TREBALL DE FI DE GRAU

Grau en Estadística

Títol: Anàlisis predictiu de les senyes arbitrals al món del bàsquet

Autor: Ismael Argemí Fernández

Director: Sergi Ramírez Mitjans i Enric Serrano

Departament: Dept. Estadística e Investigació Operativa

Convocatòria: 2023 - 2024

UNIVERSITAT DE BARCELONA

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

Crèdits/Copyright

Fitxa del treball final

Títol del treball:	Anàlisis predictiu de les senyes arbitrals al món del bàsquet
Nom de l'autor:	Ismael Argemí Fernández
Data d'entrega:	28/06/2024
Titulació o programa:	Grau d'Estadística
Àrea del Treball Final:	Mineria de dades
Idioma del treball:	Català
Paraules clau:	Machine Learning, Deep Learning, Intel·ligència artificial, Xarxes neuronals, Backpropagation, YOLO, senyalització arbitral al bàsquet.

Dedicatòria

Dedicat a la meva família, sobretot als meus pares, per tot el suport incondicional al llarg d'aquests quatre anys.

Agraïments

Vull expressar el meu agraïment més sincer al meu professor i tutor de tesi, Sergi. Gràcies per valorar el meu esforç en les teves matèries i per comptar amb mi des del principi per guiar-me en aquest projecte. Sota el teu tutelatge, he descobert un camp de l'estadística que m'apassiona i que ha enriquit la meva formació acadèmica i professional de manera significativa.

Abstract

0.1 Resum

A l'àmbit de l'analítica esportiva, la detecció precisa de les senyalitzacions dels àrbitres és crucial per a l'anàlisi del joc, l'entrenament i l'emissió. Aquest treball explora la teoria de les xarxes neuronals i aplica el model YOLOv8 per reconèixer i classificar les senyes dels àrbitres de bàsquet.

En crear i anotar un conjunt de dades complet de senyalitzacions arbitrals, el model s'entrena mitjançant diferents combinacions de tècniques de preprocessament i d'augment de dades. L'estudi explica el funcionament de les xarxes neuronals i demostra la seva aplicació pràctica en l'analítica esportiva, destacant la seva utilitat potencial per millorar l'anàlisi mitjançant la intel·ligència artificial.

Paraules clau: *Machine Learning*, *Deep Learning*, Intel·ligència artificial, Xarxes neuronals, *Backpropagation*, YOLO, senyalització arbitral.

0.2 Abstract

In the realm of sports analytics, accurate detection of referee signals is crucial for game analysis, training, and broadcasting. This thesis explores the theory of neural networks and applies a YOLOv8 model to recognize and classify the signals made by basketball referees.

By creating and annotating a comprehensive dataset of referee signals, the model is trained using different combinations of preprocessing and data augmentation techniques. The study explains the workings of neural networks and demonstrates their practical application in sports analytics, highlighting their potential utility in enhancing analysis through artificial intelligence.

Key words: *Machine Learning*, *Deep Learning*, *Artificial Intelligence*, *Neural Networks*, *Backpropagation*, YOLO, *referees signals*.

Índex

Abstract **vi**

0.1 Resum . . . vi

0.2 Abstract . . . vi

Índex **vii**

Llistat de Figures **viii**

Llistat de Taules **x**

1 Introducció **1**

1.1 Descripció general del problema . . . 1

1.2 Objectius del treball . . . 1

1.3 Motivació . . . 1

2 Marc Teòric **2**

2.1 Què és la Intel·ligència Artificial? . . . 2

2.2 Machine Learning i Deep Learning . . . 3

2.3 Xarxes Neuronals . . . 5

2.3.1 La neurona . . . 5

2.3.2 Arquitectura d'una xarxa neuronal . . . 10

2.3.2.1 Backpropagation . . . 10

2.3.2.1.1 Descens del gradient . . . 11

2.3.3 Imatges com a dades tabulars . . . 18

2.3.4 Convolucions i extracció de característiques . . . 18

2.3.5 Pooling i reducció de dimensionalitat . . . 19

2.3.6 Aplicació de tècniques de Machine Learning . . . 19

2.4 YOLOv8 . . . 19

2.4.1 Arquitectura de YOLOv8 20
2.4.1.1 CIoU (Complete Intersection over Union) 22
2.4.1.2 DFL (Distribution Focal Loss) 23
2.4.2 Aplicacions 25

3 Metodologia i aplicació **26**
3.1 Obtenció de la base de dades 26
3.2 Primer escenari 29
3.2.1 Entrenament i validació 29
3.2.2 Conjunt de prova 32
3.3 Segon escenari 36
3.3.1 Entrenament i validació 36
3.3.2 Conjunt de prova 38
3.4 Tercer escenari 41
3.4.1 Entrenament i validació 42
3.4.2 Conjunt de prova 44

4 Conclusions **46**

5 Línies futures **49**

6 Annex **50**

Bibliografia **50**

Índex de figures

2.1 Diagrama de Venn de la IA . . . 2

2.2 Exemple porta AND . . . 5

2.3 Exemple porta OR . . . 6

2.4 Exemple porta XOR . . . 6

2.5 Concatenació de regressions lineals . . . 7

2.6 Esquema típic d'una neurona artificial . . . 7

2.7 Funció esglaonada . . . 7

2.8 Funció sigmoïdal . . . 8

2.9 Funció ReLU . . . 8

2.10 Efecte funcions d'activació . . . 9

2.11 Combinació funcions sigmoïdals per porta XOR . . . 9

2.12 Esquema d'una xarxa neuronal . . . 10

2.13 Funció d'error per a descens del gradient (Himmelblau) . . . 11

2.14 Punt inicial a la funció d'error . . . 12

2.15 Aplicació del descens del gradient . . . 13

2.16 Linia del temps de les versions de YOLO . . . 20

2.17 Arquitectura completa del model YOLOv8 . . . 21

3.1 Exemple d'etiquetatge (Tir lliure sense rebot) . . . 27

3.2 Resum de les mètriques de l'escenari 1 . . . 31

3.3 Matrius de confusió normalitzades de l'entrenament i la validació (Escenari 1) . . . 32

3.4 Matriu de confusió del conjunt de prova . . . 33

3.5 Exemples de prediccions del conjunt de prova original (escenari 1) . . . 34

3.6 Matriu de confusió del conjunt de prova 2 . . . 35

3.7 Exemple de prediccions a partits fora de la mostra original (escenari 1) . . . 35

3.8 Resum de les mètriques de l'escenari 2 37
3.9 Matrius de confusió normalitzades de l'entrenament i la validació (Escenari 2) . 37
3.10 Matriu de confusió del conjunt de prova original (Escenari 2) 38
3.11 Exemple de prediccions de la mostra original (escenari 2) 39
3.12 Matriu de confusió del conjunt de prova 2 40
3.13 Exemple de prediccions a partits fora de la mostra original (escenari 2) 40
3.14 Resum de les mètriques de l'escenari 3 42
3.15 Matrius de confusió normalitzades de l'entrenament i la validació (Escenari 3) . 43
3.16 Matriu de confusió del conjunt de prova original (Escenari 3) 44
3.17 Exemple de prediccions de la mostra original (escenari 3) 44
3.18 Matriu de confusió del conjunt de prova 2 45
3.19 Exemple de prediccions a partits fora de la mostra original (escenari 3) 45

6.1 CM Norm entrenament (escenari 1) 50
6.2 CM Norm validació (escenari 1) 51
6.3 CM Norm entrenament (escenari 2) 52
6.4 CM Norm validació (escenari 2) 53
6.5 CM Norm entrenament (escenari 3) 54
6.6 CM Norm validació (escenari 3) 55

Índex de taules

3.1 Taula de freqüències de les senyes 28

3.2 Taula de freqüències de les senyes després de les modificacions 42

1 Introducció

1.1 Descripció general del problema

En l'àmbit de l'aprenentatge automàtic, les xarxes neuronals han revolucionat la visió per computadora, i YOLOv8 s'ha destacat com una de les arquitectures més avançades en termes de precisió i velocitat, sent utilitzat en una gran varietat de camps com la conducció automàtica o el seguiment dels moviments dels jugadors de pista a la NBA. Tot i aquests avenços, segueix sent un desafiament significatiu l'aplicació d'aquestes tecnologies en contextos especialitzats com l'anàlisi predictiva de senyals arbitrals al bàsquet. Aquest treball investiga el funcionament intern de YOLOv8 i la seva implementació pràctica per millorar la interpretació de gestos arbitrals, crucial per a decisions precises en temps real durant els partits.

1.2 Objectius del treball

La realització d'aquest treball té com a objectius principals entendre el funcionament intern de les xarxes neuronals, des del concepte més bàsic, com és la neurona, fins a l'enrevessat algorisme d'aprenentatge que requereixen (*backpropagation*), passant per l'enteniment de l'arquitectura que conforma el model YOLOv8, per després finalitzar el treball realitzant una aplicació pràctica del model per tal de fer un anàlisi predictiu de les senyes arbitrals en el món del bàsquet.

1.3 Motivació

A nivell personal, la motivació de realitzar aquest treball ve de la curiositat generada al realitzar les assignatures Anàlisi Multivariant i Mètodes Estadístics en Mineria de Dades. Al llarg de la carrera sempre s'ha treballat amb dades tabulars i quan a les últimes classes de Mineria de Dades se'ns va presentar el concepte de les xarxes neuronals i totes les seves aplicacions, em va fer adonar-me que les dades no sempre venen en forma de taula.

Aquesta curiositat generada juntament amb el fet de que el següent pas a la meva formació acadèmica és realitzar el Màster Universitari en Tecnologies Avançades de Telecomunicació de la UPC, concretament el *path* d'aprenentatge profund per al processament multimèdia, és el que ha fet que em decanti per la realització d'aquest treball.

2 Marc Teòric

2.1 Què és la Intel·ligència Artificial?

Respondre a la pregunta de què és la intel·ligència artificial (IA) pot resultar una tasca complexa, ja que no existeix una única definició que expliqui de què es tracta. D'una manera senzilla, la IA es pot definir com qualsevol tècnica que permeti als ordinadors imitar el comportament humà i reproduir o superar la presa de decisions humana per resoldre tasques complexes de manera independent amb la mínima intervenció humana possible.

D'una manera més precisa, es pot formalitzar com la capacitat de les màquines per utilitzar algoritmes i dades, a partir dels quals aprendre i utilitzar la informació obtinguda per realitzar un procés estrictament racional similar al que realitzaria una persona humana.

Algunes de les avantatges que comporta l'ús d'IA en la realització de diverses tasques són:

- Millora de l'eficiència: La utilització de la IA permet automatitzar processos repetitius i tediosos.
- Presa de decisions precisa: Té la capacitat d'analitzar grans volums de dades en temps reals i proporcionar informació útil per la presa de decisions.
- Automatització de processos complexos: Permet l'automatització de processos que requereixen anàlisis avançat.

Dins la IA trobem diferents tipus, dos dels més rellevants són el *Machine Learning*, que consisteix en simples algoritmes de IA que aprenen i s'adapten automàticament a partir de l'experiència, i el *Deep Learning* que és una part del *machine learning* que utilitza xarxes neuronals artificials per tal d'imitar el procés d'aprenentatge del cervell humà.

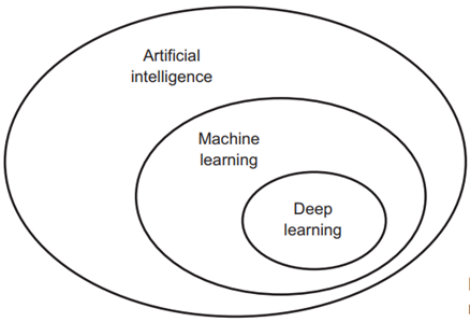


Figura 2.1: Diagrama de Venn de la IA

2.2 Machine Learning i Deep Learning

Per tal d'entendre millor les diferències entre *machine learning* i *deep learning*, en aquest apartat s'explicaran precisament els dos conceptes.

El *machine learning* o aprenentatge automàtic és el conjunt d'algorismes de IA que milloren el seu rendiment amb l'experiència, és a dir, són algorismes d'aprenentatge iteratiu que a partir del conjunt de dades d'entrenament busquen la identificació de patrons que permetin l'elaboració de prediccions cada vegada més perfeccionades.

Aquests algorismes no només s'utilitzen per a la construcció de models clàssics de predicció i classificació, sinó que tenen aplicacions com la realització de tasques cognitives, com podria ser la detecció d'objectes o la traducció de llenguatge natural. En particular, es caracteritzen per funcionar especialment bé en tasques relacionades a grans volums de dades com poden ser models de classificació, regressió i *clustering*.

Dins del ML podem clasificar-lo en tres tipus diferents:

- *Supervised Learning* / Aprenentatge Supervisat:

  Algorisme de ML que requereix una base de dades d'entrenament que inclogui exemples per l'entrada, així com respostes etiquetades o valors numèrics per la resposta. Diferenciant entre problemes de predicció i problemes de classificació.

- *Unsupervised Learning* / Aprenentatge No Supervisat:

  Pren lloc quan el sistema d'aprenentatge ha de detectar patrons sense etiquetes ni especificacions preexistents, és a dir, no hi ha cap variable resposta. De manera que les dades d'entrenament estan formades només per un conjunt de variables explicatives, a partir de les quals es pretén trobar la informació estructural d'interès.

- *Reinforcement Learning*:

  En lloc de proporcionar un conjunt de dades d'entrenament amb variables explicatives i variable/s resposta, es descriu l'estat actual del sistema, especificant un objectiu, proveint una llista de possibles accions i les seves restriccions ambientals per als seus resultats, i es deixa que el model de ML treballi en el procés d'assolir l'objectiu a partir del principi de prova i error.

El *deep learning* o aprenentatge profund és un concepte de machine learning basat en les xarxes neuronals artificials. El funcionament és diferent al del *mahcine learning*, ja que en algorismes de *deep learning* no es passa directament de les variables entrada a la sortida, sinó que hi ha capes entre mig que processen i aprenen de la sortida obtinguda a la capa anterior. Aquest tipus de models són útils en tasques de reconeixement de patrons en formes complexes com imatges, documents, àudios...

Pel que fa al *deep learning* existeixen diverses arquitectures:

- *Convolutional Neural Network (CNN)* / Xarxes neuronals convolucionals:

  Utilitzades en tasques de *computer vision* i *speech recognisition*. La arquitectura està compresa per una serie de capes que permeten l'aprenentatge jeràrquic de característiques segons la tasca de modelatge corresponent.

- *Recurrent Neural Network (RNN)* / Xarxes neuronals recurrents:

  Estan dissenyades explícitament per treballar amb estructures de dades seqüencials. Format a partir de bucles interns que permeten un aprenentatge seqüencial de patrons per modelar dependències temporals. S'utilitzen per prediccions de sèries temporals.

- *Distributed representation*:

  Desenvolupen un paper essencial en el processament del llenguatge natural, on les entitats lingüístiques es projecten en representacions numèriques dins d'un espai semàntic unificat en forma de incrustacions, que codifiquen les paraules en vectors densos de baixa dimensionalitat (*one-hot*). Això permet desenvolupar models lingüístics avançats.

- *Autoencoder*:

  Funcionen d'una manera similar als incrustadors de paraules, proporcionant una representació densa de les dades entrades, però sense limitar-se al llenguatge natural. Consten d'una etapa de codificació i una etapa de descodificació per tractar de reconstruir la entrada original a partir de les característiques apreses. Això permet que la xarxa mantingui només la informació significativa.

- *Generative adversarial neural network (GAN)* / Xarxes neruonals adversarials:

  Pertanyen a la família de models generatrius que tenen com a objectiu aprendre una distribució de probabilitat sobre un conjunt de dades d'entrenament per a que la xarxa pugui generar aleatòriament noves mostres de dades amb certa variació.

  Les GAN consten de dues subxarxes que competeixen entre elles, una xarxa generadora i una xarxa discriminatòria. La GAN treballa fins que la segona subxarxa no aconsegueix discriminar els exemples reals dels generats.

2.3 Xarxes Neuronals

Les xarxes neuronals són un mètode d'intel·ligència artificial que tracta d'ensenyar a les computadores com processar conjunts de dades d'una manera similar al que es produiria durant el procés de sinapsi dins un cervell humà. Per tal d'entendre el funcionament d'una xarxa neuronal, primer cal tenir clar el funcionament d'una neurona artificial.

2.3.1 La neurona

La neurona és la unitat bàsica de processament dins una xarxa neuronal, cadascuna de les quals rep uns valors d'entrada a partir dels quals calcula un valor de sortida, és a dir, una neurona és una funció matemàtica. Internament realitza una suma ponderada, en la que els pesos indiquen amb quina intensitat afecta cada variable o connexió d'entrada a la neurona, és a dir, són els paràmetres de la funció.

El que fa una neurona a nivell intern és un model de regressió lineal, ja que es tenen unes variables d'entrada que defineixen una recta o un pla a la que es pot variar la pendent utilitzant els paràmetres. Com a tot model de regressió, la neurona també consta d'un element independent, que en aquest cas es coneix com a biaix.

Com bé indica el propi nom de les xarxes neuronals, aquestes consisteixen en conjunts de neurones interconnectades entre si. Per tal d'entendre la necessitat de disposar de més d'una neurona, s'expliquen les portes AND, OR i XOR. Per a tots els exemples s'utilitzen conjunts de tan sol dues classes.

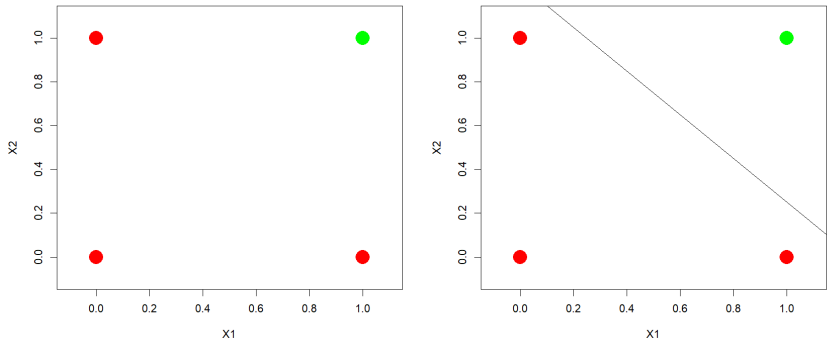


Figura 2.2: Exemple porta AND

El primer exemple es mostra la representació d'una porta AND. Per tant, l'objectiu és trobar els valors dels paràmetres per tal de trobar una frontera entre les dues classes que volem classificar. En aquest cas, la solució es pot trobar utilitzant la recta generada per una sola neurona.

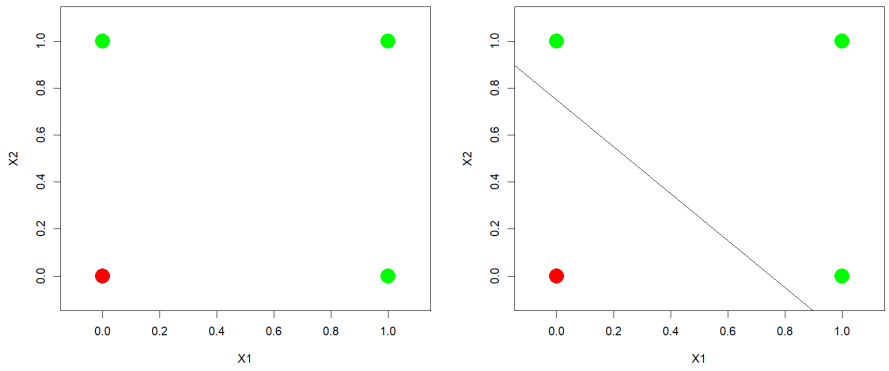


Figura 2.3: Exemple porta OR

El segon exemple es tracta d'una porta OR, on ajustant els valors dels paràmetres es pot arribar, de igual manera que a la porta AND, a trobar una sola recta que separi les dues classes.

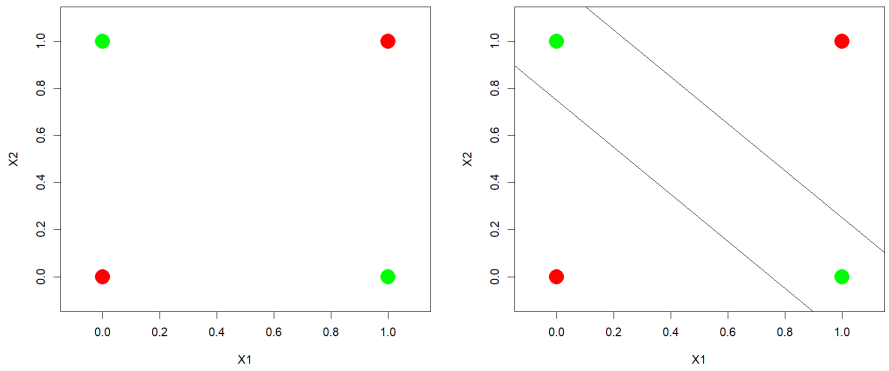


Figura 2.4: Exemple porta XOR

El tercer exemple ens mostra el cas d'una porta XOR, situació on és impossible separar les dues classes amb una sola recta. Aquesta és una de les limitacions d'utilitzar una sola neurona. La solució és molt senzilla, utilitzar dues neurones artificials. Amb dos neurones es tenen dues rectes que si que aconseguirien separar els dos grups.

Aquest exemple il·lustra el fet de que quant més complicada sigui la separació de les classes, major nombre de rectes i, per tant, major nombre de neurones es necessita. De igual manera, quant major sigui el nombre de classes que es vulgui aprendre a classificar, major serà el treball que tingui la xarxa neuronal per aprendre les característiques úniques de cadascuna d'aquestes.

Considerant tot el que s'ha vist fins ara, pot semblar que la representació matemàtica d'una neurona pot tenir la següent forma:

$$Y = WX + b \tag{2.1}$$

Com una xarxa neuronal és una concatenació de neurones, si s'apliquessin les neurones com

s'ha vist fins ara, voldria dir que una xarxa es una concatenació de regressions lineals, però està comprovat que realitzar aquesta concatenació dona com a resultat una nova regressió lineal, és a dir, que seria equivalent a haver utilitzat una sola neurona.

Figura 2.5: Concatenació de regressions lineals

Per tal d'evitar això s'utilitzen les funcions d'activació. Fins ara cada neurona era una suma ponderada que donava com a resultat un valor de sortida. Ara aquest valor de sortida passarà per la funció d'activació, que el que farà serà distorsionar el resultat afegint deformacions no lineals i d'aquesta manera poder concatenar de forma efectiva la computació de conjunts de neurones.

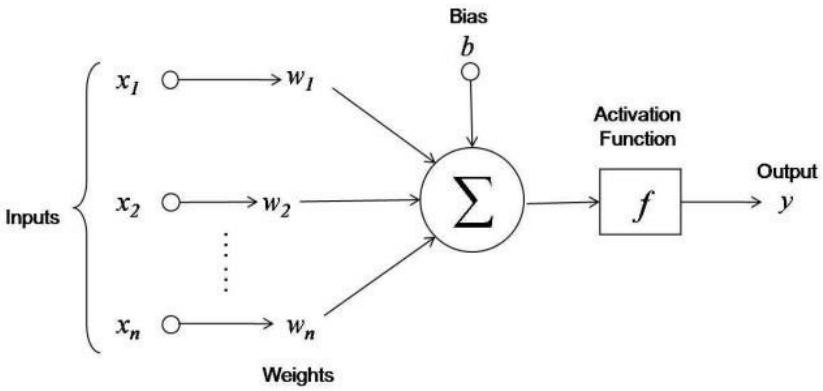


Figura 2.6: Esquema típic d'una neurona artificial

La deformació produïda al valor de sortida de la neurona dependrà de la funció d'activació utilitzada. La més senzilla de totes és la funció esglaonada 2.7, que separa als valors de sortida segons un valor llindar. Tot els que es trobi per sota del llindar prendrà el valor 0 i el que es trobi per sobre prendrà el valor 1.

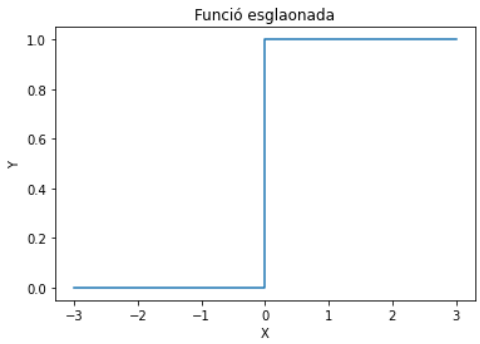


Figura 2.7: Funció esglaonada

També existeixen altres funcions d'activació com pot ser la funció sigmoïdal 2.8, que fa que els valors molt grans es saturin a 1 i els valors molt petits es saturin a 0, per tant, no només s'aconsegueix la deformació que s'està buscant, sinó que també serveix per a representar probabilitats.

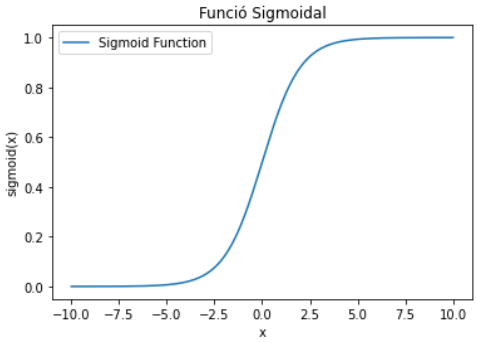


Figura 2.8: Funció sigmoïdal

Una altra de les més utilitzades es la funció ReLU 2.9, que pren valor 0 per a tots els valors d'entrada negatius, mentre que per als valors positius actua com una funció linear.

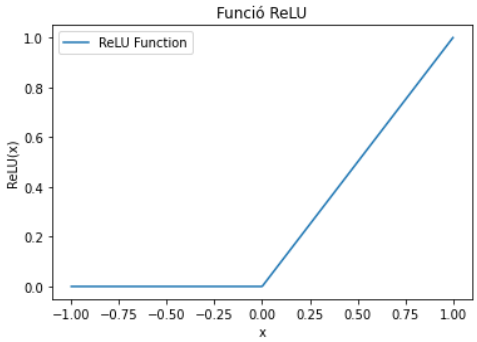


Figura 2.9: Funció ReLU

De fet, durant les explicacions de les portes AND, OR i XOR, ja s'estava fent servir una funció d'activació. Per tal d'apreciar l'efecte de la funció cal afegir el pla generat per la neurona, l'efecte de la funció d'activació és distorsionar aquest pla. La geometria del pla distorsionat que sigui superior al llindar pertanyerà a un grup i el que quedi per sota a un altre.

A continuació es grafiquen unes representacions 2.10 que mostren l'efecte d'una funció d'activació al pla d'una única neurona, capaços de resoldre problemes lineals, però no són suficient en aquelles situacions on separar les classes és una mica més complicat i es requereix de més d'una neurona, és a dir, aquells problemes no lineals on es requereix d'una xarxa neuronal, com podria ser el cas d'una porta XOR 2.4.

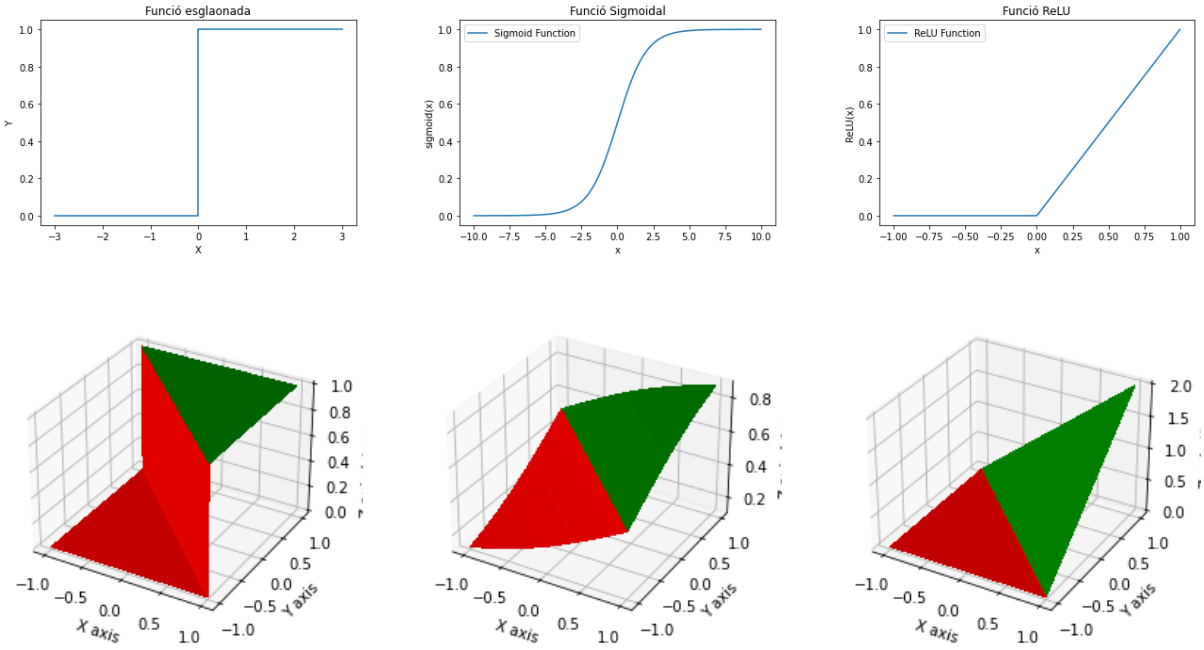


Figura 2.10: Efecte funcions d'activació

Per tal de solucionar la porta XOR es necessita d'un mínim de dues neurones. Es pot solucionar, per exemple, utilitzant com a funció d'activació la funció sigmoïdal, aplicant-la amb pendents oposades per a cada neurona, de manera que el pla generat és:

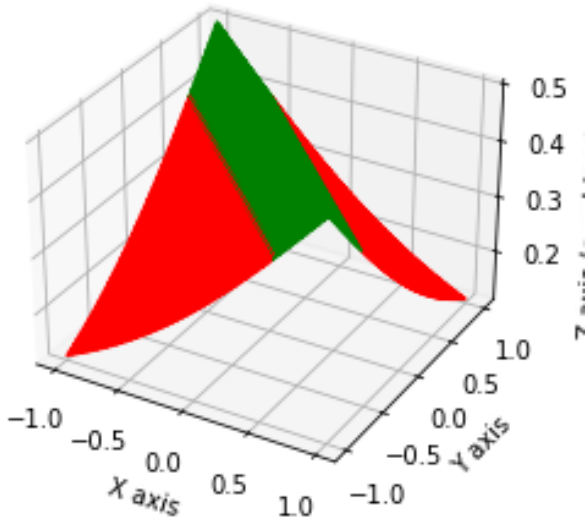


Figura 2.11: Combinació funcions sigmoïdals per porta XOR

Com es pot veure a la imatge 2.11, tot el que queda per sobre de 0.4 és assignat a una classe, mentre que el que es troba per sota s'assigna a l'altre classe.

2.3.2 Arquitectura d'una xarxa neuronal

Una xarxa neuronal està dividida en tres parts principals: La capa d'entrada, la capa de sortida i entre elles es troben una o més capes ocultes.

- La capa d'entrada està formada per totes aquelles neurones que introdueixen la informació a la xarxa. En aquestes no es produeix cap tipus de processament, sinó que actuen com a receptors i propagadors.
- Les capes ocultes són les responsables d'aprendre un *mapping* no linear entre la capa d'entrada i la de sortida. El nombre de capes ocultes depèn de diversos factors com el rati d'aprenentatge o les funcions d'activació utilitzades.
- La capa de sortida és la última de totes i és aquella en la que és produeix el resultat definitiu.

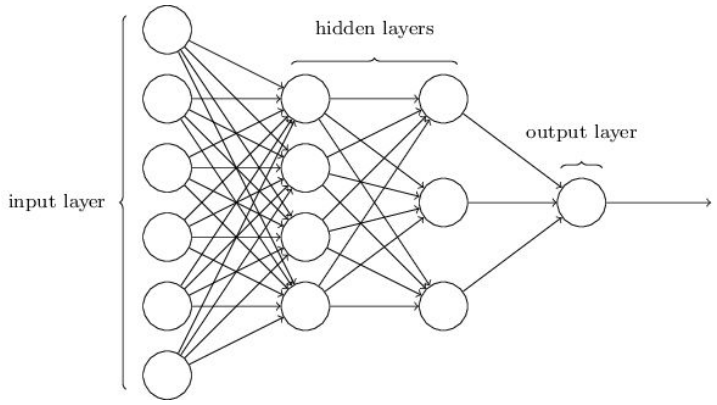


Figura 2.12: Esquema d'una xarxa neuronal

Dins les capes ocultes, cada neurona processa la informació rebuda de la capa anterior i emet un resultat per la següent capa. La funció d'activació de cada neurona regula el pes de cada entrada rebuda, donant major importància a una connexions entrants concretes. Cadascuna de les neurones d'una mateixa capa actuen de forma paral·lela i poden enviar informació només a capes posteriors, en cap cas poden enviar informació a capes anteriors.

2.3.2.1 Backpropagation

Per tal d'ajustar els paràmetres pes i biaix, la xarxa ha de passar per una fase d'entrenament, com qualsevol altre model de ML i DL. La diferència està en que l'algorisme d'aprenentatge utilitzat a les xarxes neuronals és el *backpropagation* [2] .

El *backpropagation* és un algorisme d'aprenentatge automàtic iteratiu que ajusta els pesos de les connexions de la xarxa neuronal per tal de minimitzar una mesura de la diferencia entre els

resultats obtinguts i els resultats desitjats a obtenir, la funció d'error. Com a resultat d'ajustar els pesos, les capes ocultes de la xarxa passen a representar característiques rellevants del domini de la tasca que intenta realitzar-se i les seves regularitats es capten mitjançant la interacció de les neurones.

2.3.2.1.1 Descens del gradient

El que fa l'algorisme de *backpropagation* és buscar el mínim de la funció d'error utilitzant el mètode del descens del gradient. La combinació de pesos que minimitza la funció d'error es considera com la solució del problema d'aprenentatge.

Quan la funció d'error es convexa, per tal de trobar el valors dels pesos que donen lloc al mínim només cal fer la derivada de la funció i igualar-la a zero ($f'(x) = 0$). Per les pròpies propietats de les funcions convexes, es sap que la solució que es trobi serà veritablement el mínim global. Altrament, per a les funcions no-convexes aquesta propietat no es compleix, sinó que pot haver múltiples punts mínims i, per tant, múltiples equacions a resoldre. A més d'altres punts en els que la derivada doni zero, com un màxim local o punts d'inflexió.

Al resoldre un model de regressió lineal es força que la funció d'error sigui l'error quadràtic mitjà perquè així s'obté una funció de forma convexa i, per tant, fàcil de calcular. Però la diversitat de models i funcions d'error al món de l'intel·ligència artificial obliga a trobar una solució (el descens del gradient) per a les funcions no-convexes.

Per tal d'explicar el funcionament del descens del gradient, partirem d'un exemple on la funció d'error és la funció de Himmelblau 2.13:

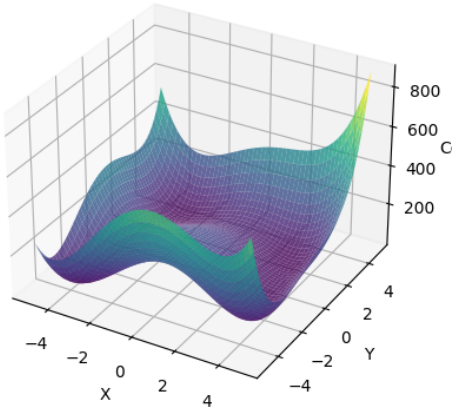


Figura 2.13: Funció d'error per a descens del gradient (Himmelblau)

En aquesta funció tridimensional els eixos X i Y representen els paràmetres i l'eix vertical C representa l'error. S'utilitza només dos paràmetres per visualitzar-ho en un gràfic de tres dimensions, però cal recordar que a ML es sol treballar amb models de milers de paràmetres.

Com al iniciar l'entrenament els valors dels paràmetres prenen un valor aleatori, això equival a dir que ens trobem en un punt aleatori de la superfície, per exemple el punt $(X,Y) = (4,4)$.

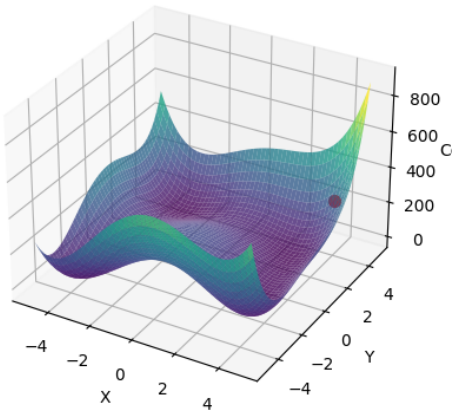


Figura 2.14: Punt inicial a la funció d'error

L'objectiu es descendir sense conèixer la superfície que envolta el punt actual, per tant, el que cal fer és avaluar la pendent en el punt, cosa que com ja s'ha vist equival a calcular la derivada de la funció error al punt, però com la funció error utilitzada és multi-dimensional caldrà calcular les derivades parcials per a cadascun dels paràmetres.

$$\frac{\partial error}{\partial X} \quad \frac{\partial error}{\partial Y} \tag{2.2}$$

Cadascun dels valors de les derivades parcials indicarà la pendent a l'eix del seu respectiu paràmetre. Conjuntament, les derivades parcials 2.2 conformen el vector gradient, que indica la direcció cap a la que la pendent ascendeix el màxim.

$$\left[\frac{\partial error}{\partial X}, \frac{\partial error}{\partial Y}\right] = \nabla f \tag{2.3}$$

Com l'objectiu es descendir i no ascendir, el que es fa es prendre la direcció oposada, és a dir, al actualitzar els paràmetres es restarà el vector gradient 2.3.

$$\theta := \theta - \nabla f \tag{2.4}$$

Això donaria lloc a un nou conjunt de paràmetres i, per tant, modificaria la posició actual, procés que es repetirà fins arribar a una zona on seguir iterant ja no suposi una variació significativa del cost (error), és a dir, un punt on la pendent convergeixi a 0, un mínim local.

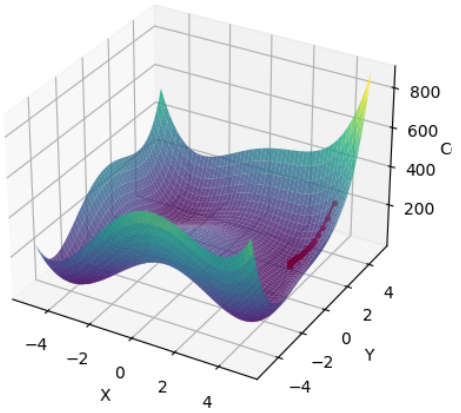


Figura 2.15: Aplicació del descens del gradient

De manera que l'algorisme queda de la següent manera:

$$\text{REPETIR FINS A LA CONVERGÈNCIA} \qquad \theta := \theta - \nabla f \tag{2.5}$$

No obstant, a l'algorisme li falta un últim paràmetre conegut com a *learning rate* o rati d'aprenentatge, el qual es sol representar amb una α i defineix quant afecta el vector gradient a l'actualització dels paràmetres a cada iteració.

$$\text{REPETIR FINS A LA CONVERGÈNCIA} \qquad \theta := \theta - \alpha\nabla f \tag{2.6}$$

En funció del valor que prengui el *learning rate*, $\alpha \in [0,1]$, l'algorisme avançarà més o menys a cada iteració. Si pren un valor petit, l'algorisme haurà de realitzar un gran nombre d'iteracions per apropar-se al cost mínim. I si pren un valor massa elevat, l'actualització serà massa gran, causant que els paràmetres no arribin a convergir al mínim.

Al treballar amb xarxes neuronals, el gradient té el mateix significat que a la regressió lineal, com varia el cost quan varia un paràmetre, però la forma en la que variar un paràmetre pot afectar al cost de la xarxa neuronal és complexe, ja que pot afectar a un gran nombre de connexions. A més, l'efecte del paràmetre es veu controlat pels paràmetres de les capes anteriors, és a dir, existeix una cadena de responsabilitats que dificulta el càlcul de les derivades parcials.

És gràcies a l'algorisme de *backpropagation* que es poden obtenir les derivades parcials dels paràmetres de la xarxa respecte a l'error. De igual manera que es feia abans es farà ús del descens del gradient per optimitzar la funció d'error, fent ús de la tècnica de *backpropagation* per a calcular el vector de gradients dins la complicada arquitectura de la xarxa neuronal.

El que es fa al aplicar la tècnica de *backpropagation* és realitzar un anàlisi de la cadena de responsabilitats, per responsabilitzar a aquelles neurones culpables de fortes senyals d'error.

Aquest anàlisi de la responsabilitat de cada neurona només té sentit si es realitza cap enrere, des de la senyal d'error fins a les primeres capes. Això es degut a que l'error de les capes anteriors depèn de l'error de les capes posteriors.

D'aquesta manera, aplicant la retropropagació de l'error es pot determinar la part de culpa que té cada neurona al resultat final. Un cop imputats els errors a les neurones de l'última capa es repeteix el mateix procés com si aquest fos l'error final de la xarxa. És a dir, aplicar *backpropagation* és operar sempre de forma recursiva, movent l'error cap enrere, de forma que al arribar a la primera capa s'haurà obtingut quin és l'error de cada neurona i dels seus paràmetres, només propagant l'error una sola vegada. Serà a partir de l'obtenció dels errors que es calcularan les derivades parcials de cada paràmetre, conformant així el vector gradient, que és justament el que necessita l'algorisme del descens del gradient per minimitzar l'error.

Cal tenir en compte que dins la xarxa existeixen dos tipus de paràmetres diferents que són els pesos w i el terme de biaix b. Això vol dir que en veritat s'haurà de calcular dos tipus diferents de derivades parcials, una respecte al paràmetre w i una altra respecte al paràmetre b.

$$\frac{\partial C}{\partial w} \qquad \frac{\partial C}{\partial b} \tag{2.7}$$

En aplicar la tècnica de backpropagation es treballa cap enrere, per tant, es comença per calcular les derivades dels paràmetres de l'última capa (L).

$$\frac{\partial C}{\partial w^L} \qquad \frac{\partial C}{\partial b^L} \tag{2.8}$$

Per calcular les derivades és important analitzar quin es el camí que connecta el valor del paràmetre amb el cost final. A l'última capa el camí no es gaire llarg, tot i que té diversos passos. Recordant el funcionament de la neurona, el paràmetre w participa a una suma ponderada.

$$Z^L = W^L X + b^L \qquad \text{On X representa els resultats de la capa anterior} \tag{2.9}$$

El resultat de la suma ponderada es passa per la funció d'activació $a()$ i el resultat d'aquesta serà avaluat per la funció de cost $C()$ per determinar així l'error de la xarxa, $C(a(Z^L))$.

El fet de passar el resultat d'una funció per una funció i aquest mateix resultat passar-lo per una altra, és conegut com a composició de funcions i per a calcular la derivada d'una composició de funcions cal utilitzar una eina de càlcul matemàtic anomenada *chain rule* o regla de la cadena. La *chain rule* expressa el fet que per a calcular la derivada d'una composició de funcions només cal multiplicar les derivades intermèdies.

$$Z^L = W^L a^{L-1} + b^L \qquad\qquad C(a^L(Z^L)) \tag{2.10}$$

$$\frac{\partial C}{w^L} = \frac{\partial C}{\partial a^L}\frac{\partial a^L}{\partial z^L}\frac{\partial z^L}{\partial w^L} \tag{2.11}$$

$$\frac{\partial C}{b^L} = \frac{\partial C}{\partial a^L}\frac{\partial a^L}{\partial z^L}\frac{\partial z^L}{\partial b^L} \tag{2.12}$$

- Derivada de l'activació respecte al cost: Com varia el cost de la xarxa quan varia l'*output* de l'activació de les neurones de l'última capa.

  Per tant, si la funció de cost fos l'error quadràtic mig,

$$C(a_i^L) = \frac{1}{2}\sum_j (y_j - a_j^L)^2 \quad \Longrightarrow \quad \frac{\partial C}{\partial a_j^L} = (a_j^L - y_j) \tag{2.13}$$

- Derivada de l'activació respecta a Z (suma ponderada): Com varia l'*output* de la neurona quan variem la suma ponderada d'aquesta.

  Si per exemple la funció d'activació és la funció sigmoïdal,

$$a^L(z^L) = \frac{1}{1+e^{-z^L}} \quad \Longrightarrow \quad \frac{\partial a^L}{\partial z^L} = a^L(z^L)(1 - a^L(z^L)) \tag{2.14}$$

- Derivades parcials dels paràmetres: Com varia la suma ponderada amb respecte a una variació dels paràmetres (w i b).

$$z^L = \sum_i a_i^{L-1} w_i^L + b^L \tag{2.15}$$

  La derivada de la suma ponderada respecte al terme de biaix és:

$$\frac{\partial z^L}{\partial b^L} = 1 \tag{2.16}$$

  I la derivada de la suma ponderada respecte al pes, on la derivada és el valor d'entrada a la neurona que connecta la connexió per a la qual el paràmetre fa referencia:

$$\frac{\partial z^L}{\partial w^L} = a_i^{L-1} \tag{2.17}$$

De manera que la solució per als paràmetres de l'última capa es calcula computant la següent formula:

$$\frac{\partial C}{w^L} = \frac{\partial C}{\partial a^L}\frac{\partial a^L}{\partial z^L}\frac{\partial z^L}{\partial w^L} \qquad \frac{\partial C}{b^L} = \frac{\partial C}{\partial a^L}\frac{\partial a^L}{\partial z^L}\frac{\partial z^L}{\partial b^L} \tag{2.18}$$

$$\frac{\partial C}{\partial a_j^L} = (a_j^L - y_j) \qquad \frac{\partial a^L}{\partial z^L} = a^L(z^L)(1-a^L(z^L)) \qquad \frac{\partial z^L}{\partial b^L} = 1 \qquad \frac{\partial z^L}{\partial w^L} = a_i^{L-1} \tag{2.19}$$

Utilitzant la intuïció es pot veure que a les formules de les derivades parcials dels paràmetres, el bloc de les dues primeres derivades parcials multiplicades representa en quin grau es modifica el cost quan es produeix un petit canvi a la suma ponderada de la neurona. Si la derivada pren un valor gran, vol dir que un petit canvi a la suma de la neurona es veurà reflectit al resultat final, mentre que si el valor de la derivada pren un valor petit, el canvi de la suma no es veurà reflectit al resultat final.

$$\frac{\partial C}{\partial a^L}\frac{\partial a^L}{\partial z^L} = \frac{\partial C}{\partial z^L} \tag{2.20}$$

És a dir, aquesta derivada explicarà la responsabilitat que tingui la neurona sobre el resultat final i, per tant, a l'error. Si la neurona es responsable de l'error final, s'haurà d'utilitzar aquesta informació per culpabilitzar-la. És per això que aquesta derivada 2.20 es sol conèixer com l'error imputat a la neurona i es sol representar com a δ^L.

$$\delta^L = \frac{\partial C}{\partial z^L} \tag{2.21}$$

Per tal de simplificar es pot reestructurar l'expressió inicial en funció de l'error de les neurones de la capa L.

$$\frac{\partial C}{w^L} = \frac{\partial C}{\partial a^L}\frac{\partial a^L}{\partial z^L}\frac{\partial z^L}{\partial w^L} = \delta^L\frac{\partial z^L}{\partial w^L} = \delta^L a_i^{L-1} \qquad \frac{\partial C}{b^L} = \frac{\partial C}{\partial a^L}\frac{\partial a^L}{\partial z^L}\frac{\partial z^L}{\partial b^L} = \delta^L\frac{\partial z^L}{\partial b^L} = \delta^L 1 = \delta^L \tag{2.22}$$

Per a calcular les derivades de les $L-1$ capes restants només cal afegir una expressió més.

Aplicant la *chain rule* a la capa $L-1$ es forma la següent composició:

$$C(a^L(W^L a^{L-1}(W^{L-1}a^{L-2}+b^{L-1})+b^L)) \tag{2.23}$$

De manera que les expressions s'allarguen a les següents:

$$\frac{\partial C}{\partial w^{L-1}} = \frac{\partial C}{\partial a^L}\frac{\partial a^L}{\partial z^L}\frac{\partial z^L}{\partial a^{L-1}}\frac{\partial a^{L-1}}{\partial z^{L-1}}\frac{\partial z^{L-1}}{\partial w^{L-1}} \tag{2.24}$$

$$\frac{\partial C}{\partial b^{L-1}} = \frac{\partial C}{\partial a^L}\frac{\partial a^L}{\partial z^L}\frac{\partial z^L}{\partial a^{L-1}}\frac{\partial a^{L-1}}{\partial z^{L-1}}\frac{\partial z^{L-1}}{\partial b^{L-1}} \tag{2.25}$$

Al estar anant cap enrere, hi ha expressions que ja estan calculades:

- Error de la capa L

$$\delta^L = \frac{\partial C}{\partial a^L}\frac{\partial a^L}{\partial z^L} \tag{2.26}$$

- Derivades dels paràmetres

  Aquestes es desenvolupen d'igual manera que a la capa L, de forma que queden:

$$\frac{\partial z^{L-1}}{\partial w^{L-1}} = a^{L-2} \qquad \frac{\partial z^{L-1}}{\partial b^{L-1}} = 1 \tag{2.27}$$

- Derivada de la funció d'activació

$$\frac{\partial a^{L-1}}{\partial z^{L-1}} \tag{2.28}$$

De manera que l'única derivada que cal calcular és:

$$\frac{\partial z^L}{\partial a^{L-1}} \tag{2.29}$$

Que expressa com varia la suma ponderada d'una capa quan es varia l'output d'una neurona a la capa prèvia. Aquesta derivada és fàcil de calcular i és bàsicament la matriu de paràmetres W^L que connecta ambdues capes.

$$\frac{\partial z^L}{\partial a^{L-1}} = W^L \tag{2.30}$$

De forma que el que s'està fent és retropropagar l'error a la capa anterior, distribuint l'error en funció de quines són les ponderacions de les connexions.

A partir de tot això, es té una nova expressió a partir de la que es pot obtenir les derivades parcials.

$$\frac{\partial C}{\partial a^L}\frac{\partial a^L}{\partial z^L}\frac{\partial z^L}{\partial a^{L-1}}\frac{\partial a^{L-1}}{\partial z^{L-1}} = \delta^L W^L \frac{\partial a^{L-1}}{\partial z^{L-1}} = \frac{\partial C}{\partial z^{L-1}} = \delta^{L-1} \tag{2.31}$$

És aquí on es pot valorar l'eficàcia de la tècnica de *backpropagation*, ja que el que s'ha fet a aquesta capa és extensible a la resta de capes de la xarxa.

1. Computació de l'error de l'ultima capa (L)

$$\delta^L = \frac{\partial C}{\partial a^L}\frac{\partial a^L}{\partial z^L} \tag{2.32}$$

2. Retropropagació de l'error a la capa anterior

$$\delta^{l-1} = W^l \delta^l \frac{\partial a^{l-1}}{\partial z^{l-1}} \tag{2.33}$$

3. Càlcul de les derivades de la capa utilitzant l'error

$$\frac{\partial C}{\partial b^{l-1}} = \delta^{l-1} \qquad \frac{\partial C}{\partial w^{l-1}} = \delta^{l-1} a^{l-2} \tag{2.34}$$

I així successivament, recorrent totes les capes de la xarxa fins al final. D'aquesta manera amb un únic pas s'haurien calculat tots els errors i les derivades parcials de la xarxa fent servir només les quatre expressions enumerades.

2.3.3 Imatges com a dades tabulars

Les imatges digitals, tot i que visualment complexes, es representen en la seva forma fonamental com a matrius de números. Aquestes matrius poden ser vistes com a dades tabulars on:

- Píxels com a entrades: Cada píxel a una imatge es correspon amb una entrada a la taula, amb valors que representen la intensitat de colors (en escala de grisos) o components de color (en imatges RGB, on cada píxel té associats valors per als canals vermell, verd i blau).

- Dimensionalitat : Una imatge de $M * N$ píxels amb C canals de color pot ser representada com una taula amb $M * N * C$ entrades. Això significa que una imatge RGB de 28*28 píxels es pot veure com una taula de 784 files i 3 columnes.

2.3.4 Convolucions i extracció de característiques

Les xarxes neuronals convolucionas (CNN), com ja s'ha dit estan especialment disenyades per a operar sobre imatges de forma eficient:

- Filtres convolucionals: Les CNN apliquen petits filtres (kernels) que recorren la imatge (taula de dades) i realitzen operacions matemàtiques (convolucions). Aquestes operacions permeten a la xarxa extraure característiques locals de la imatge, com vores, textures i patrons.
- Mapes de característiques: El resultat de les convolucions són una serie de mapes de característiques que són, de fet, noves taules de dades amb característiques més abstractes i rellevants que els valors originals del píxels.

2.3.5 Pooling i reducció de dimensionalitat

Les operacions de *pooling* (com max-pooling i average-pooling)també treballen sobre aquestes taules de dades:

- Submostreig: El *pooling* redueix la dimensionalitat dels mapes de característiques, conservant la intformació més important i reduint la carrega computacional per a les posteriors capes de la xarxa.

2.3.6 Aplicació de tècniques de Machine Learning

Una vegada la CNN ha transformat les imatges (taules de dades originals) en mapes de característiques:

- Capes completament connectades: Les característiques extretes "s'aplanen"(*flatten*) i es passen a capes completament connectades, que funcionen de manera similiar a les capes d'entrada dels models tradicionals de *machine learning* que operen sobre dades tabulars.
- Classificació i regressió: Aquestes capes finals realitzen tasques de classificació o regressió utilitzant les característiques apreses, aplicant tècniques similars a les utilitzades en *machine learning* sobre dades tabulars.

2.4 YOLOv8

Per al desenvolupament dels models del treball s'utilitza una de les últimes versions de la família de models per a detecció d'objectes *You Only Look Once* (YOLO), coneguda per la seva precisió i velocitat. Desenvolupat per l'equip d'Ultralytics, YOLOv8 es construeix a partir de l'èxit dels seus predecessors incloent diverses innovacions clau que impulsen els límits de la detecció d'objectes en temps real.

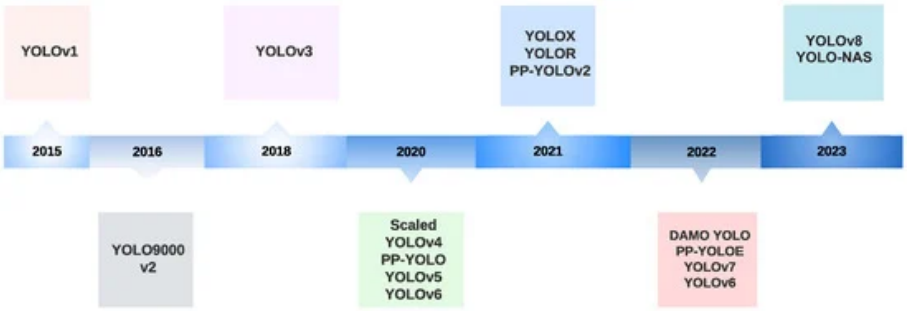


Figura 2.16: Linia del temps de les versions de YOLO

La detecció d'objectes involucra la identificació i localització d'objectes d'interès dins una imatge o vídeo. Els mètodes tradicionals depenien de les *sliding windows*, que són un tipus de mecanisme d'atenció utilitzat en xarxes neuronals que permet que el model es centri en diferents parts de la seqüència d'entrada al realitzar les prediccions, proporcionant un enfoc més flexible i conscient del contingut. D'altra banda, YOLO va revolucionar el camp al començar a tractar la detecció d'objectes com un simple problema de regressió, fent prediccions de *bounding boxes* o quadres delimitadors i probabilitats de classe directament sobre els objectes de la imatge en un sol pas, fent-lo significativament més ràpid.

2.4.1 Arquitectura de YOLOv8

L'arquitectura del model YOLOv8 es pot dividir en tres parts principals:

- *Backbone* / Xarxa troncal: Dins el *backbone* es troba la xarxa neuronal convolucional (CNN), responsable d'extraure les característiques de les imatges d'entrada. YOLOv8 utilitza el *backbone* adaptat CSPDarknet53, que en comptes d'utilitzar connexions parcials entre etapes (*Cross-Stage Partial* o CSP) com feien les anteriors versions de YOLO, ara utilitza mòduls C2f. Aquests mòduls actuen com a colls d'ampolla (*bottleneck*) parcials amb dues convolucions internes que controlen el flux d'informació.

- *Neck* / Coll: El coll o també conegut com a extractor de característiques, fusiona mapes de característiques de diferents etapes del *backbone* per capturar la informació de diverses escales. Comparat amb altres models, YOLOv8 utilitza el mòdul C2f en comptes dels tradicionals *Feature Pyramid Network* (FPN). Aquest mòdul incorpora dues branques de flux del gradient paral·leles, facilitant un flux de la informació del gradient més robust.

- *Head* / Cap: Aquesta part és la encarregada de realitzar les prediccions. YOLOv8 utilitza múltiples mòduls de detecció que prediuen quadres delimitadors, puntuadors d'objectivitat i probabilitats de classe per a cada casella de la quadrícula dins el mapa de característiques. Més tard, les prediccions s'agreguen per a obtenir les decisions finals.

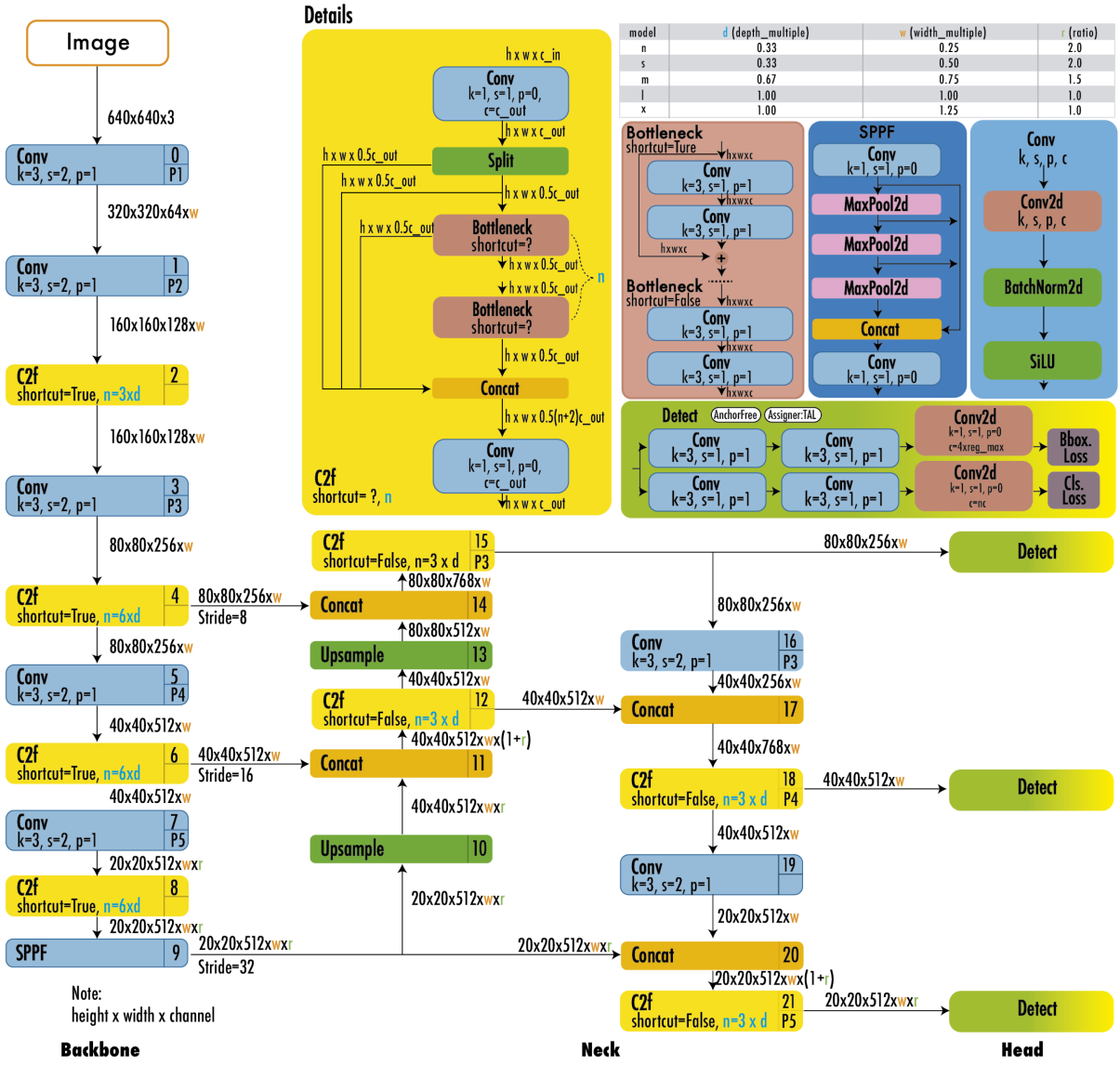


Figura 2.17: Arquitectura completa del model YOLOv8

Com a tota xarxa neuronal, es necessita una funció de cost o error que permeti valorar la precisió dels resultats del model. YOLOv8 utilitza dues funcions de pèrdua, una per a la pèrdua dels quadres delimitadors i una altra per a la pèrdua de classificació. En concret utilitza la *Complete Intersection over Union* o CIoU i la *Distribution Focal Loss* o DFL, funcions que han permès millorar el rendiment de la detecció d'objectes, sobretot quan es tracta de detecta objectes petits.

2.4.1.1 CIoU (Complete Intersection over Union)

En termes de mètriques per avaluar la regressió dels quadres delimitadors, IoU és la mètrica més popular,

$$IoU = \frac{|B \bigcap B^{gt}|}{|B \bigcup B^{gt}|} \tag{2.35}$$

on $B^{gt} = (x^{gt}, y^{gt}, w^{gt}, h^{gt})$ són els verdaders valors del quadre delimitador i $B = (x, y, w, h)$ són les prediccions del quadre. Quedant així la funció de pèrdua,

$$\mathscr{L}_{IoU} = 1 - \frac{|B \bigcap B^{gt}|}{|B \bigcup B^{gt}|} = 1 - IoU \tag{2.36}$$

Per tal que la funció de pèrdua funcioni en aquelles situacions en les que els quadres delimitadors no estan solapats, cal afegir un terme de penalització, $R(B, B^{gt})$.

$$\mathscr{L}_{IoU} = 1 - IoU + R(B, B^{gt}) \tag{2.37}$$

Comple Intersection over Union és la versió més completa de la mètrica IoU. En aquesta, la funció de penalització tracta de respondre a dues preguntes:

1. És factible minimitzar directament la distancia normalitzada entre el quadre delimitador i la seva predicció per tal de conseguir una convergència més ràpida?

   Es proposa minimitzar la distancia normalitzada entre els punts centrals dels quadres delimitadors.

   $$\frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2} \tag{2.38}$$

   On $\mathbf{b}$ i $\mathbf{b}^{gt}$ denoten els punts centrals de B i B^{gt}, $p()$ és la distància euclidiana i c és la longitud diagonal del quadre més petit que cobreix els dos quadres delimitadors.

2. Com fer que la regressió sigui més precisa i ràpida quan es superposen o fins i tot quan hi ha inclusió dins el quadre delimitador objectiu?

   Per tal de realitzar-ho, cal imposar la consistència de la relació d'aspecte.

   $$R_{CIoU} = \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2} + \alpha v \tag{2.39}$$

   on α és un paràmetre de compensació positiu i v mesura la consistència de la relació d'aspecte,

   $$v = \frac{4}{\pi^2}(\arctan\frac{w^{gt}}{h^{gt}} - \arctan\frac{w}{h})^2 \tag{2.40}$$

I el paràmetre de compensació es defineix com,

$$\alpha = \frac{v}{(1-IoU)+v} \tag{2.41}$$

donant així major prioritat al factor d'àrea superposada per a la regressió.

De manera que la funció de pèrdua és,

$$\mathscr{L}_{CIoU} = 1 - IoU + R_{CIoU} = 1 - \frac{|B \bigcap B^{gt}|}{|B \bigcup B^{gt}|} + \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2} + \alpha v \tag{2.42}$$

2.4.1.2 DFL (Distribution Focal Loss)

El *Focal Loss* original es va proposar per adreçar-se als escenaris de detecció d'objectes en una sola etapa on hi havia un desequilibri extrem entre les classes de primer pla (*foreground*) i de fons (*background*).

$$\mathbf{FL}(p) = -(1-p_t)^{\gamma}\log(p_t), \quad p_t = \begin{cases} p, & \text{when } y = 1 \\ 1-p, & \text{when } y = 0 \end{cases} \tag{2.43}$$

on $y \in \{1,0\}$ especifica la classe i $p \in [0,1]$ denota la probabilitat estimada per la classe d'etiqueta $y = 1$. γ és el paràmetre d'enfocament ajustable.

Específicament, FL consisteix d'una part estàndard d'entropia creuada $-log(p_t)$ i una part de factor d'escala dinàmic $(1-p_t)^{\gamma}$, on el factor d'escala redueix automàticament la contribució d'exemples fàcils durant l'entrenament i centra ràpidament el model en els exemples durs.

Per tal de solucionar el problema d'inconsistència entre les fases d'entrenament i predicció, es va presentar el *Quality Focal Loss* (QFL), el qual consisteix en una representació unida de la qualitat de localització i la puntuació de classificació.

En aquest s'amplia la part d'entropia creuada $-log(p_t)$ a la seva versió completa $-((1-y)\log(1-\sigma)+y\log(\sigma))$. La part del factor d'escala $(1-p_t)^{\gamma}$ es generalitza a la distancia absoluta entre l'estimació σ i l'etiqueta continua y, $|y-\sigma|^{\beta}$. Combinant les dues parts ampliades es formula l'objectiu de pèrdua complet.

$$QFL(\sigma) = -|y-\sigma|^{\beta}((1-y)log(1-\sigma)+ylog(\sigma))$$

on σ és el resultat de la funció sigmoide, i β és un paràmetre que controla la taxa de ponderació a la baixa suaument.

DFL és una millora en la representació dels quadres, en aquest s'adopten els desplaçaments relatius de la ubicació dels quatre costats d'un quadre delimitador com a objectiu de la regressió.

Convencionalment les operacions de regressió de quadres delimitadors modelen l'etiqueta y com una distribució *Dirac delta* $\delta(x-y)$, on es satisfà $\int_{-\infty}^{+\infty}\delta(x-y)dx = 1$ i normalment s'implementa a través de capes completament connectades.

Al utilitzar DFL es proposa aprendre directament la distribució general subjacent $P(x)$ sense introduir cap altre a priori. Donant el rang de l'etiqueta y ($y_0 <= y <= y_n,\ n \in \mathbf{N}^+$), es pot estimar el valor $\hat{y}$ a partir del model:

$$\hat{y} = \int_{y_0}^{y_n} P(x)xdx$$

Per tal de ser consistent amb les xarxes neuronals convolucionals, es transforma la integral sobre el domini continu a una representació discreta, a través de discretitzar el rang $[y_0, y_n]$ en intervals equivalents $\Delta = y_{i+1} - y_i,\ \forall i \in [0, n-1]$. Conseqüentment, donada la propietat de distribució discreta $\sum_{i=0}^{n} P(y_i) = 1$, es pot estimar el valor $\hat{y}$ de la regressió com:

$$\hat{y} = \sum_{i=0}^{n} P(y_i)y_i$$

Com a resultat, $P(x)$ es pot implementar fàcilment a través d'una capa *softmax* $S()$ que consta de $n+1$ unitats, denotant $P(y_i)$ com a S_i per simplificar.

Com l'aprenentatge dels quadres delimitadors és només per a mostres positives sense el risc de problemes de desequilibri de classe, senzillament s'aplica la part completa de l'entropia creuada a QFL per a la definició de DFL:

$$DFL(S_i, S_{i+1}) = -((y_{i+1} - y)log(S_i) + (y - y_i)log(S_{i+1}))$$

$S_i = \frac{y_{i+1}-y}{y_{i+1}-y_i}$, $S_{i+1} = \frac{y-y_i}{y_{i+1}-y_i}$

Intuïtivament, DFL pretén centrar-se a augmentar les probabilitats dels valors al voltant de l'objectiu. El mínim global de la funció DFL pot garantir que l'estimació de l'objectiu $\hat{y}$ és infinitament proper a l'etiqueta corresponent.

$$\hat{y} = \sum_{j=0}^{n} P(y_i)y_i = S_i y_i + S_{i+1}y_{i+1} = \frac{y_{i+1}-y}{y_{i+1}-y_i}y_i + \frac{y-y_i}{y_{i+1}-y_i}y_{i+1} = y$$

Cosa que també garanteix la seva correcció com a funció de pèrdua.

2.4.2 Aplicacions

Totes les innovacions i avantatges respecte a les seves versions predecessores i altres models de detecció d'objectes fa que YOLOv8 sigui adequat i àmpliament utilitzat a un ampli ventall d'aplicacions, dins el qual s'inclouen:

- Vehicles autònoms:

  Durant la elaboració dels models de conducció autònoma, un dels elements crucials es disposar d'un model de detecció d'objectes precís per tal que els vehicles autònoms naveguin de forma segura i evitant els obstacles.

  Gràcies a la alta precisió de YOLOv8 i la seva detecció espacial, fa que sigui una molt bona alternativa al entrenar aquest tipus de models d'IA.

- Seguretat i vigilància:

  Es pot utilitzar per a la detecció d'anomalies, detecció d'intrusions i seguiment d'objectes a sistemes de seguretat.

- Comerç minorista i fabricació:

  Es pot emprar per a la identificació de productes, gestió d'inventari i control de qualitat.

- Robòtica:

  Permet als robots percebre els seu entorn i interactuar amb els objecte de forma intel·ligent.

- Imatges mèdiques:

  El model pot ajudar en el diagnostic mèdic identificant automàticament objectes dins imatges mèdiques, com tumors o anomalies.

3 Metodologia i aplicació

Per a poder entrenar el model YOLOv8 i que aquest permeti fer un anàlisi predictiu de les senyes arbitrals en el món del bàsquet, el primer del que cal disposar és una bona base de dades amb la que poder entrenar i validar el model. Com l'objectiu és entrenar una xarxa neuronal convolucional capaç de diferenciar les senyes realitzades pels àrbitres, la base de dades haurà d'estar composada d'imatges de partits de bàsquet a les que es pugui apreciar als àrbitres realitzant les senyes.

3.1 Obtenció de la base de dades

Per tal d'obtenir les imatges es va contactar amb la FCBQ, organització que va facilitar un document d'Excel que inclou informació d'un gran nombre de partits. Dins el document d'Excel proporcionat, hi ha un llistat de 432 partits de diferents categories de bàsquet català entre les que hi havia Super Copa, Lliga EBA o Copa Catalunya, entre d'altres. De cada partit es proporciona informació com el nom de la categoria i la fase, el nom de l'equip local i del visitant, la data del partit, una variable binària indicant si el partit havia estat retransmès en directe, prenent valor 1, i 0 en cas contrari, i al final es facilita l'enllaç del vídeo. De tota la informació subministrada, l'única que veritablement es necessita per a la realització del treball és l'enllaç del vídeo.

Veient que els enllaços dels partits no pertanyen a la mateixa plataforma en tots els casos, s'ha optat per considerar com a possibles opcions només aquells partits que disposaven d'un enllaç de Youtube, això amb l'objectiu d'assegurar la possibilitat de descarregar el vídeo. Considerant els diferents colors de les samarretes utilitzades pels àrbitres, que són negre, gris, morat i taronja, s'ha seleccionat dos partits per a cadascun dels colors de l'equipació dels àrbitres, de manera que es tindria una mostra de vuit partits. Malauradament, a cap dels partits dels quals es disposava d'un enllaç de Youtube s'utilitzava la samarreta taronja, de manera que s'ha optat per partir d'una primera mostra de sis partits.

Un cop seleccionada la mostra de sis partits, el següent pas és descarregar els vídeos. Per tal de realitzar-ho s'ha definit una funció a la que s'ha anomenat *Download*. Dins la funció s'utilitza la funció *Youtube* del paquet *pytube* de Python per a poder guardar el vídeo de Youtube dins un objecte per descarregar-lo mitjançant la funció base de Python *download*.

Ja havent descarregat el vídeos, cal extreure totes les imatges dels mateixos, ja que un vídeo no deixa de ser una concatenació d'imatges una rere l'altre i en funció de la quantitat d'imatges per segon que tingui el vídeo es defineixen els FPS o *Frames Per Second* d'aquest mateix. En aquest cas els vídeos tenien 30 FPS i tenint en compte que la durada dels mateixos era aproximadament de 105 minuts, suposa que de mitjana cadascun dels partits està format per 189.000 imatges.

Amb l'objectiu de reduir el nombre total de *frames* extrets de cada vídeo i, per tant, la càrrega de feina posterior intentant no afectar significativament als resultats, s'ha decidit agafar un de cada 5 *frames* de manera que seria com reduir la quantitat de FPS dels vídeos a 6 FPS.

Per tal d'extreure les imatges s'ha definit la funció *recorteframes*, que justament permet extreure un de cada 5 *frames* de cadascun dels vídeos que se li especifiqui, generant una carpeta per a cadascun dels vídeos i enviant les imatges corresponents a cadascuna d'aquestes. Un cop finalitzada l'extracció dels *frames*, el nombre total d'imatges ascendeix a 193.579.

Cadascuna de les 193.579 imatges s'ha hagut de revisar individualment i seleccionar només aquelles a les que apareix algun àrbitre realitzant una senya. Si existís un model previ per a la predicció de les senyes, s'hagués pogut utilitzar per a la selecció de les imatges, però com no s'ha trobat cap amb aquest objectiu en específic, la tasca s'ha hagut de realitzar manualment, consumint una quantitat de temps considerable. Degut al temps que estava ocupant la revisió individual de les imatges, es va aturar al quart vídeo. En cada instant del partit en el que l'àrbitre realitza una senyalització, hi ha una seqüència d'entre 5 i 10 *frames* en el que es pot apreciar a l'àrbitre realitzant-la. Per tal de no agafar un gran nombre d'imatges pràcticament idèntiques, es realitza una petita selecció d'entre 2 i 4 *frames* de cada seqüència.

Un cop acabada la selecció d'imatges, el nombre d'imatges definitiu de la mostra que s'utilitzarà per a l'entrenament, validació i posterior predicció dels models és de 2907 imatges. Com a qualsevol altre model de ML d'aprenentatge supervisat, els models que es realitzaran requereixen d'una 'variable resposta', que en aquest cas serà una variable categòrica indicant la senya que està realitzant l'àrbitre.

Figura 3.1: Exemple d'etiquetatge (Tir lliure sense rebot)

Per tal de realitzar l'etiquetatge de les imatges s'ha utilitzat Roboflow, que és un plataforma de software que facilita la creació, gestió i el desplegament de models de visió per computadora. També ofereix altres eines per a la anotació i etiquetatge d'imatges, així com per al preprocessament i augment de les dades, amb l'objectiu de millorar la precisió i eficiència dels models d'aprenentatge automàtic.

Dins de Roboflow s'ha generat un projecte al que s'han carregat les 2907 imatges seleccionades prèviament. Com ja s'ha dit, Roboflow permet etiquetar les imatges facilitant una eina que habilita la selecció de l'àrea corresponent a etiquetar. De igual manera que al realitzar la selecció de les 2907 imatges, ara s'han de tornar a revisar una a una per seleccionar l'àrbitre i etiquetar-lo amb el nom de la senya que està realitzant.

Algunes de les senyes que s'han considerat des del procés de selecció d'imatges són, faltes (en atac i en defensa), tècniques, antiesportives, punts (de dos i tres), tirs lliures, canvis, tipus de faltes, per exemple cop al cap, peus o fi de possessió, entre d'altres. Un cop etiquetades les 2907 imatges, la taula de freqüència de les classes és:

Etiqueta	Freqüència
Falta	492
Tir de tres	451
Tir lliure amb rebot	449
Tir lliure sense rebot	342
Dos punts	262
Tir lliure anotat	155
Dos tirs lliures	128
Canvis	120
Tir de tres anotat	89
Temps mort	42
Un tir lliure	41
Agafar del braç	39
Cop al braç	39
Lluita	38
Empenta	32
Passos	29
Cop a la mà	22
Bloqueig	21
Falta en atac	20
Antiesportiva	19
Cop al cap	15
Peus	12
Hand-checking	11
Fi de la possessió	10
Tècnica	6
Tacte	4
Rodejar amb els braços	3
Tres tirs lliures	3

Taula 3.1: Taula de freqüències de les senyes

Com es pot apreciar a la taula 3.2, hi ha un total de 28 classes, de les quals algunes apareixen molt més que unes altres. Això és lògic pensar-ho ja que algunes situacions són molt menys freqüents que unes altres dins els partits. Per exemple, que un defensor faci una falta mentre

un atacant està tirant de tres i que aquest falli i, per tant, l'arbitre senyali falta i tres tirs lliures, és una situació poc comuna, mentre que els tirs de dos punts anotats són una situació molt més repetitiva a un partit. Roboflow proporciona un *Health check* de la base de dades i ens indica que totes les etiquetes des de Temps mort cap abaix estan infrarepresentades.

Un cop etiquetades totes les imatges, es genera des de Roboflow la versió de la base de dades que s'utilitzarà en l'entrenament del model YOLOv8 per al primer escenari. Dins de Roboflow es permet generar diverses versions de la mateixa base de dades per si l'usuari vol afegir etapes de preprocessament de les imatges o augmentacions de la base de dades. Per a tots els escenaris es seleccionen els mateixos processos de preprocessament, que són *auto-orient* i *resize*, les recomanades per Roboflow. La primera s'assegura que totes les imatges tinguin la mateixa orientació i la segona transforma la mida de les imatges, originalment de 1280x720 píxels, a imatges quadrades de 640x640 píxels unificant la grandària de les imatges.

3.2 Primer escenari

Per al primer escenari, s'utilitza com a base de dades el conjunt sencer de 2907 imatges etiquetades amb una de les 28 classes (senyes) considerades, a les quals s'ha aplicat un preprocessament que com ja s'ha dit, incorpora l'*auto-orient* i el *resize* de 1280x720 a 640x640 píxels. Per a aquest escenari no s'ha utilitzat cap de les opcions d'augmentació de la base de dades que disposa Roboflow. Al generar una versió de la base de dades, Roboflow permet realitzar la divisió de la base de dades en els conjunts d'entrenament, validació i prova. Per a tots els escenaris s'ha optat per realitzar una divisió 70/20/10%, de manera que el conjunt d'entrenament disposa de 2035 imatges, el conjunt de validació 581 i el conjunt de prova 291.

3.2.1 Entrenament i validació

Degut a la gran càrrega computacional que suposa l'entrenament del model s'ha optat per realitzar-lo des de Google Colab, que és una plataforma gratuïta basada al núvol que permet escriure i executar codi de Python a través d'una interfície de quadern Jupyter. La part interessant d'utilitzar Google Colab es que posa a la disposició dels usuaris escollir l'entorn d'execució, fent més eficient l'execució de tasques d'aprenentatge automàtic. Depenent de la disponibilitat dels GPU, s'ha treballat amb A1000 GPU o L4 GPU que són processadors especialitzats en renderitzar imatges i vídeos.

Des de Google Colab es descarrega la versió de la base de dades i es procedeix a executar l'entrenament del model. Per a aquest mateix, s'ha escollit no modificar el paràmetres predefinitis i procedir amb els valors recomanats per Ultralytics:

- *epochs* = 100 ⟵ Nombre total d'*epochs* a l'entrenament. Cada *epoch* representa un pas complet per tot el conjunt de dades.
- *patience* = 100 ⟵ Nombre d'*epochs* que cal esperar sense millora a les mètriques de validació abans d'aturar l'entrenament.
- *lr* = 0.01 ⟵ *Learning rate*. Modificar aquest valor és clau per al procés d'optimització, influenciant com de ràpidament s'actualitzen els pesos del model.
- *cls* = 0.5 ⟵ Pes de la pèrdua de la classificació sobre el total de la funció de pèrdua, afectant a la importancia que se li dona a la correcta predicció de classe.
- *dfl* = 1.5 ⟵ Pes de la pèrdua focal de distribució, utilitzat en certes versions de YOLO per a la classificaciño detallada.
- *val* = TRUE ⟵ Habilita la validació durant l'entrenament, permetent una avaluació periòdica del rendiment del model en un conjunt de dades.
- Entre d'altres.

Durant el període de temps que dura l'execució de l'entrenament del model, es van produint unes sortides per a cada *epoch* mostrant diverses mètriques per avaluar quantitativament el rendiment del model en termes d'exactitud, precisió, recall i eficàcia global per detectar i classificar objectes dins d'imatges.

- ***box-loss*** (Per entrenament i validació): Mesura fins a quin punt les prediccions dels quadres delimitadors coincideixen amb els quadres delimitadors verdaders.
- ***cls-loss*** (Per entrenament i validació): Mesura fins a quin punt el model classifica els objectes detectats en les categories correctes.
- ***dfl-loss*** (Per entrenament i validació): Mesura la precisió de les coordenades del quadre delimitador previst centrant-se en exemples difícils de classificar i aprenent la distribució de les coordenades del quadre delimitador.
- ***precision***: Proporció de detección *True Positive* sobre la suma de *True Positives* i *False Positives*.

$$Precision = \frac{TP}{TP+FP} \tag{3.1}$$

- ***recall***: Proporció de *True Positive* sobre la suma de *True Positive* i *False Negatives*.

$$Recall = \frac{TP}{TP+FN} \tag{3.2}$$

- ***mAP50*** (Mean Average Precision): Mètrica d'ús habitual en la detecció d'objectes que mesura la precisió mitjana (mAP) quan el llindar d'intersecció sobre unió (IoU) s'estableix en 0,50.

- ***mAP50-95*** (Mean Average Precision): És la mitjana dels valors de precisió mitjans per a totes les classes en aquests diferents llindars d'IoU. Aquesta mètrica proporciona una avaluació més detallada del rendiment del model, ja que requereix que el model faci prediccions més precises a llindars d'IoU més alts.

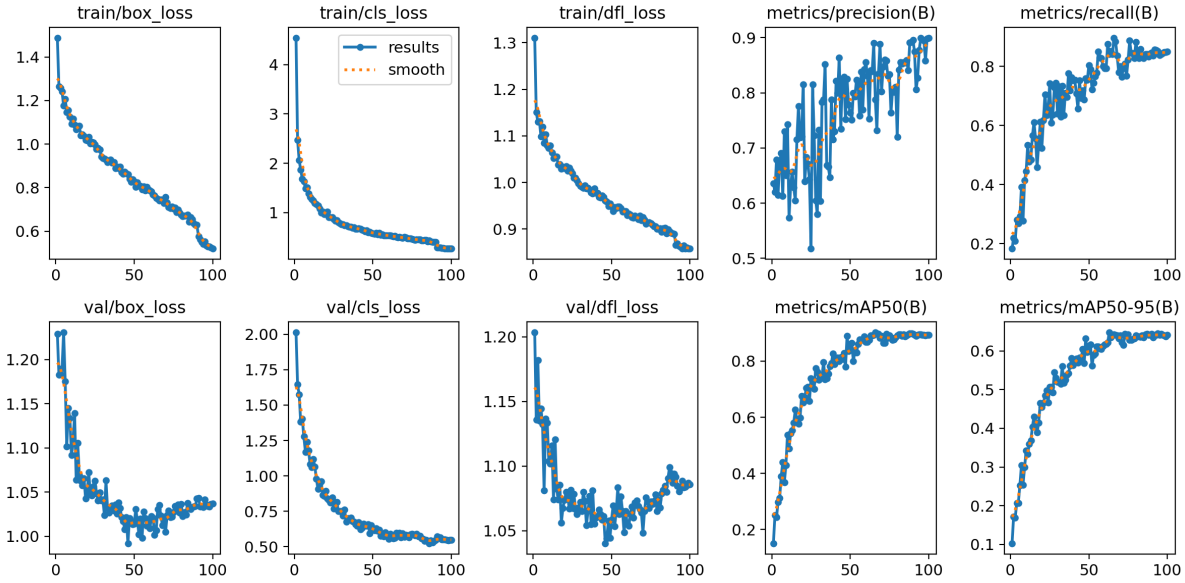


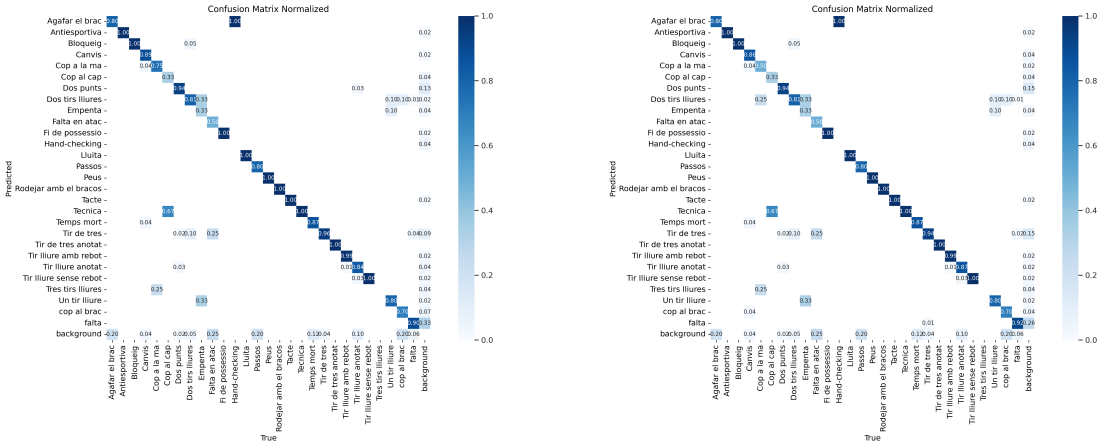
Figura 3.2: Resum de les mètriques de l'escenari 1

Com es pot apreciar a la gràfica del *box-loss* de l'entrenament, té una tendència clarament decreixent cosa que indica que el model està aprenent i millorant en termes de predir amb precisió els quadres delimitadors durant l'entrenament. Pel que fa a l'evolució de la mètrica a la validació, la tendència decreixent dels primers 50 *epochs* indica que inicialment, el model el model aconsegueix millorar la precisió en la localització. Malgrat això, el lleuger increment del *box-loss* produït a partir del *epoch* 50, suggereix dificultat per generalitzar a les noves dades o sobreajustament.

Pel que fa al *cls-loss* es pot veure una tendència similar a l'entrenament i a la validació, amb una tendència decreixent fins a l'*epoch* 50, a partir del qual la mètrica redueix considerablement la tendència decreixent. Aquesta estabilització a l'entrenament, suggereix que el model ha convergit a un punt en el que els *epochs* posteriors no milloraran significativament el rendiment de la classificació de les dades d'entrenament. L'estabilització a la validació indica que el rendiment de la classificació de les dades de validació s'ha estabilitzat, cosa que indica un rendiment coherent.

La constant pendent decreixent del *dfl-loss* a l'entrenament és senyal de que el model està millorant la seva capacitat de predir correctament les coordenades dels quadres delimitadors. De manera similar al que succeeix a la mètrica *box-loss* de la validació, es pot observar com la *dfl-loss* decreix fins a l'*epoch* 50 i a partir d'aquest incrementa lleugerament, cosa que pot ser conseqüència d'un sobreajustament del model.

La clara tendencia creixent de la *precision* i el *recall* indiquen que el model està millorant la seva capacitat de classificar i detectar objectes. Assolint una *precision* i un *recall* al *epoch* 100 de 0,89 i 0,85, respectivament. Pel que fa a *mAP50* i *mAP50-95*, la tendència es creixent durant els primers 60 *epochs*, punts a partir del qual s'estabilitzen en 0,89 i 0,64, respectivament. Les tendències creixents i posteriors estabilitzacions d'aquestes mètriques són indicadors de que el model a assolit un punt de convergència on afegir més *epochs* no millorarà el rendiment del model.



(a) CM Norm Entrenament (b) CM Norm Validació

Figura 3.3: Matrius de confusió normalitzades de l'entrenament i la validació (Escenari 1)

A les matrius de confusió normalitzades de l'entrenament i la validació (afegides en un tamany més gran a l'annex), es pot apreciar que les classes de les quals es disposa de poques instancies, tenen un *recall* o sensitivitat molt menor que les classes de les quals es disposa d'un major nombre d'instàncies. Això pot estar degut al desbalanceig existent a la base de dades, cosa que pot generar un biaix al model cap a les classes de les quals es disposa d'un major nombre d'instàncies.

3.2.2 Conjunt de prova

Després d'entrenar i validar el model YOLOv8, el següent pas és avaluar el rendiment del mateix mitjançant un conjunt d'imatges de prova que el model no ha vist durant l'entrenament o la validació. Això implica executar el model en aquestes imatges de prova per predir les

senyes i els seus quadres delimitadors. En realitzar la predicció es posa com a llindar mínim de confiança de 0.50, és a dir, que només es mostraran aquelles etiquetes que superin el 0.50 de confiança, sent 0 el mínim i 1 el màxim.

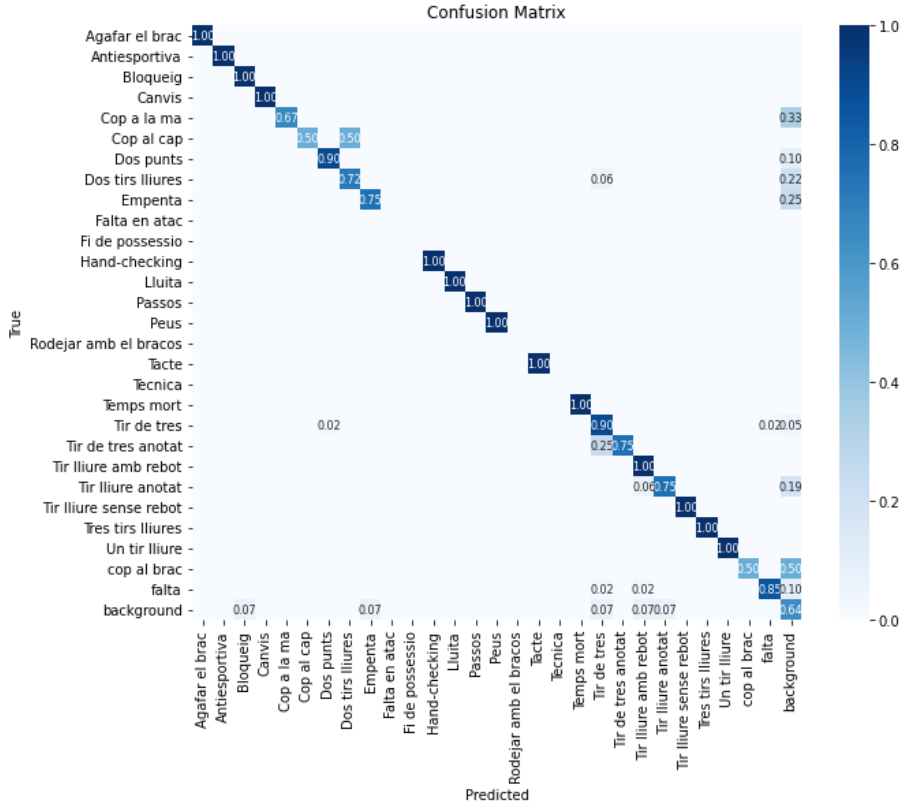


Figura 3.4: Matriu de confusió del conjunt de prova

De igual manera al que es podia veure a les matrius de confusió de l'entrenament i la validació, sembla que el model ha sigut capaç d'aprendre les característiques diferenciadores de les classes de les quals es disposa d'un gran nombre d'instàncies, com Canvis, Dos punts, Tir de tres, Tir lliure amb rebot i sense rebot... Pel que fa a les classes infrarepresentades com Falta en atac, Fi de possessió, Rodejar amb el braços, Tècnica, Cop al cap o Cop al braç, el model no ha sigut capaç d'aprendre les seves característiques amb igual precisió que les de les classes ben representades. Probablement aquesta diferència tan gran en quant a la sensitivitat a l'hora de detectar les classes, és degut, com ja s'ha dit, al propi desbalanceig de la base de dades, generant així un biaix cap a les senyes més representades. Tot i així, l'*accuracy* assolida és de 0.889.

A mode de mostra, es es presenten dos exemples dels resultats proporcionats d'una predicció correcta i una incorrecta.

A l'esquerra es mostra un exemple on el model ha sigut capaç d'identificar correctament la senya realitzada per l'àrbitre, predient correctament que aquest està senyalitzant 'Tir de tres' amb un nivell de confiança de 0.92. A la dreta es mostra la situació oposada, una imatge en la que el model no ha predit bé la senyalització, ja que ha detectat 'Tir de tres' amb una confiança

de 0.85 quan en realitat l'àrbitre està senyalitzant 'Tir de tres anotat'. Tot i predir incorrectament la senyalització realitzada per l'àrbitre, es pot veure que en ambdues situacions ha sigut capaç de localitzar correctament el quadre limitador al voltant de l'àrbitre.

(a) Exemple de predicció correcta a l'escenari 1

(b) Exemple de predicció incorrecta a l'escenari 1

Figura 3.5: Exemples de prediccions del conjunt de prova original (escenari 1)

Com l'objectiu del model es que es pugui arribar a utilitzar en un gran nombre de partits, això implica que haurà de respondre bé a situacions que no ha vist mai, és a dir, en partits que no han sigut utilitzats per al seu entrenament. És per això que s'ha aprofitat el fet de que no s'han utilitzat dos dels sis partits escollits per comprovar si veritablement el model funciona com s'espera. De cadascun dels dos partits que no s'han utilitzat per a l'entrenament s'ha seleccionat una mostra de 10 imatges, conformant una mostra de 20 imatges entre els dos. Al tractar-se d'una mostra molt reduïda, no apareixen totes les classes.

Com es pot veure a la matriu de confusió 3.6, sembla que el model no és capaç de generalitzar les característiques apreses en imatges de partits que no ha vist el model, ja que només s'assoleix una *accuracy* de 0.235. Aquest problema pot estar degut a diversos factors:

- Sobreajustament: El model pot haver sobreajustat certes condicions presents en el conjunt d'entrenament, fent que rendeixi correctament per imatges que s'assemblen molt, però malament per imatges noves.
- Falta de diversitat a les dades: Falta d'imatges d'un major nombre de partits.
- Augment de les dades: Falta d'augment de dades, resultant en un rendiment bo per aquelles imatges similars a les del conjunt d'entrenament i pobre per imatges noves.
- Desbalanceig de classes.

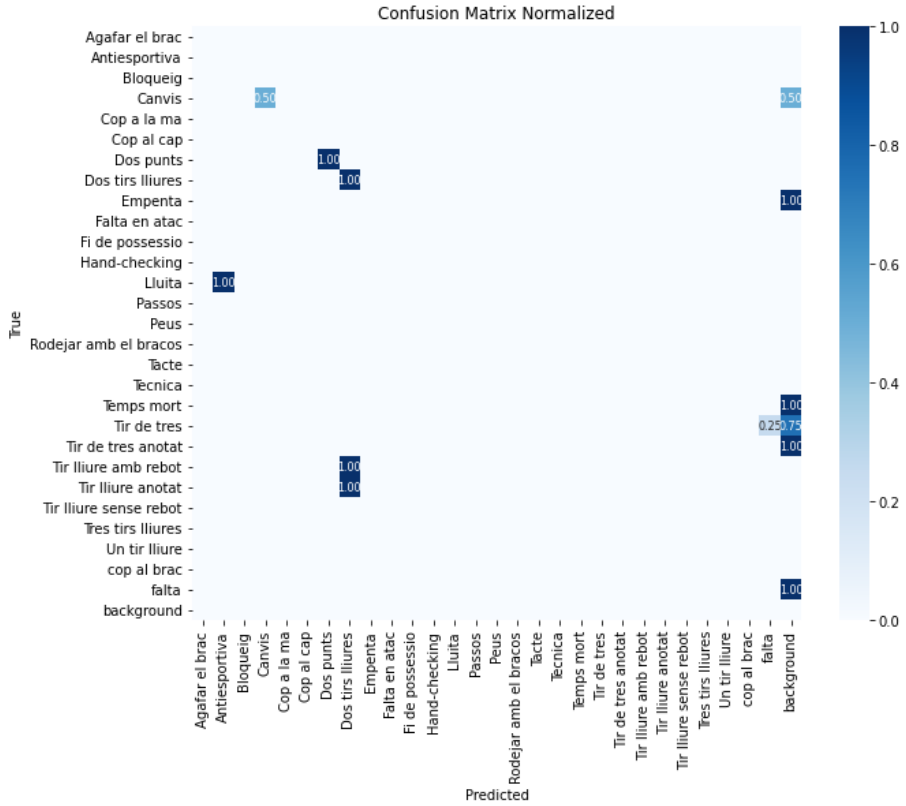
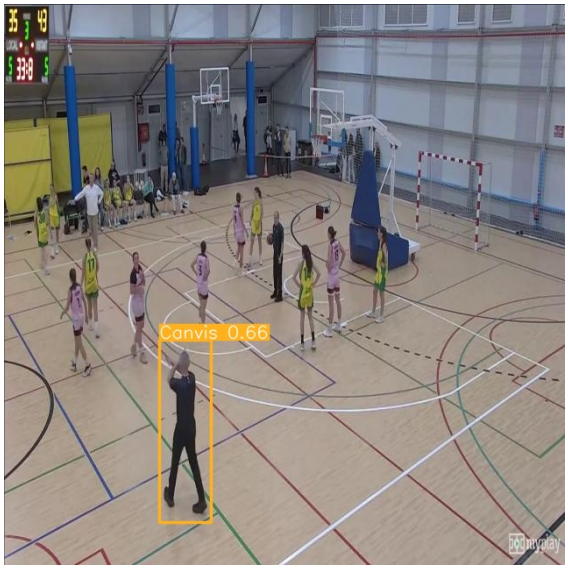


Figura 3.6: Matriu de confusió del conjunt de prova 2

De igual manera que s'ha fet amb el conjunt de prova d'imatges dels partits utilitzats a l'entrenament, es procedeix a mostrar un cas de predicció correcta i un de incorrecta.



(a) Exemple de predicció correcta a l'escenari 1

(b) Exemple de predicció incorrecta a l'escenari 1

Figura 3.7: Exemple de prediccions a partits fora de la mostra original (escenari 1)

A la imatge de l'esquerra es mostra un exemple on s'ha predit correctament la senyalització marcada per l'àrbitre, predient 'Canvis' amb una confiança de 0.66. D'altra banda, a la imatge

de la dreta es pot apreciar que s'ha predit 'Dos tirs lliures' amb una confiança de 0.88, quan en realitat l'àrbitre està senyalitzant 'Tir lliure amb rebot'. D'aquesta manera es comprova que el model té problemes a l'hora d'haver de generalitzar les característiques ja que confon les classes.

3.3 Segon escenari

Veient que un dels possibles problemes del primer escenari és la falta d'augment de les dades, s'ha realitzat una nova versió de la base de dades afegint justament un augment de la base de dades al afegir noves versions de les imatges de manera que la base de dades passa a conformar-se per un conjunt de 6953 imatges.

- En blanc i negre:

  Amb l'objectiu que el model desvinculi el color de les imatges com una característica rellevant, de manera que es centri en la forma de l'objecte a classificar, en aquest cas la senya realitzada per l'àrbitre (28 senyes en total).

- Amb saturació:

  Versions de les imatges amb saturació entre el -34% i 34%. De manera que es força al model a aprendre a reconèixer l'objecte en situacions on la intensitat de la il·luminació i dels colors canvia.

3.3.1 Entrenament i validació

El procés seguit per a l'entrenament ha sigut el mateix que al primer escenari, mantenint tots els paràmetres idèntics. De igual manera que en el primer escenari, la validació es va duent a terme a mesura que es van realitzant els 100 *epochs* de la fase d'entrenament i en finalitzar es retorna el gràfic resum de les mètriques d'avaluació 3.8.

Com es pot apreciar, l'evolució a l'entrenament de les funcions *box-loss*, *cls-loss* i *dfl-loss* són pràcticament idèntiques a les obtingudes al primer escenari. Cal destacar que comencen i finalitzen amb valors més petits que els obtinguts a l'escenari 1, de manera que es pot afirmar que durant l'entrenament el model sembla haver estat capaç d'aprendre més sobre les característiques de les classes.

Pel que fa a l'evolució del *box-loss* i del *dfl-loss* de la validació, es pot apreciar que el canvi de tendència decreixent a creixent es produeix en *epochs* previs al 50, que és quan es produïa el canvi de tendencia al primer escenari, i a més la pendent creixent és més pronunciada. Això

de igual manera que en el primer escenari, pot ser símptoma de sobreajustament i inestabilitat a l'hora de generalitzar les característiques apreses a noves dades, que pot estar degut al fet d'haver augmentat les dades sense haver incrementat el nombre de partits de la mostra.

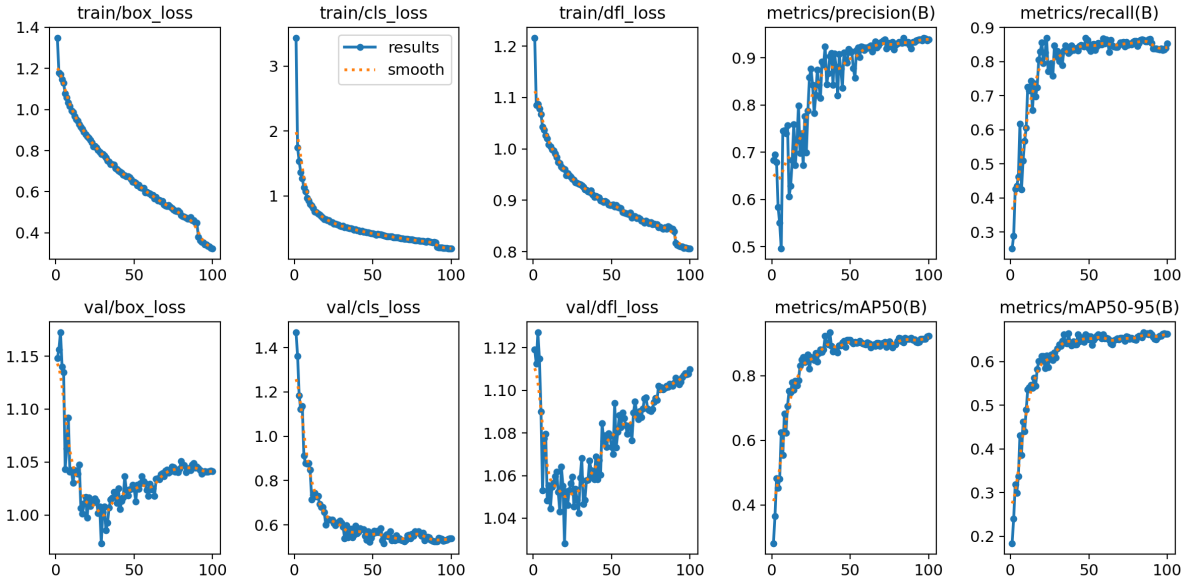
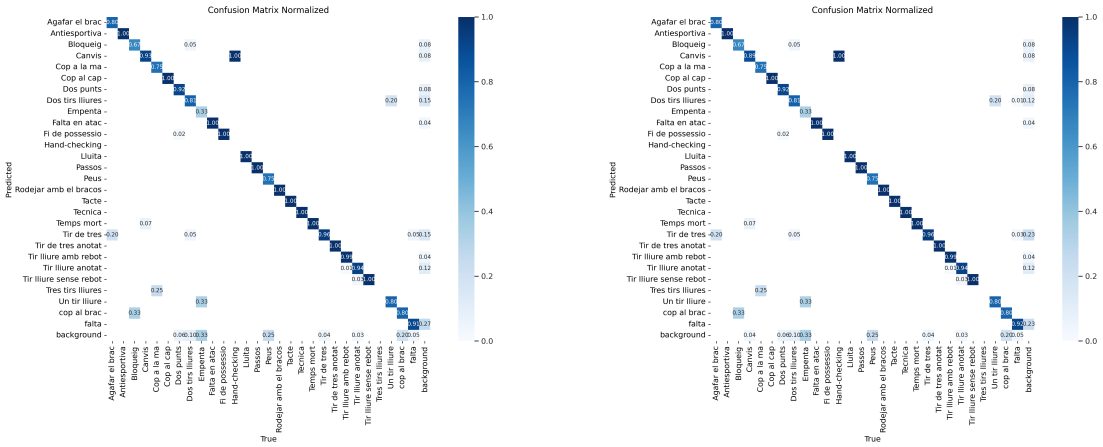


Figura 3.8: Resum de les mètriques de l'escenari 2

Observant l'evolució de la *precision*, el *recall*, el *mAP50* i el *mAP50-95*, tots ells acaben convergint a l'*epoch* 100 a uns valors que són 0.94, 0.85, 0.92 i 0.66, respectivament. El model pren valors iguals o superiors de les mètriques avaluatives, respecte a els valors que prenien al primer escenari, cosa que podria significar un increment al rendiment. Cal tenir en compte, com ja s'ha comentat, que a la validació es poden veure senyals d'un possible sobreajustament i a més, el fet que hi hagi un clar desbalanceig a la base de dades fa que el model generi un biaix clar cap a les classes de les quals es disposen més instàncies.



(a) CM Norm Entrenament (b) CM Norm Validació

Figura 3.9: Matrius de confusió normalitzades de l'entrenament i la validació (Escenari 2)

Pel que es pot apreciar a les matrius de confusió normalitzades d'entrenament i validació del segon escenari i si es comparen amb les del primer escenari, sembla que el model, gràcies a l'augment de les dades has sigut capaç de millorar la detecció de les classes que disposen d'un menor nombre d'instàncies, és a dir, ha sigut capaç d'aprendre amb major precisió les característiques diferencials de les classes infrarepresentades.

3.3.2 Conjunt de prova

Per al segon escenari es repeteix el mateix procés que al primer escenari, realitzant les prediccions del conjunt de prova original per comprovar el rendiment del model en imatges dels partits de la mostra original i després repetint el procés amb el conjunt de prova de 20 imatges dels dos partits que no s'han utilitzat a la mostra, podent comprovar així el rendiment del model en imatges diferents a les vistes durant l'entrenament.

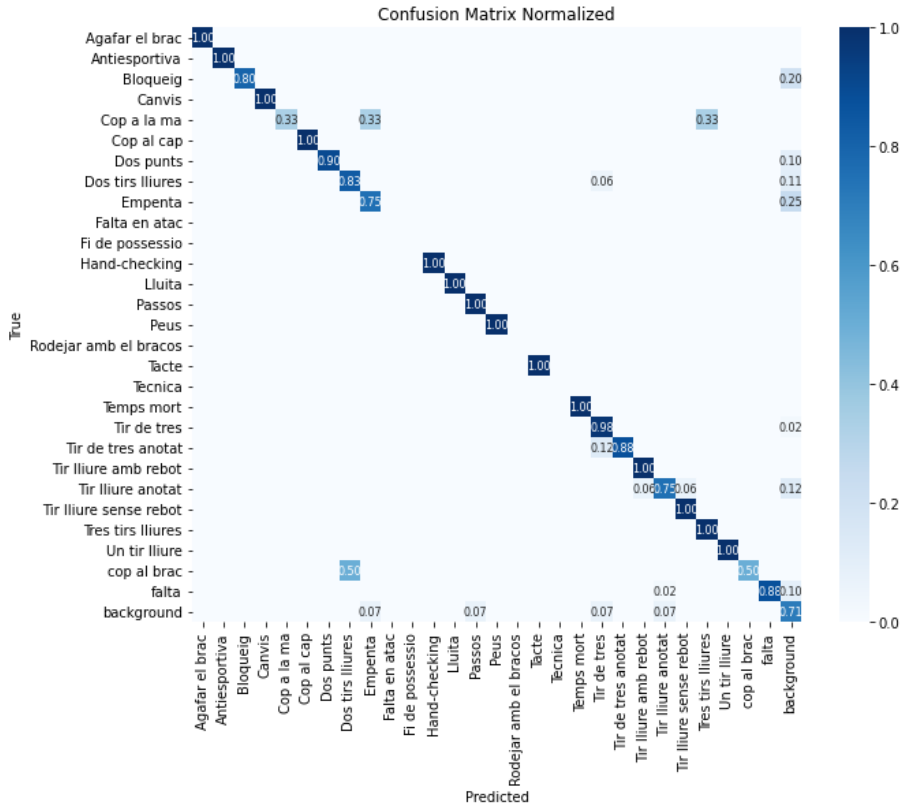


Figura 3.10: Matriu de confusió del conjunt de prova original (Escenari 2)

Tot i que sigui complicat comparar la matriu de confusió obtinguda amb la del primer escenari, si es calcula l'*accuracy* aquesta dona 0.914 de manera que es pot confirmar que el model a millorat el rendiment de la classificació respecte a l'obtingut a l'escenari 1. El fet d'haver realitzat l'augment de les dades sembla haver ajudat a que el model millori el rendiment a l'hora de classificar les classes infrarepresentades.

D'igual manera que en el primer escenari, es procedeix a mostrar els resultats d'una classificació correcta i una incorrecta.

Al primer cas es pot apreciar que el model és capaç de predir correctament la senya realitzada per l'àrbitre, predient 'Tir lliure amb rebot' amb una confiança de 0.93. D'altra banda, a l'exemple de la dreta el model no és capaç de predir correctament, ja que l'àrbitre està senyalant 'Falta', mentre que el model prediu 'Tir lliure anotat' amb una confiança de 0.60.

(a) Exemple de predicció correcta a l'escenari 2 (b) Exemple de predicció incorrecta a l'escenari 2

Figura 3.11: Exemple de prediccions de la mostra original (escenari 2)

Si es grafica la matriu de confusió per a les prediccions de les imatges de partits externs a la mostra original 3.12 es pot veure que el model ha empitjorat la seva capacitat de generalitzar les característiques apreses durant l'entrenament. De fet l'*accuracy* ha empitjorat respecte a la del primer escenari, ha passat de 0.235 a 0.176.

Aquest empitjorament de la capacitat del model per generalitzar les característiques apreses a noves instancies pot estar degut, com ja s'ha comentat durant la validació, al fet d'haver realitzat l'augment de dades sense haver incrementat el nombre de partits considerats a la mostra, és a dir, el nombre d'imatges ha incrementat però el nombre de situacions no, de manera que el model ha après només a diferenciar les situacions ja conegudes, motiu pel que funciona tant bé al conjunt de prova de la mostra.

Com s'ha pogut comprovar, al realitzar les prediccions del conjunt de prova original el model és capaç de predir correctament les classes, ja que les imatges són similars a les utilitzades durant l'entrenament, però al intentar predir les senyalitzacions realitzades pels àrbitres en imatges de partits que no han sigut utilitzats a l'entrenament, la xarxa neuronal no es capaç de generalitzar les característiques apreses per reconèixer les senyes a les noves imatges.

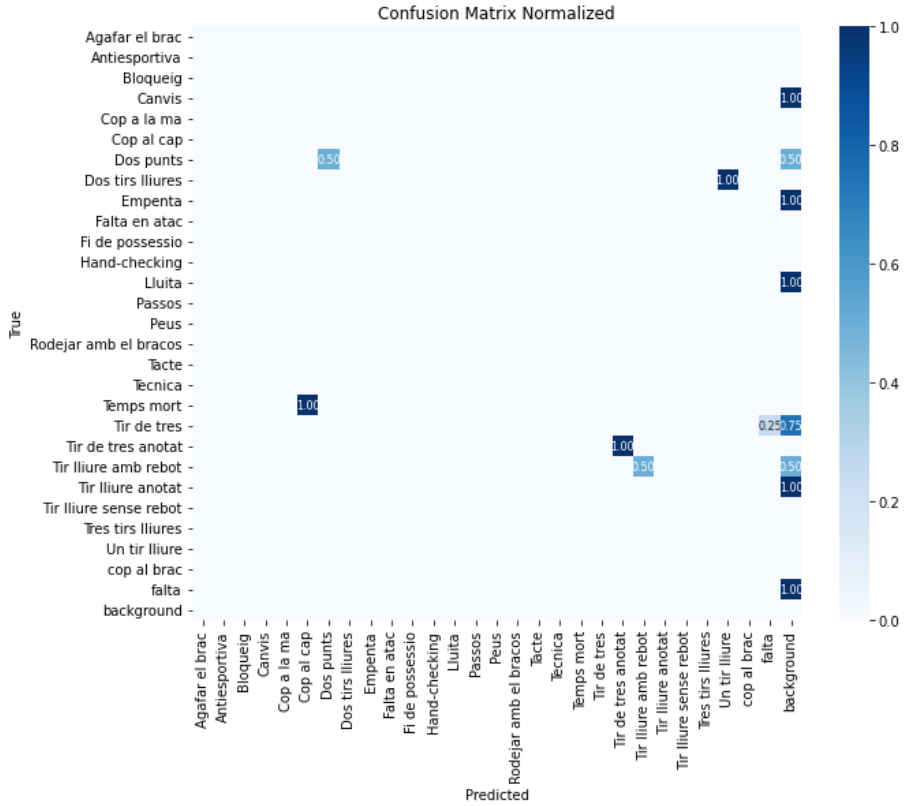


Figura 3.12: Matriu de confusió del conjunt de prova 2

La majoria de prediccions no superen el llindar mínim de 0.5 de confiança, és per això que la majoria de prediccions de la xarxa són *background*. Cal destacar que les poques prediccions que és capaç de realitzar correctament pertanyen a les classes que no es troben infrarepresentades.

(a) Exemple de predicció correcta a l'escenari 2

(b) Exemple de predicció incorrecta a l'escenari 2

Figura 3.13: Exemple de prediccions a partits fora de la mostra original (escenari 2)

A l'exemple de l'esquerra el model es capaç de predir correctament que l'àrbitre està senyalitzant 'Tir de tres anotat' amb una confiança de 0.66. En canvi, a l'exemple de la dreta es pot veure que el model no ha retornat cap senya, és a dir, el model no ha detectat que classe amb una confiança superior al llindar de confiança de 0.5 i, per tant, ha detectat la imatge com a 'background'.

3.4 Tercer escenari

Veient que la xarxa no sembla ser capaç de generalitzar les característiques apreses en aquelles imatges de partits externs a la mostra i que només arriba a predir correctament en alguna ocasió imatges on l'àrbitre està realitzant alguna de les senyes que no estan infrarepresentades a la mostra, s'ha decidit revisar el llistat de classes per veure si algunes d'elles es poden solapar a una sola classe i, sobretot, si totes elles són veritablement rellevants. Reduint el nombre de classes s'espera que la xarxa tingui una menor dificultat per trobar les diferencies entre elles i, per tant, millori la seva capacitat de classificació. Els canvis realitzats són:

- **No distingir entre 'Falta' i 'Falta en atac'**: S'ha decidit no distingir entre aquests dos tipus de falta ja que en el fons les conseqüències dins el partit per al jugador que la realitza són les mateixes. Ambdues classes s'han sobreescrit a 'falta'.

- **No distingir entre 'Tir lliure sense rebot' i 'Tir lliure amb rebot'**: Ambdues classes han estat sobreescrites a la etiqueta 'Tir lliure'.

- **Eliminar la diferenciació del tipus d'agressió realitzada**: Es mantenen les senyalitzacions de 'Falta', 'Tècnica' i 'Antiesportiva', però s'eliminen les senyalitzacions posteriors indicant quina ha sigut l'agressió realitzada com 'Empenta', Cop al cap' o 'Hand-checking', entre d'altres. Ja que no afecten al partit.

Com es pot veure a la taula 3.2, s'ha aconseguit reduir el nombre de classes de 28 a 17. Haver reduït el nombre de classes considerades no vol dir que les imatges que tenien anotades les classes eliminades ja no es considerin a l'entrenament del model, aquestes seran utilitzades com a exemples de 'Background', mostrant així a la xarxa situacions on no ha de reconèixer cap senyalització. De igual manera que al segon escenari, s'han utilitzat les augmentacions de les imatges amb versions de les mateixes en blanc i negre i saturades entre el -34% i el 34%. Al considerar el mateix nombre d'imatges i els mateixos augments de dades, el nombre d'imatges totals de la base de dades és el mateix que al segon escenari, 6953 imatges en total amb la diferencia d'haver reduït considerablement el nombre de classes.

Etiqueta	Freqüència
Tir lliure	791
Falta	512
Tir de tres	451
Dos punts	262
Tir lliure anotat	155
Dos tirs lliures	128
Canvis	120
Tir de tres anotat	89
Temps mort	42
Un tir lliure	41
Lluita	38
Passos	29
Antiesportiva	19
Peus	12
Fi de la possessió	10
Tècnica	6
Tres tirs lliures	3

Taula 3.2: Taula de freqüències de les senyes després de les modificacions

3.4.1 Entrenament i validació

Per tal de no alterar el procés d'entrenament i comparar si en realitzar els canvis es produeix alguna millora respecte als escenaris previs s'ha realitzat el mateix entrenament, mantenint tots els valors dels paràmetres com fins ara.

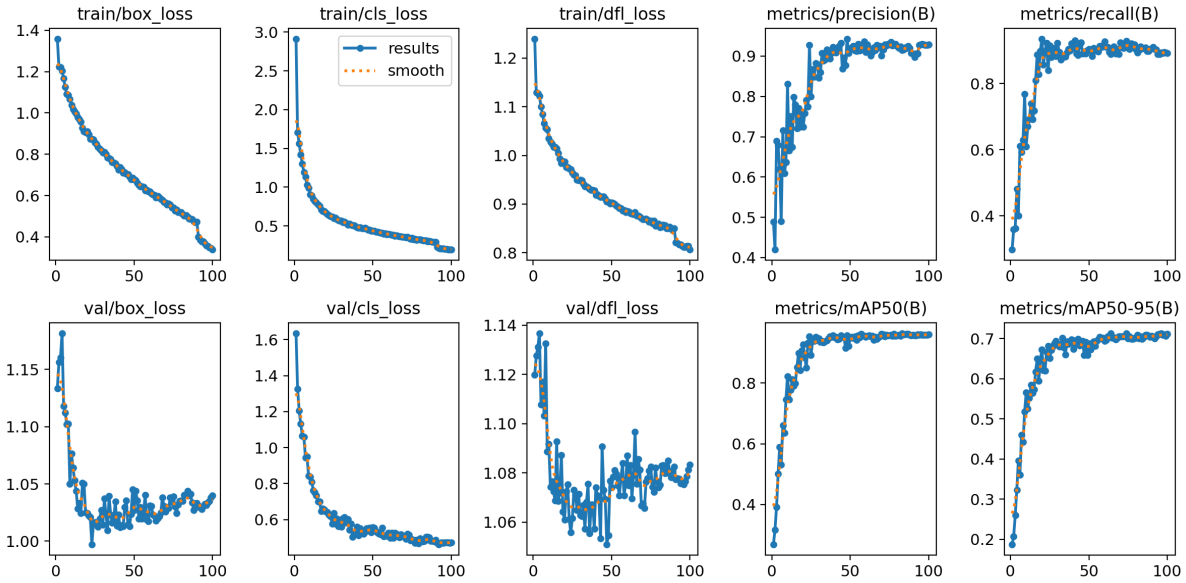


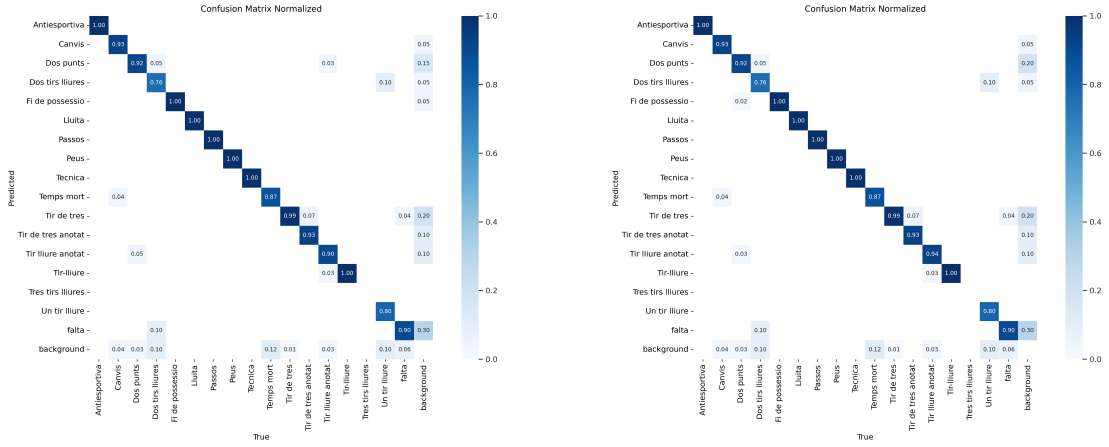
Figura 3.14: Resum de les mètriques de l'escenari 3

Sembla que la reducció del nombre de classes no ha tingut cap efecte significatiu en l'evolució de les funcions de pèrdua *box-loss*, *dfl-loss* i *cls-loss* de l'entrenament, ja que tant al primer *epoch* com a l'últim i durant tots el intermedis els valors són similars als de l'escenari 2.

Pel que fa a l'evolució de les mateixes a la fase de validació, sembla que s'ha aconseguit eliminar part de la pendent creixent que apareixia a l'escenari 2 a partir de l'*epoch* 30. Això pot estar degut a la pròpia reducció del nombre de classes, ja que ara la complexitat que ha de suportar el model és menor, cosa que condueix a una millor generalització al conjunt de validació.

Com als escenaris previs, els valors de *precision*, *recall*, *mAP50* i *mAP50-95* convergeixen als valors 0.93, 0.89, 0.96 i 0.71, respectivament. L'increment produït respecte a l'escenari 2 al *mAP50* i *mAP50-95* és consistent amb el fet d'haver reduït el nombre de classes ja que ara el model a de distingir entre un nombre menor de categories, reduïnt la complexitat de la tasca de classificació.

Amb un nombre major de classes hi ha una major possibilitat de confusió entre classes similars, de forma que reduir el nombre de classes ha ajudat a que el model pugui establir límits de decisió més clars.



(a) CM Norm Entrenament (b) CM Norm Validació

Figura 3.15: Matrius de confusió normalitzades de l'entrenament i la validació (Escenari 3)

Tot i haver utilitzat les imatges de les classes descartades per incorporar exemples al model de situacions a les que no ha de detectar cap senya, no ha sigut capaç de predir cap dels exemples 'background' com a 'backgond', confonent-los sempre amb alguna de les classes.

Com es pot comprovar el *recall* de les classes considerades es manté prenent valors elevats per a totes les classes, a excepció de la classe 'Tres tirs lliures' que degut a la seva poca representació, totes les instàncies semblen haver anat al conjunt de prova. Això té com a conseqüència que el model no hagi après a diferenciar-lo de la resta, però degut a la seva poca presencia es continua sense modificacions.

3.4.2 Conjunt de prova

Com es pot veure a la matriu de confusió normalitzada del conjunt de prova original 3.16, sembla que de igual manera que a l'escenari 2 el model mostra un bon rendiment al reconèixer la gran majoria de classes, assolint una *accuracy* de 0.914. Com ja s'ha comentat prèviament, el fet d'utilitzar un nombre molt reduït de partits i que la quantitat de *frames* de les mateixes situacions sigui relativament elevat, fa que model pateixi de sobreajustament i que al predir imatges molt similars a les vistes durant l'entrenament, la precisió sigui molt elevada.

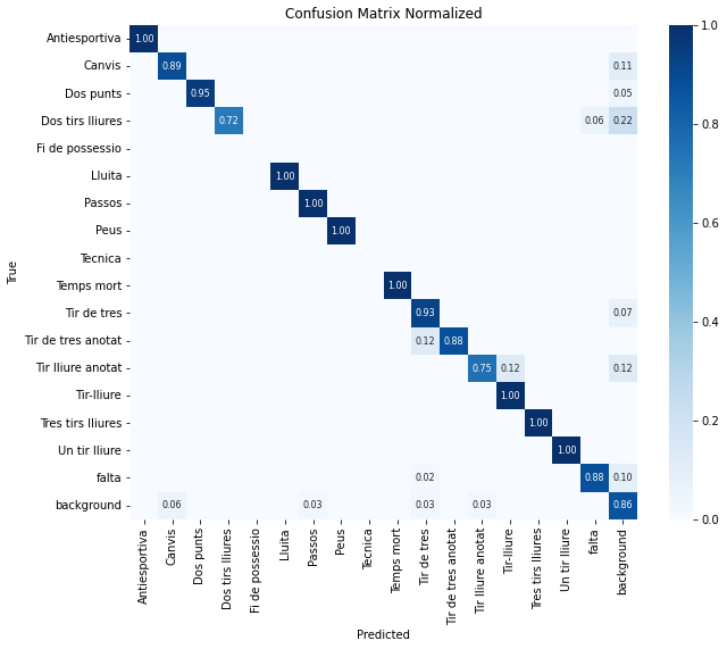


Figura 3.16: Matriu de confusió del conjunt de prova original (Escenari 3)

(a) Exemple de predicció correcta a l'escenari 3

(b) Exemple de predicció incorrecta a l'escenari 3

Figura 3.17: Exemple de prediccions de la mostra original (escenari 3)

A la imatge de l'esquerra es mostra una situació on el model ha sigut capaç de detectar als dos arbitres realitzant la senyalització de 'Tir de tres' amb una confiança de 0.9 i 0.72 a cadascun. Això demostra que el model es capaç de detectar múltiples objectes a les imatges. D'altra banda, a la imatge de la dreta es pot observar com l'àrbitre està senyalitzant 'Falta', però el model no detecta cap senyalització amb una confiança superior a 0.5.

Pel que fa al rendiment del model al conjunt de prova d'imatges de partits externs a la mostra original 3.18, es pot apreciar que com als dos escenaris previs el model no es capaç de generalitzar de les característiques apreses, assolint una *accuracy* de 0.235.

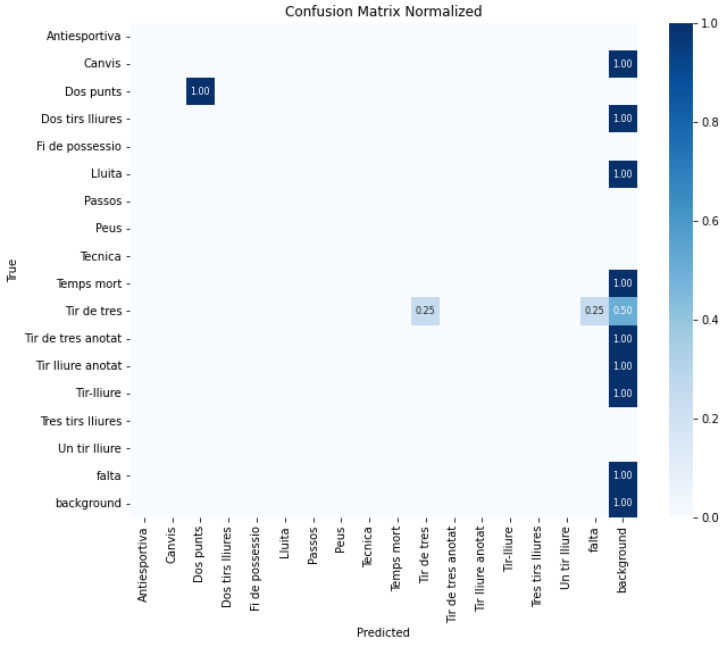
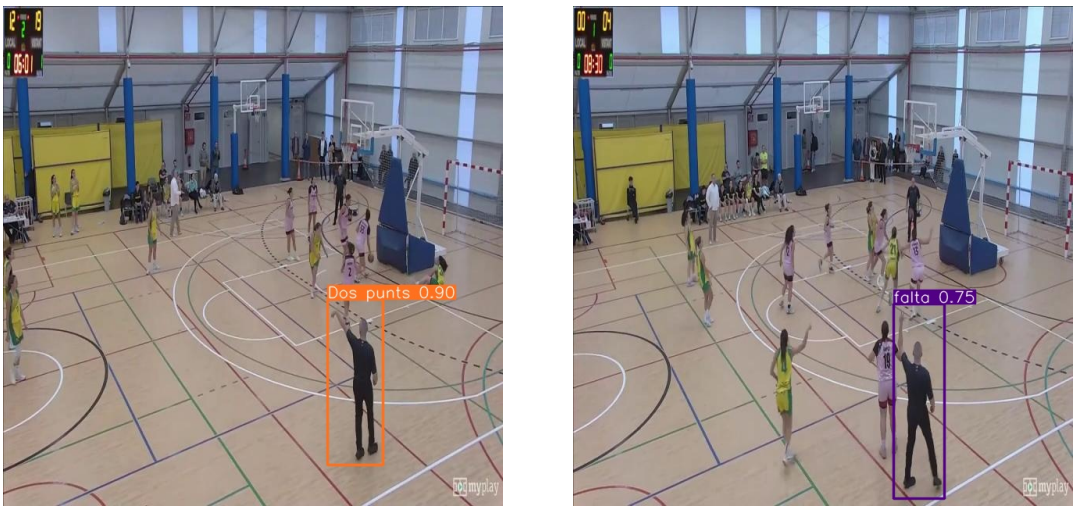


Figura 3.18: Matriu de confusió del conjunt de prova 2



(a) Exemple de predicció correcta a l'escenari 3

(b) Exemple de predicció incorrecta a l'escenari 3

Figura 3.19: Exemple de prediccions a partits fora de la mostra original (escenari 3)

4 Conclusions

Aquest treball de fi de grau s'ha realitzat amb l'objectiu d'entendre el funcionament intern de les xarxes neuronals, procés en el que s'ha pogut demostrar que gran part del contingut de les mateixes són coneixements treballats dins la carrera d'estadística. A més, s'ha realitzat una aplicació inicial amb el model YOLOv8 per a fer un anàlisis predictiu de les senyes arbitrals en el món del bàsquet, proposant tres escenaris diferents cadascun dels quals buscava millorar els resultats de l'anterior i en els que el valors dels paràmetres d'entrenament s'han mantingut sempre igual.

S'ha hagut de generar una base de dades des de zero, podent experimentar tot el procés a realitzar, trobant dificultats durant la construcció i veient la importància de disposar d'unes bones dades on la freqüència de les classes ha d'estar balancejada per a que el model pugui aprendre correctament les característiques diferencials de les classes. Ja havent-se finalitzat el treball es pot afirmar que el procés de selecció i posterior etiquetatge de les imatges ha estat la fase de l'aplicació que ha generat major càrrega de treball manual i, per tant, la part que major espai temporal ha ocupat.

El primer escenari de tots considerava com a base de dades el conjunt de 2907 imatges seleccionades i etiquetades amb una de les 28 classes (senyes arbitrals) considerades del conjunt total de 193579 imatges extretes dels vídeos dels partits, incorporant com a preprocessament les funcions *auto-orient* i *resize* disponibles a Roboflow amb l'objectiu d'assegurar que totes les imatges tenen la mateixa quantitat de píxels.

L'evolució de les funcions de pèrdua *box-loss* i *dfl-loss* de la validació al llarg dels 100 *epochs* pot suggerir sobreajustament o dificultats per generalitzar les característiques apreses durant l'entrenament, condicions que podien ser causades pel desbalanceig present a la base de dades. D'altra banda els valors elevats, propers a 0.9, de la *precision*, *recall* i *mAP50* indiquen un bon rendiment del model Yolov8.

En aplicar el model del primer escenari per fer les prediccions a les imatges del conjunt de prova original es pot apreciar que el model sembla tenir un rendiment extraordinari, assolint una *accuracy* de 0.889. Amb l'objectiu de comprovar si l'excelent rendiment mostrat al conjunt de prova original es replica al posar en pràctica el model en una situació real, és a dir, si l'*accuracy* és igual de elevada al realitzar prediccions en imatges de partits que no han sigut utilitzats durant l'entrenament, s'utilitza el model per fer les prediccions sobre un conjunt de 20 imatges de partits externs a la mostra original. Es pot apreciar que el rendiment del model és pobre, mostrant dificultats per a detectar les classes i confonent-les en un gran nombre de les prediccions. Les poques prediccions correctes pertanyen a les classes amb major nombre d'instàncies a la mostra d'entrenament.

En el segon escenari, veient els problemes de biaix, cap a les classes més representades, presents

al primer escenari i la pobre capacitat per generalitzar les característiques apreses, es procedeix a aplicar un augment de les dades per tal de tenir un major nombre d'instàncies de les classes infrarepresentades a la mostra. S'afegeixen versions de les imatges en blanc i negre i saturades de les imatges que s'espera que desvinculin els colors de les imatges de maners que el model millori a l'hora de predir les senyes en el conjunt de prova extern a la mostra.

El fet d'afegir les noves versions de les imatges sembla haver millorat l'aprenentatge durant l'entrenament ja que les funcions de pèrdua prenen valors menors i la *precision*, el *recall* i les *mAP* valors més elevats, però en observar l'evolució d'aquestes a la validació es pot observar que l'augment de dades no ha tingut l'efecte esperat. Durant les 25/30 primeres etapes de l'entrenament el model sembla estar millorant el seu rendiment, ja que els valors de les funcions de pèrdua es van reduint, però a partir de l'*epoch* 30 aquests valors comencen a incrementar-se fins arribar a prendre valors molt similars als observats a l'escenari 1.

En realitzar les prediccions sobre el conjunt de prova original es pot observar una lleugera millora del rendiment del model, assolint una *accuracy* de 0.914. Però en utilitzar el conjunt de prova extern a la mostra, el model sembla haver empitjorat la seva capacitat per generalitzar les característiques a noves imatges, incrementant el nombre d'instàncies predites com a nul·les ('background'). Mentre a l'escenari 1 s'havia aconseguit una *accuracy* de 0.235, en el segon escenari aquesta s'ha vist reduïda fins a 0.176.

Com s'ha pogut comprovar, la incorporació de l'augment de dades no ha estat suficient per eliminar el biaix cap a les classes amb major representació. És per això, que s'ha considerat l'eliminació i solapament d'algunes de les senyes considerades, arribant a reduir el nombre de classes de 28 a 17.

El tercer escenari és una nova aplicació del segon, havent rebaixat el nombre de senyes i utilitzant les imatges de les senyes eliminades com a exemples de situacions on el model no ha de predir cap de les 17 senyes restants.

En eliminar una gran nombre de les classes s'ha aconseguit que la tasca assignada al model (classificació) sigui més senzilla. A les gràfiques de validació es pot veure com l'increment dels valors les funcions de pèrdua a partir de l'*epoch* 30 present a l'escenari 2, ha sigut eliminat pràcticament en la seva totalitat, transformant-se en una estabilització del valors de les funcions de pèrdua. A més, el *mAP50* i el *mAP50-95* han presenciat un increment considerable, assolint 0.96 i 0.71, respectivament, a l'*epoch* 100. Aquest increment és consistent amb el fet d'haver reduït el nombre de classes, ja que d'aquesta manera s'ha disminuït la complexitat de la tasca de classificació.

En computar les prediccions del conjunt de prova original el model assoleix una *accuracy* de 0.914, mentre que al computar les prediccions del conjunt de prova extern a la mostra original s'assoleix una *accuracy* de 0.235, idèntica a l'escenari 1.

Com s'ha pogut veure, les modificacions aplicades a la base de dades original no han sigut suficients per millorar el rendiment real del model, obtenint una *accuracy* idèntica a l'escenari 1 i a l'escenari 3. D'aquesta manera es pot concloure que la base de dades utilitzada no compleix amb els requisits necessaris per a entrenar correctament el model i poder obtenir els resultats desitjats. El clar desbalanceig entre classes a la base de dades i la falta de partits diferents utilitzats a la mostra fan que el model pateixi sobreajustament cap a les classes amb major representació i es dificulti la generalització de les característiques apreses a noves imatges, probablement al considera imatges de només 4 partits diferents ocasiona que el model confongui el fons (l'entorn de l'àrbitre) com a part de l'objecte a detectar.

Tot i així, no totes les conclusions de l'aplicació del model són negatives, els bons resultats obtinguts als conjunts de prova originals a tots tres escenaris són un indici de que disposant d'una base de dades representativa i balancejada es podria arribar a entrenar el model YOLOv8 per a realitzar la tasca de classificació al nivell de rendiment desitjat.[9]

5 Línies futures

Com s'ha pogut veure a l'elaboració dels resultats, sembla que els principals problemes venen causat pel fet de que la base de dades generada no és de bona qualitat, pel que els primers canvis a realitzar per a posteriors aplicacions són:

- Incorporació d'un major nombre de partits a la mostra per aconseguir que el model desvinculi el fons de l'objecte a detectar i que la mostra sigui representativa.
- Afegir un major nombre d'imatges de les classes infrarepresentades per aconseguir una base balancejada i que durant l'entrenament del model no es generi biaix cap a cap classe.
- Reducció del nombre de senyes considerades a només aquelles que són veritablement rellevants en el desenvolupament del partit. Fent així que la tasca de classificació sigui el més senzilla possible.
- Incrementar la grandària de la base de dades.

Durant el procés de seleccionar i etiquetar les imatges, el temps que s'ha de dedicar a aquesta tasca és molt elevat. De manera que disposar d'un equip de persones que es dediques a realitzar aquesta tasca conjuntament agilitaria tot el procés.

A més, per tal que la aplicació del model fos professional caldria incorporar *cross-validation* durant la fase d'entrenament per tal d'escollir els valors dels paràmetres que millor s'ajusten al model, reduïnt així el risc de sobreajustament i proporcionant estimacions de rendiment més fiables.

Un cop s'aconseguís un bon rendiment del model, aquest es podria utilitzar per a ajudar a les taules dels partits de bàsquet, que són l'equip de persones que realitza l'acta del partit, a tenir un millor seguiment del mateix i a evitar possibles confusions amb la senyalització realitzada per l'àrbitre.

A més la possibilitat d'aplicar un seguiment temporal de les situacions de partit, podria ajudar als equips a preveure en quins moments dels partits l'equip contrari té major tendència a llençar de tres o després de quines situacions són més propensos a realitzar falta, podent així realitzar estratègies per maximitzar els resultats positius de l'equip al partit.

6 Annex

En aquest apartat es mostren en una grandària adequada aquelles figures que no es poden apreciar correctament dins el treball

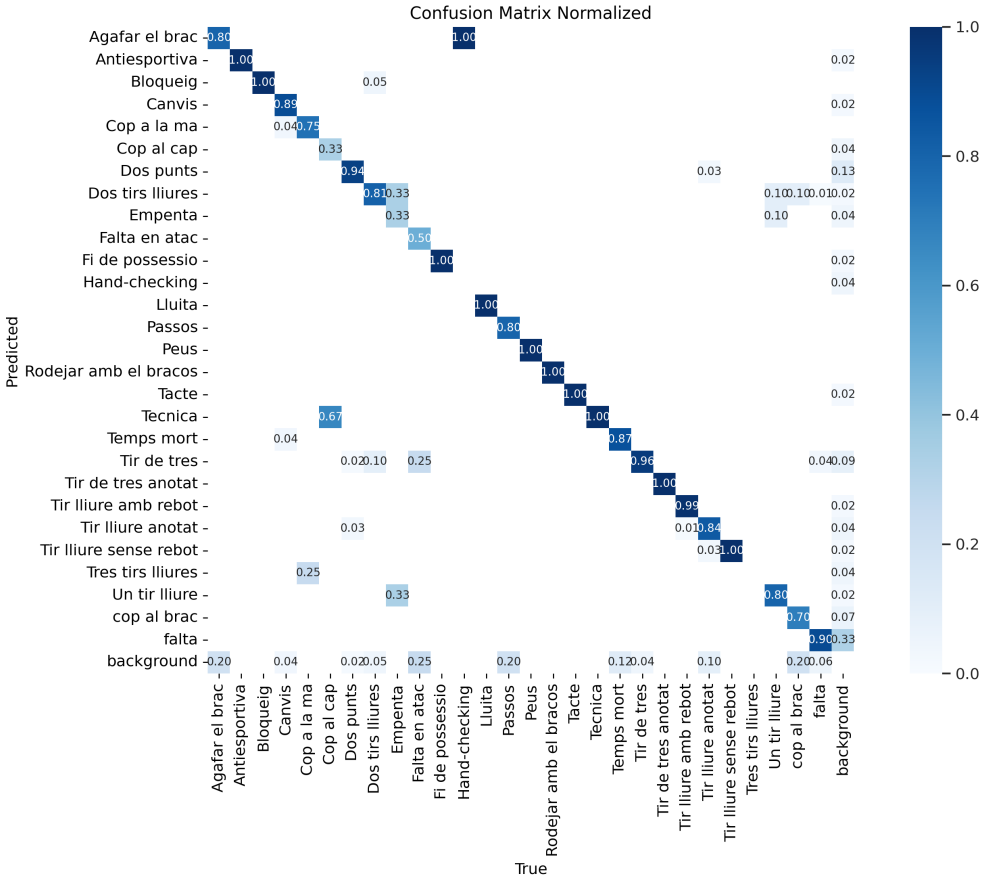


Figura 6.1: CM Norm entrenament (escenari 1)

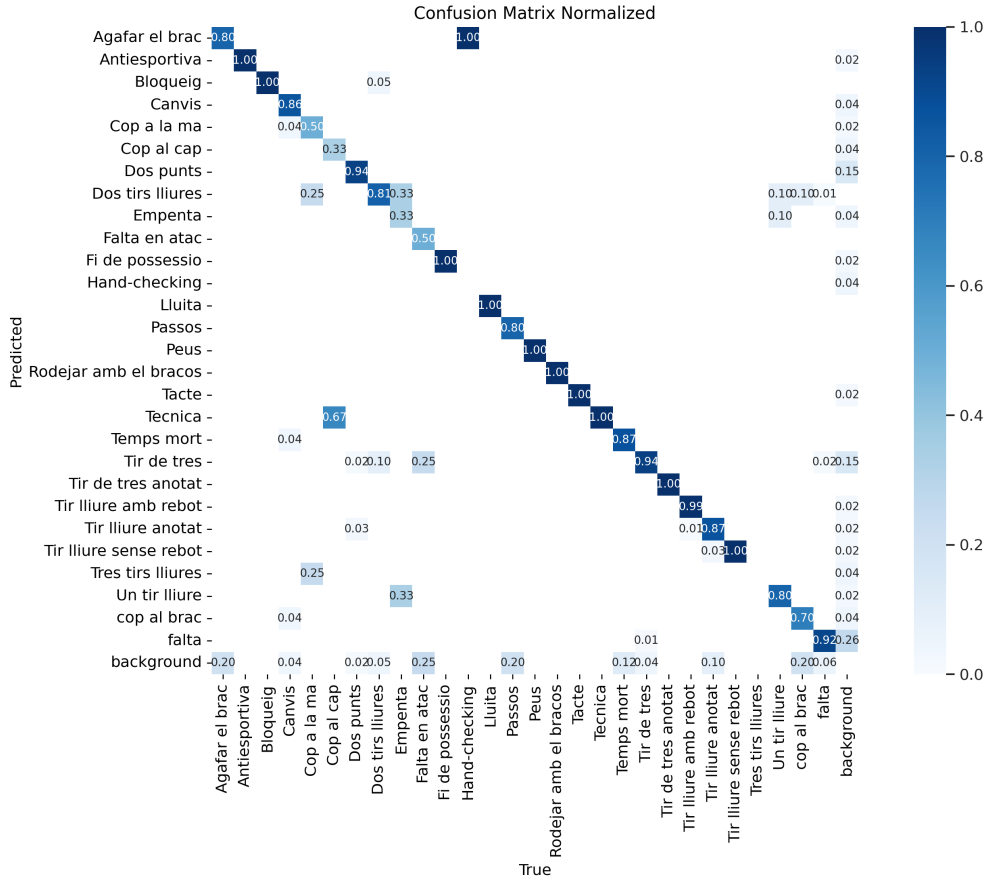


Figura 6.2: CM Norm validació (escenari 1)

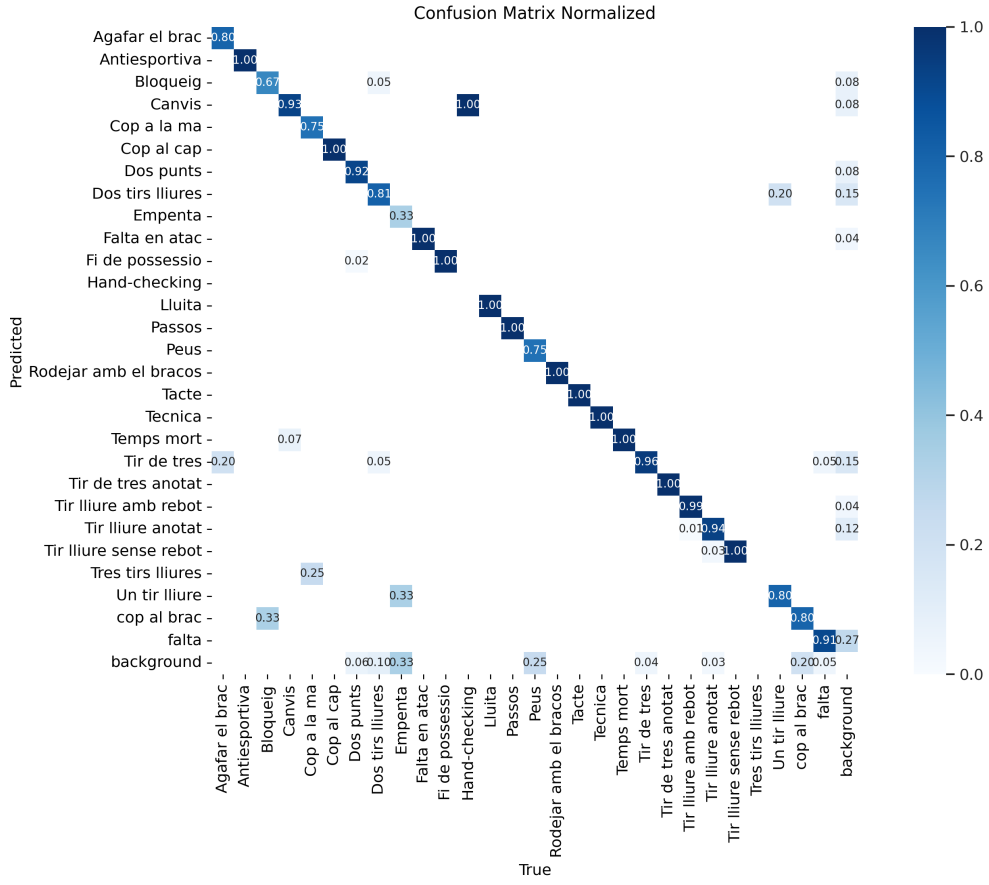


Figura 6.3: CM Norm entrenament (escenari 2)

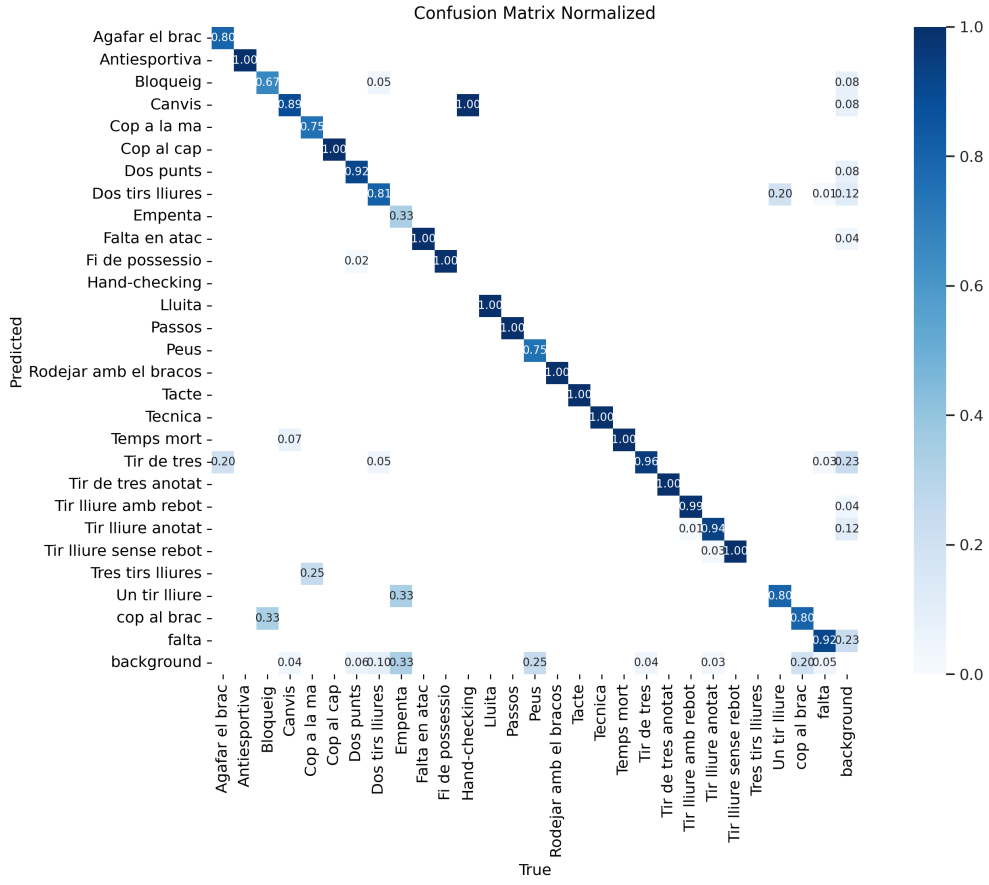


Figura 6.4: CM Norm validació (escenari 2)

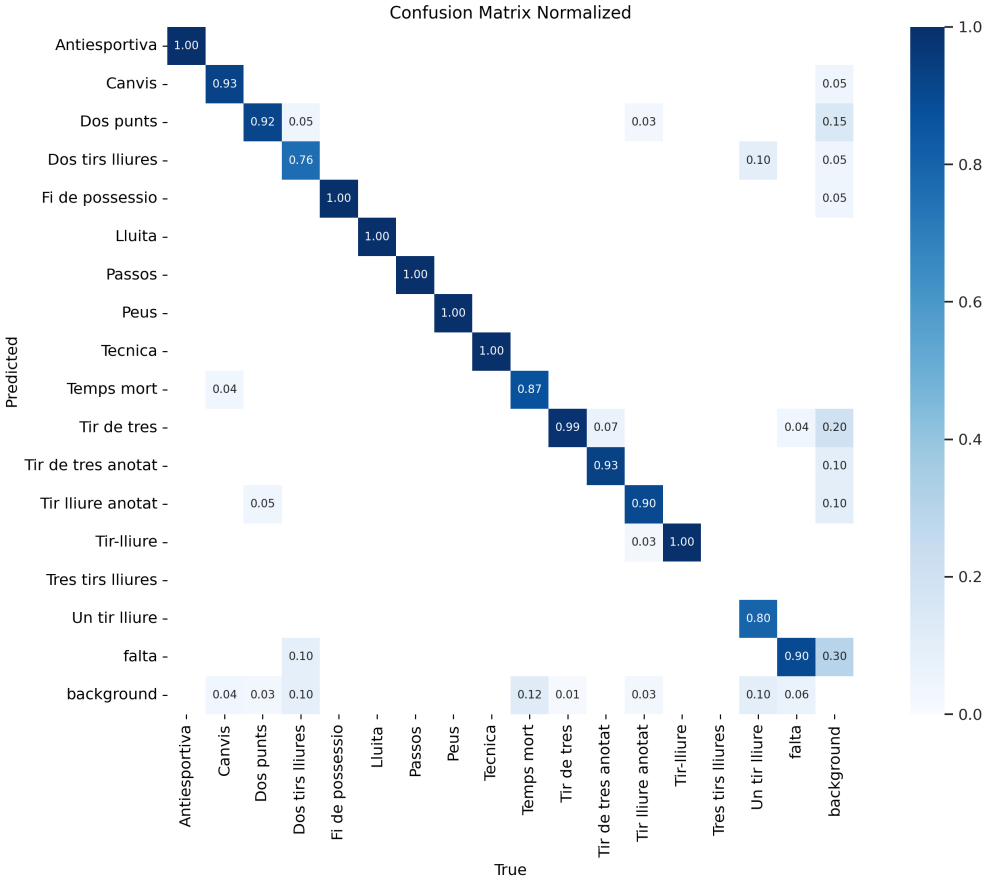


Figura 6.5: CM Norm entrenament (escenari 3)

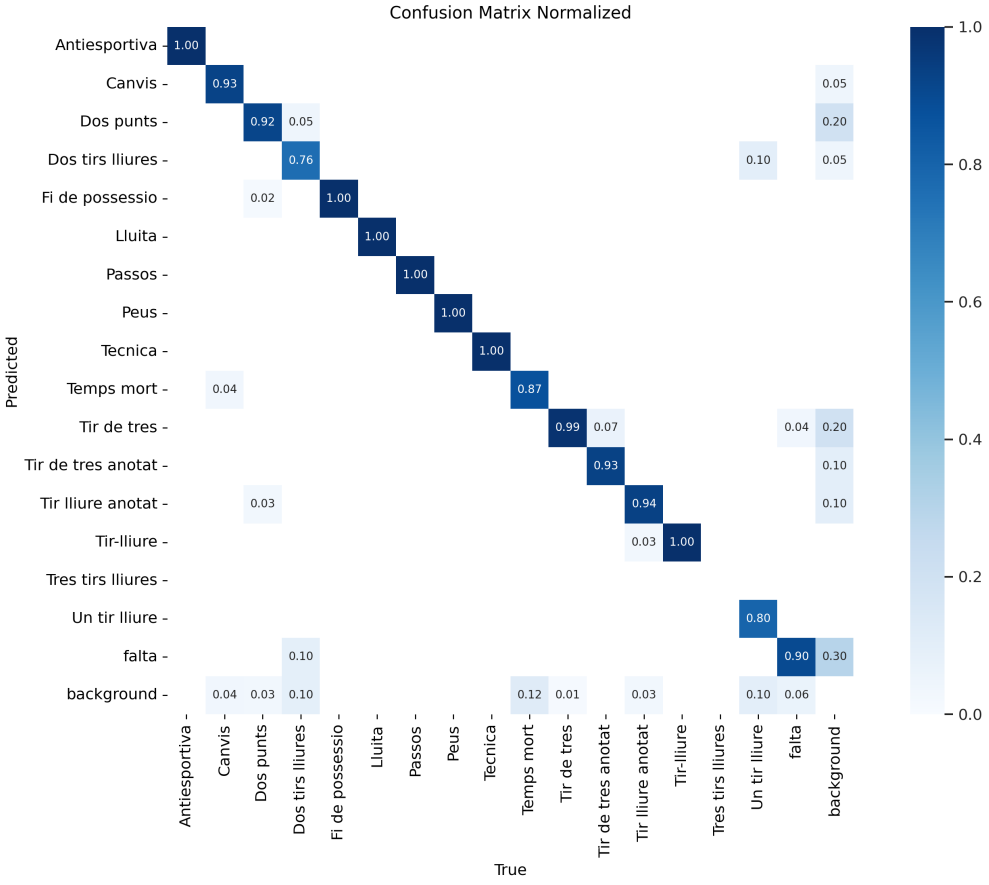


Figura 6.6: CM Norm validació (escenari 3)

Bibliografia

[1] Pedro Almagro Blanco. Algoritmo de retropropagación.

[2] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. 323:533–536.

[3] Carlos Santana (DotCSV). ¿qué es el descenso del gradiente? algoritmo de inteligencia artificial | dotcsv.

[4] Carlos Santana (DotCSV). ¿qué es una red neuronal? parte 1 : La neurona | dotcsv.

[5] Carlos Santana (DotCSV). ¿qué es una red neuronal? parte 2 : La red | dotcsv.

[6] Carlos Santana (DotCSV). ¿qué es una red neuronal? parte 3 : Backpropagation | dotcsv.

[7] Carlos Santana (DotCSV). ¿qué es una red neuronal? parte 3.5 : Las matemáticas de backpropagation | dotcsv.

[8] Elastic. ¿ qué son las incrustaciones de palabras ?

[9] Glenn Jocher, Burhan-Q, and Rizwan Munawar. Model validation with ultralytics YOLO.

[10] Glenn Jocher, Fatih C. Akyon, Laughing, Dzmitry Plashchynski, Burhan, Ayush Chaurasia, Muhammad Rizwan Munawar, and Jason Sohn. Ultralytics configuration.

[11] Juan Du. Understanding of object detection based on CNN family and YOLO. 1004:012029.

[12] Juan Pedro. Detailed explanation of YOLOv8 architecture.

[13] Xiang Li, Wenhai Wang, Lijun Wu, Shuo Chen, Xiaolin Hu, Jun Li, Jinhui Tang, and Lian Yang. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. 33:21002–21012.

[14] Gaurav Mohan. Developing a basketball minimap for player tracking using broadcast data and applied homography.

[15] Juan Monroy, Adriana Ramírez, Roberto Alejo, and Erika López. Aspectos relevantes para mejorar el desempeño del algoritmo backpropagation.

[16] Raúl Rojas. *Neural Networks: A Systematic Introduction*. The Backpropagation Algorithm. Springer Berlin Heidelberg.

[17] Paula Rooney. The NBA's digital transformation is a game-changer.

[18] Devin Schumacher. CSPDarknet53.

[19] scikit learn. ConfusionMatrixDisplay.

[20] Valentyn Sickhar. Confusion matrix for image classification.

[21] Miguel Strefezza Bianco and Yasuhiko Dote. ALGORITMO DE APRENDIZAJE ACELERADO PARA BACKPROPAGATION.

[22] Diwan Tausif, G. Anirudh, and Jitendra V. Tembhurne. Object detection using YOLO: challenges, architectural successors, datasets and applications.

[23] Juan Terven, Diana-Margarita Córdova-Esparza, and Julio-Alejandro Romero-González. A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS.

[24] Jane Torres. YOLOv8 architecture: A deep dive into its architecture.

[25] Zheng, Zhaohui and Wang, Ping and Liu, Wei and Li, Jinze and Ye, Rongguang and Ren, Dongwei. Distance-IoU loss: Faster and better learning for bounding box regression. 34:12993–13000.