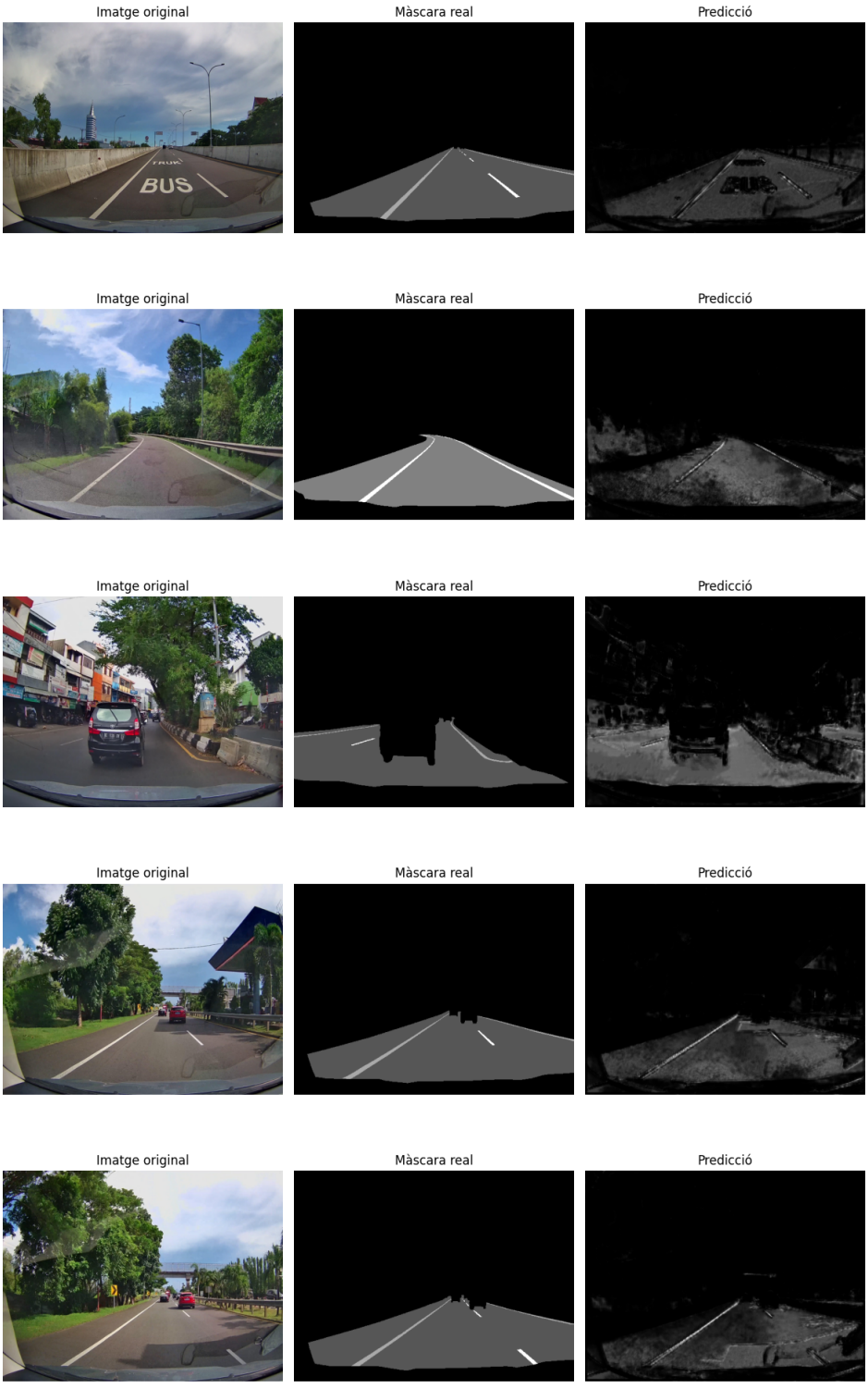

Figura 17: Mostra de prediccions del Model 1

S'observa que realment el model prediu màscares molt semblants a les reals i el més destacable dintre de les prediccions és el fet que prediu molt bé totes les parts de les imatges que són diferents a la carretera i per tant es pot considerar que el model funciona correctament.

Un cop s'han obtingut les prediccions, s'han fet proves sobre com millorar les imatges generades pel model. S'han valorat diverses tècniques de visió per computador i finalment s'ha escollit el procés que s'explica tot seguit.

Cal recordar que s'està tractant amb imatges en blanc i negre i per tant, només tenen un canal de color i per tant tots els valors possibles de cada píxel es situen entre 0 i 255. Per tant, alhora de millorar la predicció i diferenciar quines parts de la imatge pertanyen a la carretera i quines no, s'ha decidit obtenir una mostra d'una imatge de les dades, veure quina màscara prediu el model i realitzar un histograma dels valors dels píxels de la imatge per tal d'observar la seva distribució. (Veure Figura 22).

Figura 18: Imatge original

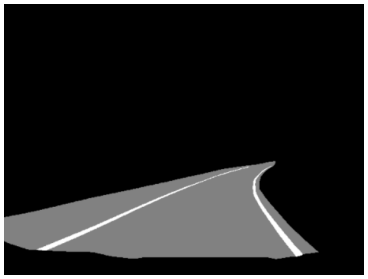

Figura 19: Màscara real

Figura 20: Predicció

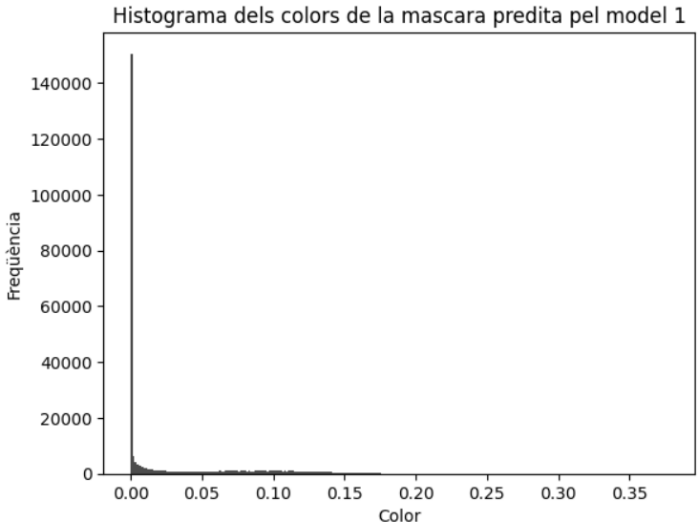

Figura 21: Histograma dels valors de la predicció

Figura 22: Exemple de funcionament del model

Com s'observa a l'histograma de la Figura 22, la majoria dels píxels tenen un valor molt proper a zero (color negre com es pot observar a la predicció). Per tant, s'ha decidit que cal trobar un cert llindar a partir del qual es classifiquin els píxels en color blanc o negre. Primerament, s'han fet proves escollint llindars de manera arbitrària i s'han obtingut resultats prou bons, el problema resideix en el fet que la complexitat d'obtenir una predicció amb la imatge millorada no pot dependre d'un llindar establert arbitràriament.

Com a procés per trobar de manera consistent un llindar prou adequat per les màscares i que aquestes resultin en imatges prou clares s'ha optat per utilitzar un mètode iteratiu on s'ha establert uns possibles valors del llindar (entre 0 i 1 cada 0.01) i alhora per avaluar aquest, s'ha calculat el Dice Coefficient mitjà per cada grup de prediccions un cop s'ha aplicat el llindar corresponent a cada iteració (Veure Figura 23).

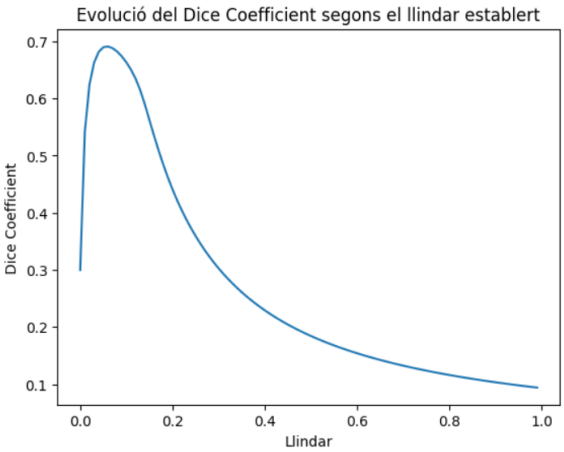

Figura 23: Evolució del Dice Coefficient segonds el llindar establert

S'observa que hi ha un punt on el valor del Dice Coefficient és màxim i, per tant, és on el llindar fa que les màscares es binaritzin (adoptin valors 0 o 1) de la manera òptima per millorar la qualitat de les prediccions.

Anteriorment s'ha mencionat el fet que en la definició de la fòrmula del Dice Coefficient s'ha establert un llindar arbitrari per tal de poder calcular la mètrica. A continuació es mostra el gràfic d'evolució d'aquest on a part de variar el llindar de les imatges també es varia el valor del llindar establert a la funció del Dice Coefficient.

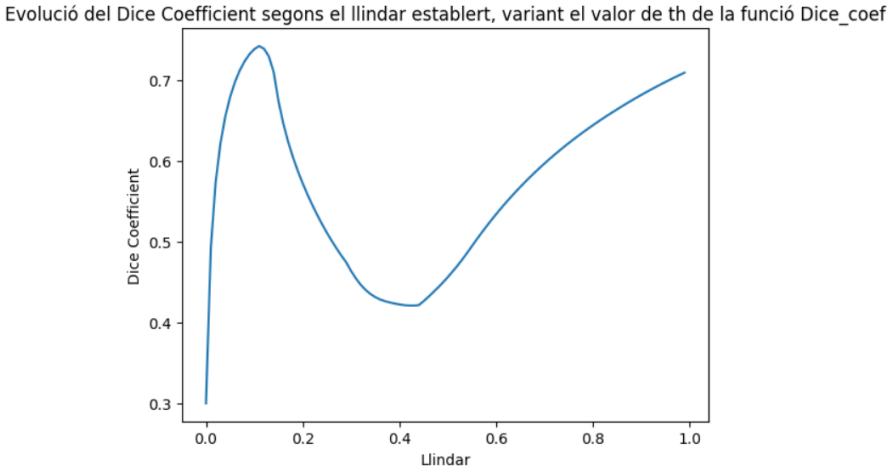

Figura 24: Evolució del Dice Coefficient variant el valor del llindar a la funció de la mètrica

El punt on la corba assoleix el màxim (el qual és el mateix tant en la funció de la Figura 23 i la Figura 24, ja que el valor que es busca és entre 0 i 0.2 aproximadament perquè el màxim que s'assoleix a la Figura 24, no es valora, puix que és inferior al que s'assoleix en l'interval [0,0.2]) és en el valor 0.06 del llindar i, per tant, si l'apliquem observem la següent transformació: (Veure Figura 31)

Figura 25: Imatge real

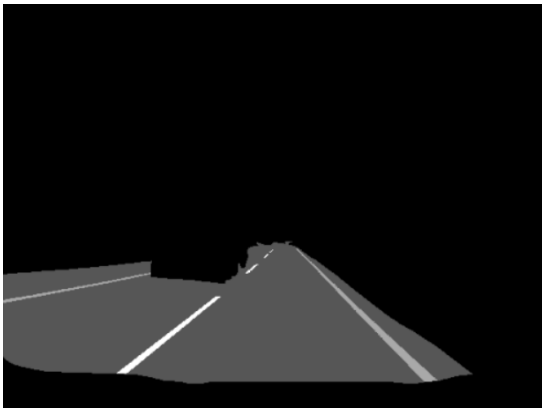

Figura 26: Màscara real

Figura 27: Mostra d'imatge amb la corresponent màscara real

Figura 28: Màscara predita

Figura 29: Màscara predita binaritzada

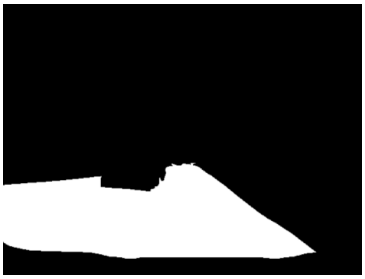

Figura 30: Màscara real binaritzada

Figura 31: Exemple de binarització de les màscares

Com es pot observar a la Figura 31, un cop binaritzada la màscara predita aplicant-hi el llindar de 0.06, s'aconsegueix un resultat prou semblant a la màscara real també binaritzada amb el mateix llindar i com a conseqüència, s'ha aconseguit millorar les prediccions del model aplicant-hi un postprocessat a les màscares. Així i tot, caldria consultar experts en tècniques de visió per computador per acabar de decidir quina tècnica és l'adequada per millorar encara més el resultat.

Finalment, es podria afirmar que s'ha aconseguit un model el qual a partir d'una imatge d'una carretera és capaç de discriminar correctament quina part d'aquesta correspon a la calçada i quina part no.
De cara a treballs futurs seria interessant valorar l'opció d'enfocar la línia d'investigació cap a intentar suavitzar la màscara predita pel model. Sabent que les carreteres són uniformes, i tal com s'observa a la Figura 29, es podria idear un algorisme per "tapar els forats"de la màscara.

3.1 Actualitat

Aquesta tipología de models utilitzats per segmentar imatges, actualment tenen un ús molt extés a l'indústria ja que amb la suficient capacitat computacional i el correcte entrenament i ajustament dels hiperparàmetres, es poden arribar a obtenir resultat realment espectaculars en quant a segmentació d'imatges.

Per exemple actualment s'estan utilitzant l'evolució dels models de segmentació nomenats models fundacionals, els quals permeten identificar, segmentar i nombrar tots els objectes que apareixen en una imatge (o vídeo) independentment de si el model l'ha vist amb anterioritat o no. Un clar exemple d'aquests models són el SEEM (Segment Everything Everywhere All at Once) el qual ha estat desenvolupat per Microsoft (5).

Un altre model amb aquestes és esl SAM (Segment Anything Model) el qual ha estat desenvolupat per Meta (empresa a la qual pertany Facebook) per tal de segmentar qualsevol imatge que se li passi al model. A continuació es mostra un exemple del seu funcionament (Figura 32).

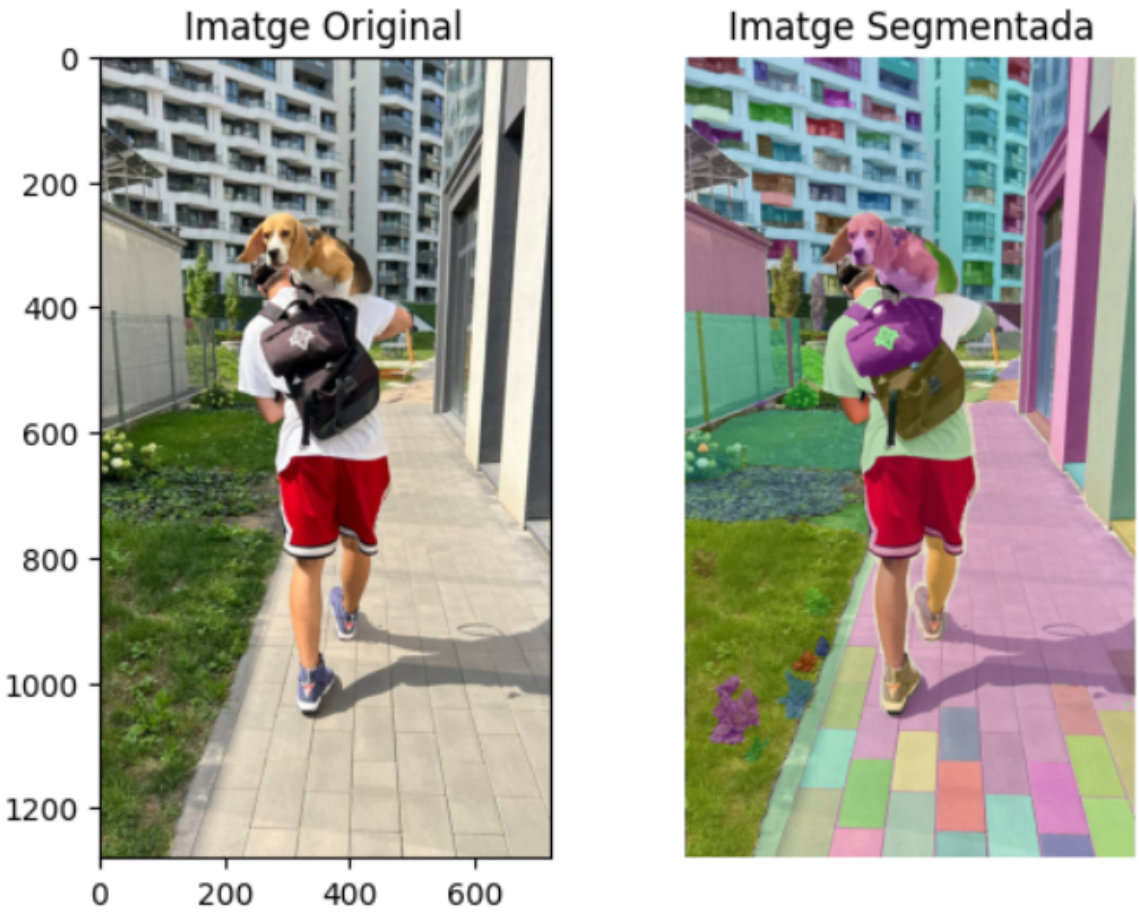

Figura 32: Exemple d'imatge segmentada amb SAM

Com es pot observar el model detecta tots els diferents objectes que hi ha a la imatge de manera molt precisa. Això es deu a que te una complexitat i capacitat computacional molt elevada i alhora un gran volum d'imatges etiquetades correctament.

A continuació es pot apreciar com funciona el model amb una imatge aleatòria del conjunt de dades utilitzat en aquest estudi. (Veure Figura 33).
Si es vol tenir accés al codi complet del treball es pot trobar al següent repositori de Github.

Figura 33: Imatge segmentada amb SAM

4 Bibliografia

1. Falkner, Stefan, Surbhi Goel, and Zachary C. Lipton, 2018. *Reproductiblity in machine learning: A practitioner's guide.* arXiv preprint, arXiv:1802.01445

2. Singh, Amrendra, et al, 2018. *Road Segmentation Using CNN and Distributed Representation* arXiv preprint, arXiv:1809.07688

3. Lyu.,Bai,L., & Huang, X, 2019. *Road Segmentation Using CNN and Distributed LSTM* arXiv.org, link

4. Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, Ross Girshick, 2023. *Segment Anything*,arXiv.org . link

5. Xueyan Zou, Jianwei Yang, Hao Zhang, Feng Li, Linjie Li, Jianfeng Wang, Lijuan Wang, Jianfeng Gao, Yong Jae Lee, 2023. *Segment Everything Everywhere All at Once* arXiv preprint. link

6. Chen C, Qin C, Qiu H, Tarroni G, Duan J, Bai W and Rueckert D, 2020. *Deep Learning for Cardiac Image Segmentation.* Front. Cardiovasc. Med. 7:25. doi: 10.3389/fcvm.2020.00025 link

7. Rosenblatt F. 1958. *The perceptron: a probabilistic model for information storage and organization in the brain* Psychological review,65(6),386-408. link

8. McCulloch, W.S., Pitts, W. 1943. *A logical calculus of the ideas immanent in nervous activity.* Bulletin of Mathematical Biophysics 5,115-133 link

9. Asimov Institute. 2019. *The Neural Network Zoo.* Recuperat el 24 de maig de 2024.link

10. MathWorks. *Dice Coefficient Definition.* Recuperat el 25 de maig de 2024. link

11. Diederik P. Kingma, Jimmy Ba. 2015. *Adam: A Method for Stochastic Optimization.* Conference Paper at the 3rd International Conference for Learning Representations, San Diego, 2015 link

12. Sutskever, I., Martens, J., Dahl, G. & Hinton, G.. 2013. *On the importance of initialization and momentum in deep learning* Proceedings of the 30th International Conference on Machine Learning link

13. Ruder,S. 2017. *An overview of gradient descent optimization algorithms* ArXiv.org link

14. Esteva de Sagrera, J. (2006). *La navaja de Occam* Offarm,25(6),9. Recuperat de link

5 Annex

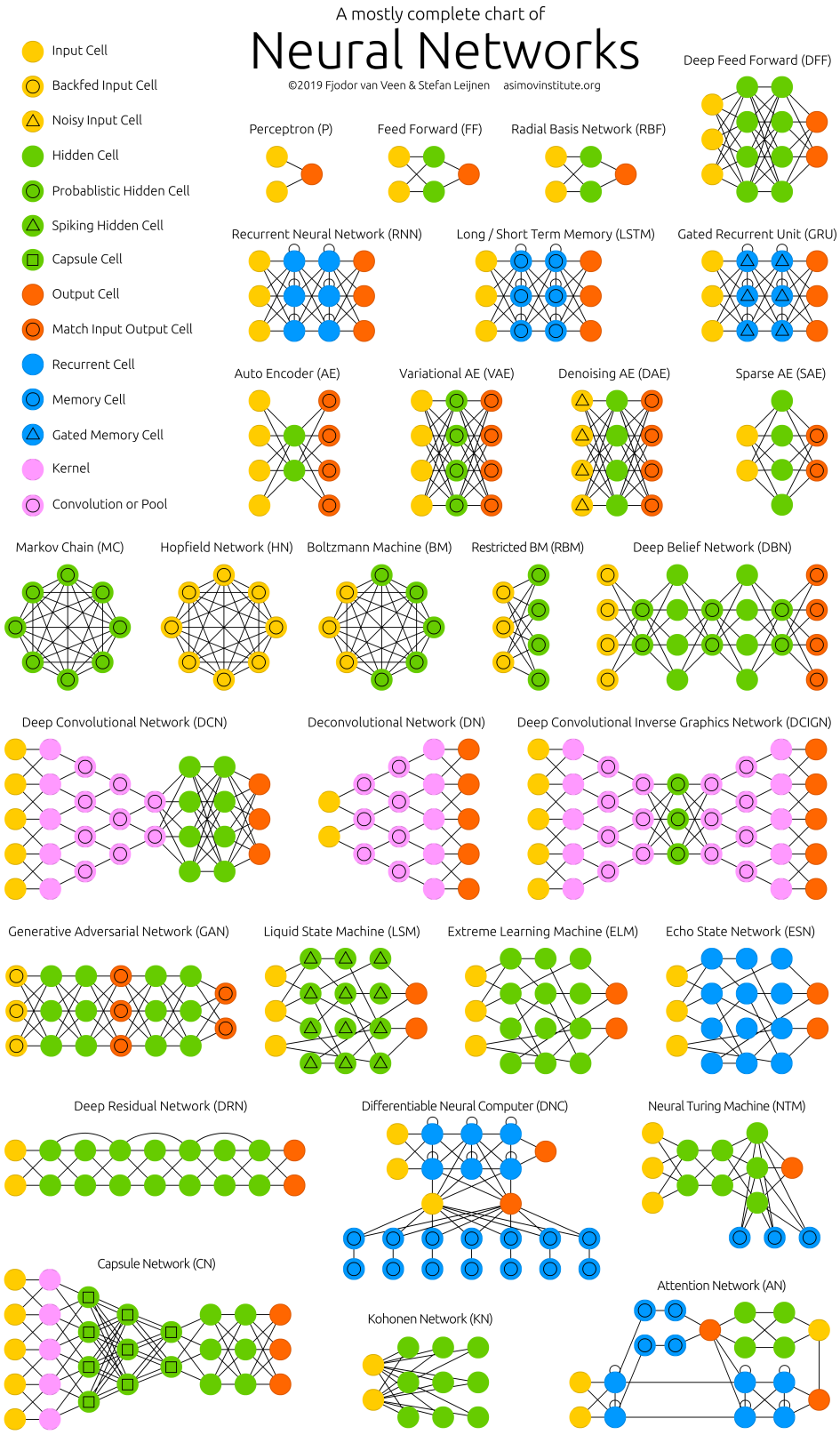

Figura 34: Arquitectures de Xarxes Neuronals

Mètrica	**Descripció**
Accuracy	Proporció de prediccions correctes sobre el total de prediccions. $$\text{Accuracy} = \frac{\text{Nombre de prediccions correctes}}{\text{Nombre total de prediccions}}$$
Precisió	Proporció de verdaders positius sobre el total de prediccions positives. Útil en problemes de classificació binària i multiclasse. $$\text{Precisió} = \frac{\text{Verdaders Positius (VP)}}{\text{Verdaders Positius (VP)} + \text{Falsos Positius (FP)}}$$
Recall (Sensibilitat)	Proporció de verdaders positius sobre el total d'exemples que realment són positius. Mesura la capacitat del model per identificar correctament tots els positius i és clau en problemes com per exemple el diagnòstic de malalties. $$\text{Recall} = \frac{\text{Verdaders Positius (VP)}}{\text{Verdaders Positius (VP)} + \text{Falsos Negatius (FN)}}$$
F1-Score	Mitjana harmònica de la Precisió i Recall. S'utilitza per avaluar models amb conjunts de dades desbalancejades. $$\text{F1-Score} = 2 \cdot \frac{\text{Precisió} \cdot \text{Recall}}{\text{Precisió} + \text{Recall}}$$
AUC-ROC (Àrea sota la corba ROC)	Mesura la capacitat del model per distingir correctament entre classes. Un valor de l'AUC més elevat indica un major rendiment del model.
MSE (Error Quadràtic Mig)	Mitjana dels quadrats dels errors. Utilitzat comunment en problemes de regressió. $$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$ on y_i són els valors reals, $\hat{y}_i$ són els valors predits i n és el nombre total d'observacions.
MAE (Error Absolut Mitjà)	Promig dels errors absoluts. És una mesura de magnitud de l'error més robusta en quant a l'impacte dels outliers. $$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i\rvert$$ on y_i són els valors reals, $\hat{y}_i$ són els valors predits i n és el nombre total d'observacions.

Taula 4: Principals mètriques de validació de Xarxes Neuronals

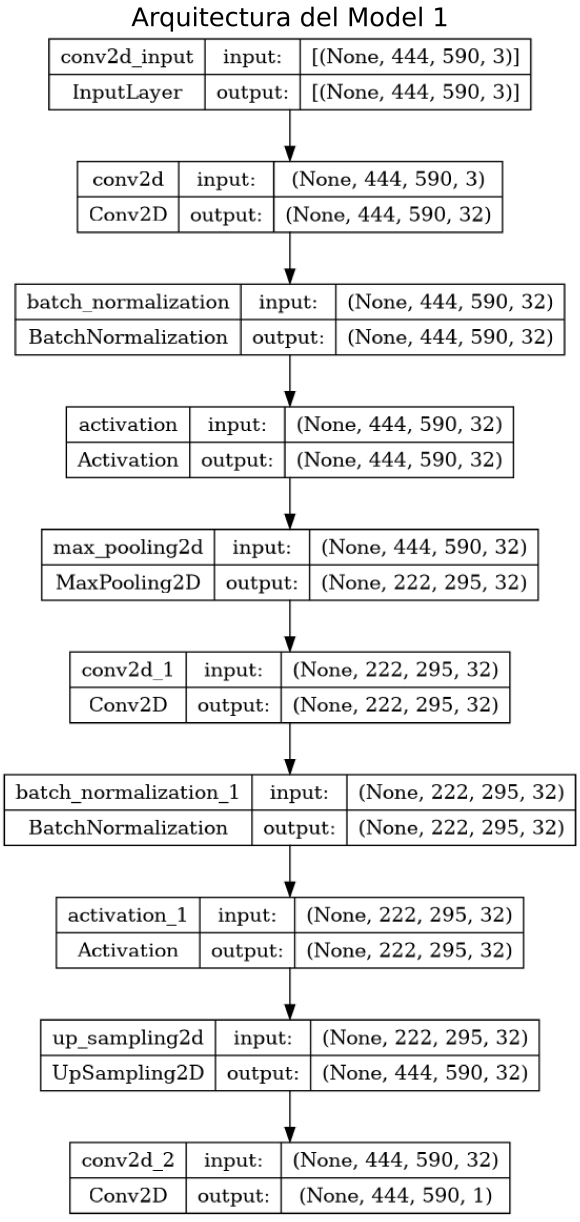

Figura 35: Diagrama de capes i dimensions del model 1

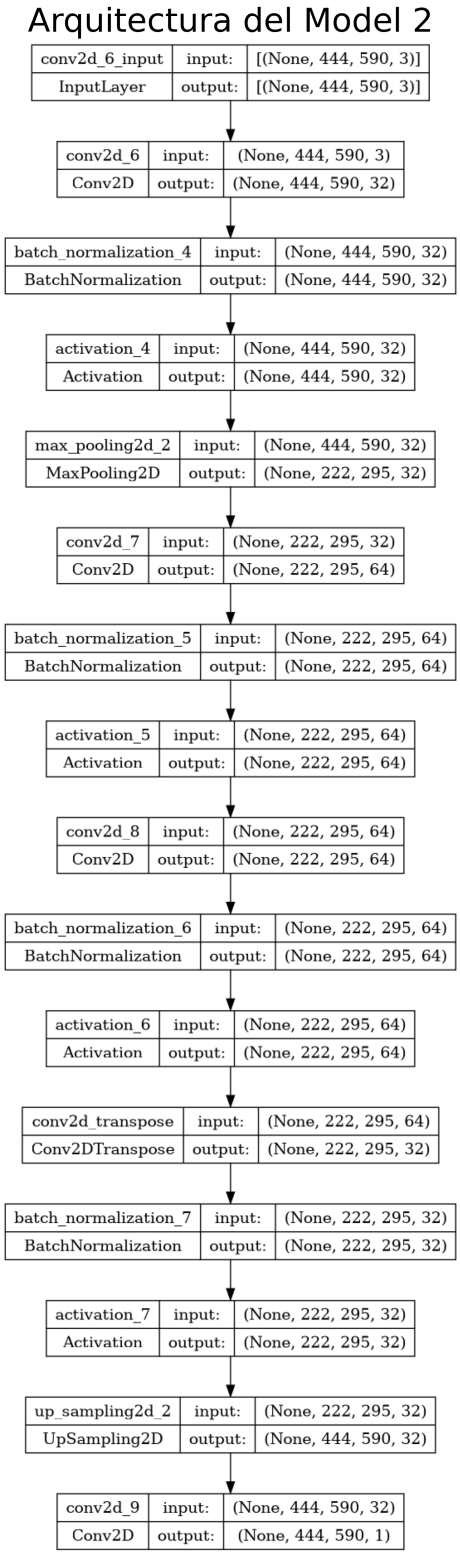

Figura 36: Diagrama de capes i dimensions del model 2

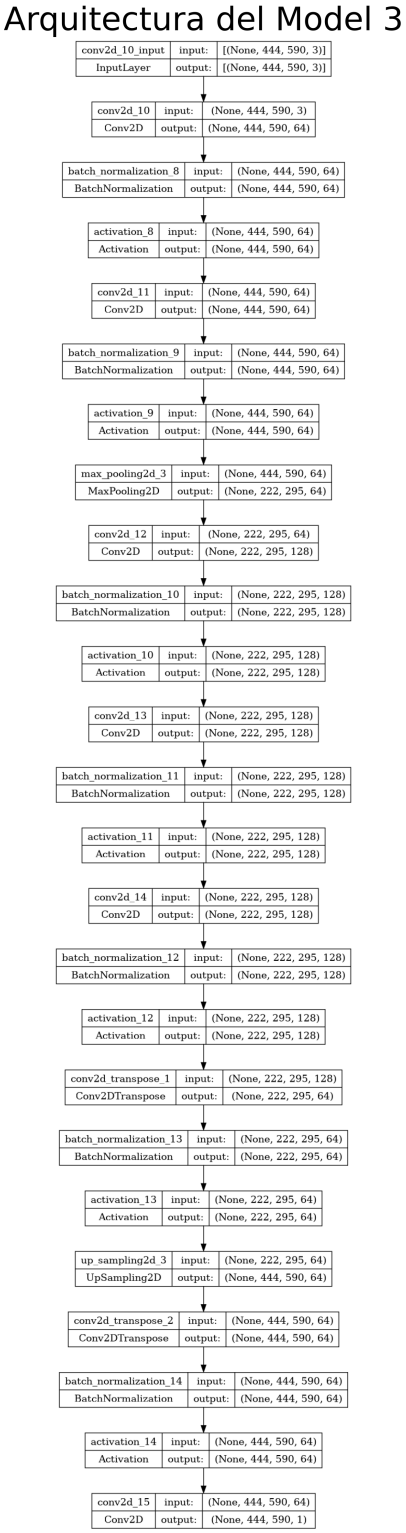

Figura 37: Diagrama de capes i dimensions del model 3