UNIVERSITAT DE BARCELONA
Facultat de Matemàtiques i Informàtica

DOBLE GRAU DE MATEMÀTIQUES I ENGINYERIA INFORMÀTICA

Treball final de grau

TRACKING DE JUGADORS EN IMATGES AMB TRANSFORMERS

Autor: Enric Calvo Ventura

Directora Informàtica:	Dra. Petia Ivanova Radeva
Director Matemàtiques:	Dr. Joan Carles Tatjer Montaña
Realitzat a:	Departament de Matemàtiques i Informàtica
Barcelona,	20 de juny de 2021

Abstract

Data is becoming increasingly important across the board, and so is within sports disciplines. Particularly, in football. For instance, the position of the different players on the pitch is in itself data which has been proved useful for applications such as injury prevention, player scouting and so on. This type of data, designated "tracking data", is obtained at the time of this study using GPS technology on a professional level. Since local leagues and football institutions are responsible for the management of this data, it can prove to be difficult for outside actors to obtain access to it. In reality, this means this data ends up being accessible to high income and top league clubs and institutions only. For this very reason, the need to find alternative ways to generate and obtain such data arises. This project focuses its scope on using computer vision as an alternative to the previously stated.

The aim of this work therefore, is to beforehand acquire a theoretical view and understanding in the branch of machine learning and convolutional neural networks and their application to detect people in football videos. In particular we will explore the use of Transformers, an architecture of convolutional neural networks that appeared very recently and involved a paradigm shift in state of the art models to process the source material and generate the raw data. With a specialized dataset of exclusively football images, we have been able to train a DETR model and compare its results with other existent models as a reference. With the results in hand, we have explored ways of improve such models.

We have obtained a trained model that successfully manages to detect the players on a football match with the caveat of it being dependent on the source material's quality. While it does succeed in most regular game play, it struggles in situations where the source material presents occlusions and accumulations of players in the image (for example during a corner kick, where many players can accumulate near the goal). We have been able to even slightly improve the results of the trained model by managing double detections, but the size of our dataset has proved to be a constraint in this direction. Finally, we have discussed some possible future lines of improvement to achieve better results, such as increasing our dataset or using a wider range of frames to reduce the margin of error when players are occluded.

Resum

Les dades estan agafant cada vegada més importància en el món de l'esport i, en particular, en el del futbol. Per exemple, la posició que ocupen els jugadors en el terreny de joc és una informació útil per a tasques de prevenció de lesions, scouting de jugadors, etc. Aquest tipus de dada, anomenada "tracking data", s'obté a nivell professional mitjançant dispositius GPS. Les lligues locals són les encarregades de gestionar aquestes dades, fet que dificulta obtenir-les d'altres països. A més, la forma en que s'obtenen aquestes dades resulta molt costosa i només a l'abast dels clubs professionals. Per aquest motiu sorgeix la necessitat d'obtenir-les amb mètodes alternatius. Aquest treball estudia la visió per computador, com a alternativa a l'esmentat anteriorment.

L'objectiu d'aquest treball és adquirir una base teòrica en la branca del machine learning i les xarxes neuronals convolucionals per a poder estudiar el problema de detecció de jugadors de futbol en imatges usant Transformers, una arquitectura que va aparèixer el 2020 i va suposar un canvi de paradigma dels models de l'estat de l'art. Amb un dataset especialitzat d'imatges de futbol, hem entrenat el model DETR i hem comparat els resultats amb altres models com Yolo. Després d'analitzar els resultats, hem buscat formes de millorar-los.

Hem obtingut un model entrenat que aconsegueix detectar amb èxit els jugadors d'un partit de futbol quan rep imatges d'una certa qualitat, però que no és capaç de detectar amb prou precisió els jugadors en casos d'oclusions i acumulacions (com és l'exemple d'un servei de córner, en que es produeixen concentracions de molts jugadors en poc espai). Hem pogut millorar lleugerament els resultats del model entrenat afegint un threshold per evitar dobles deteccions, però ens hem vist limitats pel tamany del nostre dataset. Finalment hem comentat algunes línies futures a seguir per a obtenir uns millors resultats, com és l'obtenció d'un dataset més extens, o l'utilització dels frames anteriors i posteriors al que s'està usant per a detectar jugadors oclosos.

2020 Mathematics Subject Classification. 65F, 65K, 68T, 68T05, 68T10

Agraïments

Vull agrair als meus pares i a la Júlia el suport que m'han donat durant la realització d'aquest treball. També a l'Ignasi, per tota l'ajuda que m'ha facilitat.

A més, vull donar les gràcies als meus tutors. A la Petia, per haver-me guiat en tot moment i haver-me exigit el màxim. Al Joan Carles, per la seva disposició i colaboració durant aquests mesos.

Finalment, voldria donar les gràcies al Pau, pels primers mesos de treball en que tot començava, i al Marcos, per la seva disposició i la seva gran ajuda, que m'ha permès arribar fins al final en aquest projecte.

Índex

1 Introducció

El projecte

Les dades cada cop juguen un paper més important en el món de l'esport. En particular, saber on es troben en cada moment els jugadors en un partit de futbol pot ajudar als clubs en múltiples aspectes, com prevenir lesions, analitzar els rivals, buscar jugadors, etc. De les diferents formes d'obtenir i analitzar aquestes dades, ens interessa la visió artificial. I últimament, quan estem parlant de visió artificial no podem obviar el fet que són les xarxes neuronals que permeten arribar el més lluny.

En aquest treball ens intoduirem en el món de les xarxes neuronals convolucionals. Intentarem explicar-ne el funcionament bàsic d'una manera entenedora per a un lector no especialitzat en machine learning. Finalment provarem a aplicar el model DETR per al problema de detecció i classificació de jugadors i pilota en un partit de futbol, tot comparant-lo amb altres mètodes actuals.

Per a aconseguir el nostre objectiu, necessitarem obtenir un dataset d'imatges de partits de futbol que utilitzarem per a entrenar el model. A més, estudiarem el funcionament del DETR i n'analitzarem els resultats obtinguts.

Motivació

Les dades que s'obtenen en els partits de futbol es poden classificar en 2 tipus: Les dades d'events són aquelles que fan referència a les accions que duen a terme els jugadors (xutar, passar la pilota, etc), mentre que les dades de tracking aporten la posició dels futbolistes en el pla XY (camp de futbol) en tot moment.

Els clubs professionals utiltzen dispositius GPS per a obtenir les coordenades dels futbolistes. Aquest mètode, dotat d'alta precisió, comporta elevats costos econòmics. A més, les empreses encarregades d'obtenir aquestes dades solen estar controlades per les lligues de cada país, fet que en dificulta l'accés a clubs d'altres regions.

Per aquest motiu sorgeix l'interès de realitzar aquesta tasca mitjançant visió artificial. Així, aquest treball pretén estudiar la viabilitat d'aquest mètode pel que fa a la seva precisió. Abans de començar aquest treball, els meus coneixements sobre xarxes neuronals i machine learning eren molt limitats. Per aquest motiu, una de les grans motivacions del treball era entendre'n els conceptes clau per a poder encarar projectes com aquest. Així, els objectius del treball són:

- Adquirir una base teòrica sòlida en la branca del machine/de learning i les xarxes neuronals
- Aprendre a aplicar un model de xarxes neuronals: a entrenar un model existent usant un dataset propi i a visualitzar l'entrenament i comparar-lo amb altres mètodes
- Obtenir un model que detecti els jugadors i la pilota en un partit de futbol i saber avaluar-ne el rendiment.
- Anar més allà analitzant si podem estudiar les matrius dels nuclis i aplicar la descomposició SVD per optimitzar els càlculs.

Estructura de la Memòria

L'estructura de la memòria consistirà de les següents parts diferenciades: En la primera part analitzarem l'estat de l'art, començant pel cas particular de la detecció de jugadors de futbol i seguint amb els resultats dels Transformers i del DETR.

Seguidament farem una introducció teòrica, tot explicant com funcionen els mètodes que utilitzarem. Aquesta serà una part extensa ja que un dels principals objectius del treball era endinsar-se en el món del machine learning i les xarxes convolucionals, degut a que és una branca amb molta rellevància que no es tracta molt durant la carrera. Així, en aquesta introducció teòrica s'explicarà en què consisteixen els principals elements de les xarxes neuronals convolucionals, tot posant exemples per a que un lector no especialitzat en aquesta branca de la informàtica pugui entendre com funcionen.

Un cop sentades les bases de les xarxes neuronals, explicarem en què consisteixen els Transformers, per a finalment poder veure l'estructura i el funcionament del DETR. A més, donarem una petita introducció sobre la descomposició en valors singulars d'una matriu, procés que utilitzarem per a l'experimentació i l'optimització del DETR.

A continuació explicarem tot el procés que hem seguit per a fer la part pràctica del treball, explicant com i perquè hem fet cada pas. Finalment, presentarem els resultats de tot allò que hem fet durant el treball. Aquests seran analitzats i comparats per a així acabar la memòria amb la conclusió obtinguda, l'anàlisi de possibles línies d'investigació futures i una reflexió personal.

2 Estat de l'art

2.1 El problema de la detecció automàtica de jugadors en vídeos esportius

La detecció de jugadors en un partit de futbol mitjançant imatges és un problema molt relacionat amb la detecció de persones. En canvi, és molt difícil trobar informació de l'estat actual de l'art en aquest aspecte. El motiu és que principalment és un problema en el que hi treballen empreses privades que no fan públiques les seves metodologies. Actualment existeix tecnologia com per, mitjançant una gran quantitat de càmeres disposades al voltant dels camps de futbol, dur a terme una detecció de tots els futbolistes del terreny de joc amb una alta precisió. En aquest treball, volem estudiar si a partir d'imatges de televisió es poden obtenir dades de qualitat. Un exemple d'empresa que es dedica a això es SkillCorner [16]. Sense saber quins algorismes utilitzen, aquesta empresa és capaç de detectar amb èxit els jugadors i la pilota d'un partit de futbol com podem veure a la figura 1.

Figura 1: Imatge obtinguda de la web de Skillcorner on es mostra la detecció dels jugadors d'un partit de futbol.

2.2 Detecció d'objectes

El problema de detecció d'objectes, en canvi, és un problema general amb moltes aportacions públiques i mètriques molt més accessibles. Com a exemple, està el dataset COCO2, un dataset públic amb més de 200.000 imatges anotades amb fins a 80 classes d'objectes. Com podem observar a la figura 2, cada cop surten arquitectures amb millors resultats en aquest dataset.

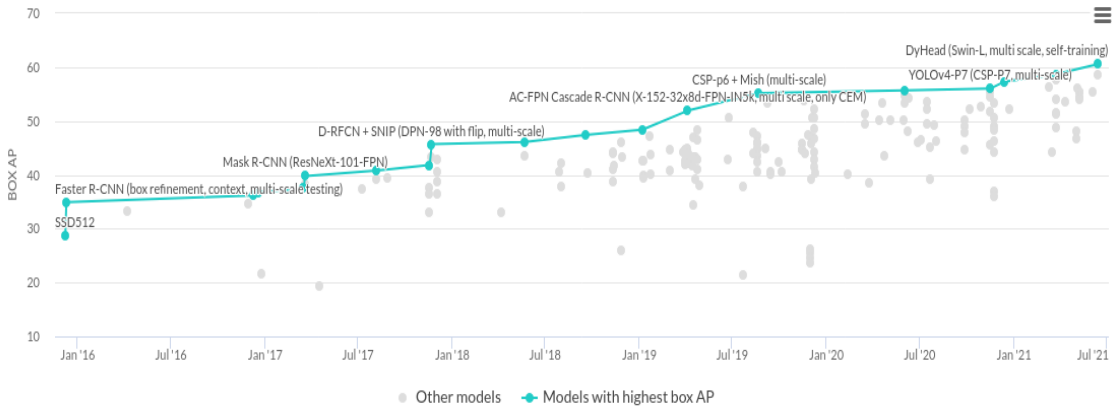


Figura 2: Imatge obtinguda de [17]. Veiem el rànking amb millors resultats en mAP en object detection en el dataset de COCO

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Figura 3: Imatge obtinguda de l'article original de Transformers [1] on veiem el rendiment respecte a l'estat de l'art del moment en el problema de traducció.

2.3 Transformers

Els transformers són una arquitectura que va sortir el 2017 en l'article "Attention is all you need" [1] basada en el mecanisme d'attention que suposava un gran canvi en relació al que s'estava fent aleshores en el problema de traducció de frases. En la figura 3 podem veure com aconseguia millorar l'estat de l'art.

D'altra banda, DETR va ser una arquitectura que va aparèixer el 2020 en l'article "End-to-End Object Detection with Transformers" [5] que per primera vegada usava l'estructura del transformer original per al problema de detecció i classificació d'objectes en imatges. Com podem veure a la figura 4 assolia l'estat de l'art anteriorment dominat per models seqüencials (RNNs, LSTMs, etc) i establia una baseline per a millorar-ne el rendiment.

Model	GFLOPS/FPS	#params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster RCNN-DC5	320/16	166M	39.0	60.5	42.3	21.4	43.5	52.5
Faster RCNN-FPN	180/26	42M	40.2	61.0	43.8	24.2	43.5	52.0
Faster RCNN-R101-FPN	246/20	60M	42.0	62.5	45.9	25.2	45.6	54.6
Faster RCNN-DC5+	320/16	166M	41.1	61.4	44.3	22.9	45.9	55.0
Faster RCNN-FPN+	180/26	42M	42.0	62.1	45.5	26.6	45.4	53.4
Faster RCNN-R101-FPN+	246/20	60M	44.0	63.9	**47.8**	**27.2**	48.1	56.0
DETR	86/28	41M	42.0	62.4	44.2	20.5	45.8	61.1
DETR-DC5	187/12	41M	43.3	63.1	45.9	22.5	47.3	61.1
DETR-R101	152/20	60M	43.5	63.8	46.4	21.9	48.0	61.8
DETR-DC5-R101	253/10	60M	**44.9**	**64.7**	47.7	23.7	**49.5**	**62.3**

Figura 4: Imatge obtinguda de l'article original de DETR [5] on veiem el rendiment respecte a l'estat de l'art del moment en la detecció d'objectes en el dataset de COCO.

3 Introducció teòrica

En aquesta secció del treball començarem introduïnt les nocions bàsiques de xarxa neuronal. S'explicaran els conceptes clau d'aquestes amb la finalitat de poder entendre com funcionen els Transformers, i en última instància, el DETR.

Pràcticament tota la teoria que s'explicarà a continuació no es dóna durant la carrera, motiu pel qual vaig passar els primers mesos del treball preparant aquesta base teòrica per a poder passar finalment a la part pràctica del treball amb uns fonaments sòlids pel que fa a les xarxes neuronals.

3.1 Xarxes neuronals

Les xarxes neuronals són models computacionals que s'inspiren en el funcionament d'un cervell humà. La idea consisteix en un conjunt de neurones artificials interconnectades. Cada connexió entre neurones simula la sinapsis, procés pel qual una neurona transmet una senyal elèctrica a una altre neurona.

Per a visualitzar bé en què consisteix, anem a veure una xarxa neuronal molt senzilla com a exemple. La funció de la xarxa serà decidir si, donada una imatge de 3×3 pixels, aquesta representa la lletra "H", la "I", la "J"o la "K".

Anem a definir una imatge. Per al cas que ens ocupa, pensarem una imatge d'un sol canal. Podem veure una imatge de dues formes: com una matriu, o com un vector.

Definició 3.1. *Anomenarem (matriu) imatge a una matriu $I \in \mathcal{M}_{h\times w}(\mathbb{R})$, on h és el nombre de files i w el nombre de columnes.*

Enlloc de pensar una imatge com una matriu, mitjançant una simple concatenació de les seves files, podem veure una imatge com un vector. Així, definim *vector imatge* a partir de la definició en forma matricial, de la següent forma:

Definició 3.2. *Sigui $I \in \mathcal{M}_{h\times w}(\mathbb{R})$ una imatge. Anomenarem vector imatge al vector $x \in \mathbb{R}^{h\cdot w}$, format per la concatenació de les files de I de la següent forma:*

$$x = (I_i)_{1\leq i\leq h} = \begin{pmatrix} I_1 & I_2 & I_3 & \cdots & I_h \end{pmatrix}$$

on I_i és la fila i-èssima de la imatge I.

Exemple 3.3. Donada una imatge de 3×3 pixels que representa la lletra "H", veiem a continuació la matriu imatge associada I, seguida del vector imatge x:

Imatge 3×3

$$\iff \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \iff \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

I: Matriu imatge

x: Vector imatge

Notació. *Utilitzant un abús de notació, a partir d'ara anomenarem imatge a la seva forma matricial o forma vectorial, segons la necessitat per treballar-hi.*

A continuació definirem una funció Indmax, la matriu de pesos W, el vector biaix b, i el conjunt de classes C, amb els quals formarem una xarxa neuronal d'una sola capa.

Definició 3.4. *Anomenarem matriu de pesos a una matriu $W \in \mathcal{M}_{n\times(h\cdot w)}(\mathbb{R})$, on n és el nombre de files i $h \cdot w$ el nombre de columnes.*

Definició 3.5. *Anomenarem vector biaix a un vector $b \in \mathbb{R}^n$.*

Definició 3.6. *Anomenarem conjunt de classes $C = \{c_1, c_2, \cdots, c_n\}$ al conjunt de paraules de sortida d'una xarxa neuronal.*

Definició 3.7. *Sigui $v \in \mathbb{R}^n$ un vector. Denotarem per* $\text{Indmax}(v)$ *la posició de l'element amb valor màxim de v, és a dir:*

$$\text{Indmax}(v) = \min\{i \in \{1, \ldots, n\} \mid v_i = \max(v)\}$$

Exemple 3.8. Sigui $v = (1, 3, 5, 2)$. Aleshores, $\text{Indmax}(v) = 3$.

Observació 3.9. En el cas que ens ocupa, h i w correspondran al nombre de files i columnes, respectivament, de la imatge que passarem per la xarxa neuronal i n serà el nombre de classes que pot detectar el nostre model (4 en el nostre exemple).

Definició 3.10. *Sigui $x \in \mathbb{R}^{h\cdot w}$ un vector imatge, $W \in \mathcal{M}_{n\times(h\cdot w)}(\mathbb{R})$ una matriu de pesos, $b \in \mathbb{R}^n$ un vector biaix i C un conjunt de classes. Anomenarem xarxa neuronal d'una sola capa a una funció f de la forma:*

$$\begin{aligned} f_{W,b}\colon \mathbb{R}^{h\cdot w} &\to C \\ x &\mapsto f_{W,b}(x) = c_i \end{aligned}$$

on $i = \text{Indmax}(W \cdot x^T + b)$ i c_i és l'element i-èssim del conjunt de classes C.

La xarxa f que hem definit classificarà una imatge segons un conjunt de classes C. Anem a veure a continuació un exemple de com funcionaria aquesta senzilla xarxa d'una sola capa.

Exemple 3.11. Siguin W una matriu de pesos, C un conjunt de classes i b un vector biaix tals que:

$$W = \begin{pmatrix} 1 & -3 & 0.2 & 1.5 & 0.8 & 1.6 & 0.7 & -1.4 & 0.1 \\ 0.9 & 2.9 & 0.3 & -1.6 & 0.9 & -1.4 & 0.6 & 1.5 & 0.2 \\ -1.2 & -2.7 & 0.4 & -1.4 & -0.7 & 1.8 & -0.7 & 1.6 & 0.4 \\ 1.1 & -2.5 & 0.4 & 1.6 & 1 & -1.4 & 0.8 & -1.2 & 0.3 \end{pmatrix};$$

$$b = \begin{pmatrix} 1 & 0 & -0.5 & 0.3 \end{pmatrix}; \qquad C = \{H, I, J, K\};$$

Denotem per $f_{W,b}$ la xarxa neuronal formada per la matriu de pesos W i el vector biaix b que, donat un vector imatge $x \in \mathbb{R}^{3\cdot3}$ associat a una imatge I, determina la classe d'aquesta d'entre els elements de C. Donada la imatge següent:

Imatge "h" 3×3

$$\iff \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

x_h: Vector imatge

Anem a passar-la per la nostra xarxa neuronal d'una sola capa per a veure'n els resultats. El primer pas és multiplicar la matriu de pesos W pel vector imatge x_h^T:

$$W \cdot x_h^T = \begin{pmatrix} 1 & -3 & 0.2 & 1.5 & 0.8 & 1.6 & 0.7 & -1.4 & 0.1 \\ 0.9 & 2.9 & 0.3 & -1.6 & 0.9 & -1.4 & 0.6 & 1.5 & 0.2 \\ -1.2 & -2.7 & 0.4 & -1.4 & -0.7 & 1.8 & -0.7 & 1.6 & 0.4 \\ 1.1 & -2.5 & 0.4 & 1.6 & 1 & -1.4 & 0.8 & -1.2 & 0.3 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 5.9 \\ -0.1 \\ -1.4 \\ 3.8 \end{pmatrix}$$

Un cop feta la multiplicació matricial, sumem el vector biaix al resultat. Posteriorment aplicarem la funció Indmax per a finalment obtenir la classe de la imatge.

$$W \cdot x_h^T + b = \begin{pmatrix} 5.9 \\ -0.1 \\ -1.4 \\ 3.8 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ -0.5 \\ 0.3 \end{pmatrix} = \begin{pmatrix} 6.9 \\ -0.1 \\ -1.9 \\ 4.1 \end{pmatrix} \implies$$

$$\implies i = \text{Indmax}(W \cdot x_h^T + b) = \text{Indmax}(\begin{pmatrix} 6.9 & -0.1 & -1.9 & 4.1 \end{pmatrix}) = 1 \implies$$

$$\implies f_{W,b}(x_h) = c_1 = H$$

En aquest cas, doncs, la xarxa neuronal ha identificat correctament la imatge com la lletra "H".

Observació 3.12. El vector resultant de l'operació $W \cdot x_h^T + b$ ens està donant informació sobre com de segura està la xarxa de que la imatge sigui d'una classe o d'una altra. Com més alt sigui el valor del element i-èssim del vector esmentat, més "confiança" té la xarxa que l'imatge és de classe c_i, i viceversa. Més endavant farem referència al vector esmentat i el definirem formalment. Abans, però, veiem un parell més d'exemples:

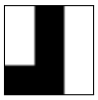
Exemple 3.13. Utilitzarem la xarxa $f_{W,b}$ de l'exemple anterior en dues imatges més. Veiem que la imatge $x_{j'}$ és igual que la x_j però desplaçada 1 pixel cap a l'esquerra.

Imatge "j" 3×3

$$\iff \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

x_j: Vector imatge

$$\text{Indmax}(W \cdot x_j^T + b) = \text{Indmax}(\begin{pmatrix} 1.5 & 0.6 & 3.7 & -1.6 \end{pmatrix}) = 3 \implies f_{W,b}(x_j) = c_3 = J$$



Imatge " j' " 3×3

$$\iff \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

$x_{j'}$: Vector imatge

$$\text{Indmax}(W \cdot x_{j'}^T + b) = \text{Indmax}(\begin{pmatrix} -1.9 & 5.9 & -3 & -1.6 \end{pmatrix}) = 2 \implies f_{W,b}(x_{j'}) = c_2 = I$$

Podem comprobar que la xarxa ha reconegut correctament la imatge x_j, però no la imatge modificada $x_{j'}$. És possible que al ser un model de xarxa tant senzill no puguem aspirar a obtenir uns grans resultats.

Si volem augmentar la complexitat de la nostra xarxa, és raonable pensar en afegir una nova matriu de pesos. Així, una forma d'implementar-ho seria aquesta: Si anomenem la nostra xarxa d'una sola capa $f_{W_1,b}$, afegint una segona matriu de pesos W_2 ens quedaria $f_{W_1,W_2,b}$ tal que:

$$\begin{aligned} f_{W_1,W_2,b}\colon \mathbb{R}^{h\cdot w} &\to C \\ x &\mapsto f_{W,b}(x) = c_i \end{aligned}$$

on $i = \text{Indmax}(W_1 \cdot W_2 \cdot x^T + b)$ i c_i és l'element i-èssim del conjunt de classes C. És immediat veure que $W_1 \cdot W_2 = W_3$, és a dir, que no hem afegit profunditat a la xarxa. La forma amb la que se soluciona aquest fet és mitjançant el que s'anomenen funcions d'activació. En donarem una definició intuitiva.

Definició 3.14. *Una funció d'activació en una xarxa neuronal és una funció que donat un nombre x, l'activa (el deixa passar a la següent capa) segons el seu valor. Està inspirada en el funcionament de les neurones del cervell biològic.*

Observació 3.15. Podem considerar l'acció d'una funció d'activació sobre un vector $v = (v_1, v_2, ..., v_n)$ com el vector resultant d'aplicar la funció d'activació sobre cada element de v.

Exemple 3.16. La funció ReLU és la funció d'activació definida de la següent forma:

$$\begin{aligned} ReLU\colon \mathbb{R} &\to \mathbb{R}^+ \\ x &\mapsto ReLU(x) = \max(0, x) \end{aligned}$$

Veiem com en aquest cas, els valors negatius no passaran a la següent capa de la xarxa ja que no es consideren rellevants.

Exemple 3.17. Un altre exemple de funció d'activació és la Sigmoid:

$$\begin{aligned} Sigmoid\colon \mathbb{R} &\to \mathbb{R}^+ \\ x &\mapsto Sigmoid(x) = \frac{1}{1+e^{-x}} \end{aligned}$$

Així doncs, una xarxa neuronal estarà formada per la concatenació de matrius de pesos W_i, vectors biaix b_i i funcions d'activació ϕ. Anem a definir formalment una xarxa de dues capes per a posteriorment veure'n un exemple.

Definició 3.18. *Sigui $W = \{W_1, W_2\}$, amb $W_1 \in \mathcal{M}_{n_1\times w}(\mathbb{R})$, $W_2 \in \mathcal{M}_{n_2\times n_1}(\mathbb{R})$ matrius de pesos. Sigui $b = \{b_1, b_2\}$, amb $b_1 \in \mathbb{R}^n_1$, $b_2 \in \mathbb{R}^n_2$ vectors biaix, i $C = \{c_1, c_2, ..., c_{n_2}\}$ un conjunt de classes. Sigui $x \in \mathbb{R}^{h\times w}$ un vector imatge i ϕ una funció d'activació. Anomenarem xarxa neuronal de 2 capes a la funció:*

$$\begin{aligned} f_{W,b,\phi}\colon \mathbb{R}^{h\cdot w} &\to C \\ x &\mapsto f_{W,b,\phi}(x) = c_i \end{aligned}$$

on:

$$i = \text{Indmax}(W_2 \cdot \phi(W_1 \cdot x^T + b_1) + b_2)$$

i c_i és l'element i-èssim del conjunt de classes C.

Exemple 3.19. Siguin $W = \{W_1, W_2\}$ matrius de pesos, $b = \{b_1, b_2\}$ vectors biaix tals que :

$$W_1 = \begin{pmatrix} 1 & -2 & 3 & -1 \\ 0 & 2 & -3 & 1 \\ 3 & 3 & 1 & -4 \end{pmatrix}; \quad W_2 = \begin{pmatrix} -1 & 3 & -1 \\ 3 & -1 & -2 \end{pmatrix}; \quad b_1 = \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix}; \quad b_2 = \begin{pmatrix} 2 \\ -3 \end{pmatrix};$$

Sigui $C = \{I, L\}$ conjunt de classes. Considerem la funció d'activació ReLU de l'exemple 3.15. Donada la imatge següent, amb el seu corresponent vector imatge:

Imatge "L" 2×2

$$\iff \begin{pmatrix} 1 & 0 & 1 & 1 \end{pmatrix}$$

x: Vector imatge

Anem a passar-la per la xarxa neuronal de 2 capes $f_{W,b,ReLU}$:

$$W_1 \cdot x^T + b_1 = \begin{pmatrix} 1 & -2 & 3 & -1 \\ 0 & 2 & -3 & 1 \\ 3 & 3 & 1 & -4 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 5 \\ -2 \\ 1 \end{pmatrix} \implies$$

$$\implies \text{ReLU}(W_1 \cdot x^T + b_1) = \text{ReLU} \begin{pmatrix} 5 \\ -2 \\ 1 \end{pmatrix} = \begin{pmatrix} 5 \\ 0 \\ 1 \end{pmatrix} \implies$$

$$\implies W_2 \cdot \text{ReLU}(W_1 \cdot x^T + b_1) + b_2 = \begin{pmatrix} -1 & 3 & -1 \\ 3 & -1 & -2 \end{pmatrix} \cdot \begin{pmatrix} 5 \\ 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 2 \\ -3 \end{pmatrix} = \begin{pmatrix} -4 \\ 10 \end{pmatrix} \implies$$

$$\implies \text{Indmax}(W_2 \cdot \text{ReLU}(W_1 \cdot x^T + b_1) + b_2) = \text{Indmax} \begin{pmatrix} -4 \\ 10 \end{pmatrix} = 2 \implies$$

$$\implies f_{W,b,ReLU}(x) = c_2 = L$$

Observació 3.20. De l'exemple anterior se'n pot deduir inductivament l'estructura d'una xarxa neuronal de n capes. Les xarxes que segueixen aquesta estructura s'anomenen FFNN (Feed Forward Neural Network).

Notació. *D'ara en endavant, quan parlem de xarxa neuronal, ens referirem a una xarxa neuronal d'una sola capa sempre i quan no s'especifiqui el contrari. Això serà degut a que buscarem un model més simple per a explicar les diferents parts teòriques.*

Un cop hem vist tant l'estructura com el funcionament d'una xarxa neuronal, cal determinar una manera d'avaluar la precisió d'aquesta (en el nostre exemple, la precisió al classificar una imatge). D'aquesta manera, tindrem un criteri per a poder modificar els paràmetres de la xarxa per a així millorar-ne el rendiment.

És així com sorgeix la funció de Loss. Tot seguit la introduïrem i en donarem un parell d'exemples.

3.1.1 Loss

La funció de Loss (o pèrdua) és l'encarregada d'avaluar la qualitat de les prediccions que fa una xarxa neuronal. La funció ens retornarà un valor alt quan la xarxa faci una mala predicció, i viceversa. Per a definir-la formalment, necessitarem abans el concepte de vector predicció.

Definició 3.21. *Siguin $f_{W,b}$ una xarxa neuronal, x un vector imatge i C un conjunt de classes. Anomenarem vector predicció de x respecte del conjunt de classes C al vector v que representa quant s'assembla la imatge x a cada classe del conjunt C:*

$$v := W \cdot x^T + b$$

Definició 3.22. *Siguin x un vector imatge de classe $c_x \in C$, amb C conjunt de n classes i v un vector predicció de x respecte de C. Anomenarem funció de Loss a una funció L de la forma*

$$\begin{aligned} L\colon \mathbb{R}^n \times C &\to \mathbb{R} \\ (v, c_x) &\mapsto L(v, c_x) \end{aligned}$$

que quantifica l'error del vector predicció.

Observació 3.23. Podem considerar la Loss d'un conjunt d'imatges com la suma de les Loss de cada imatge del conjunt.

Exemple 3.24. Veiem a continuació una funció de Loss senzilla com a exemple: L'anomenada Hinge loss (SVM). La denotarem com L_{H_l}:

$$\begin{aligned} L_{H_l}\colon \mathbb{R}^n \times C &\to \mathbb{R}^+ \\ (v, c_x) &\mapsto L(v, c_x) = \sum_{i|c_i \neq c_x} \max(0, v_i - v_x + 1) \end{aligned}$$

on v_x és l'element del vector predicció v corresponent a la classe c_x, i v_i és l'element i-èssim de v. Prenent els valors de l'exemple 3.11, tenim:

$$v = \begin{pmatrix} 6.9 & -0.1 & -1.9 & 4.1 \end{pmatrix}; \quad c_x = H; \quad v_x = 6.9;$$

$$L_{H_l}(v, c_x) = L_{H_l}(\begin{pmatrix} 6.9 & -0.1 & -1.9 & 4.1 \end{pmatrix}, H) = \sum_{i|c_i \neq H} \max(0, v_i - 6.9 + 1) =$$

$$= \max(0, -0.1 - 6.9 + 1) + \max(0, -1.9 - 6.9 + 1) + \max(0, 4.1 - 6.9 + 1) =$$

$$= \max(0, -6) + \max(0, -7.8) + \max(0, -1.8) = 0$$

En aquest exemple, la Hinge loss pren el valor 0, fet que ens diu que la xarxa ha funcionat de manera òptima. En canvi, prenent els valors del segon cas de l'exemple 3.13, tenim:

$$v = \begin{pmatrix} -1.9 & 5.9 & -3 & -1.6 \end{pmatrix}; \quad c_x = J; \quad v_x = -3;$$

$$L_{H_l}(v, c_x) = L_{H_l}(\begin{pmatrix} -1.9 & 5.9 & -3 & -1.6 \end{pmatrix}, J) = \sum_{i|c_i \neq J} \max(0, v_i - (-3) + 1) =$$

$$= \max(0, -1.9 + 3 + 1) + \max(0, 5.9 + 3 + 1) + \max(0, -1.6 + 3 + 1) =$$

$$= \max(0, 2.1) + \max(0, 9.9) + \max(0, 2.4) = 14.4$$

En aquest cas, hem obtingut una Hinge loss de 14.4, fet que ens mostra que la xarxa ha funcionat de manera errònia i caldria actualitzar-ne els valors per a millorar-ne el resultat.

Observació 3.25. A la pràctica, se suma a la funció de Loss una funció R de regularització per a simplificar el model, fet que provocarà un millor rendiment en imatges de test (imatges no usades per a l'entrenament del model).

3.1.2 Optimització

Per a que una xarxa neuronal funcioni correctament, hem de ser capaços de trobar els paràmetres que minimitzin la funció de Loss. En el cas que ens ocupa, donada una matriu de pesos W, i un vector biaix b, voldrem actualitzar-ne els valors per a que les prediccions que faci la xarxa siguin tant precises com sigui possible.

La funció de Loss depèn de la imatge x que s'avalua, del conjunt de classes C en que la volem classificar, i de la matriu de pesos W i el vector biaix b. Tenint en compte que el nostre objectiu és, fixat un conjunt de classes, optimitzar el funcionament de la xarxa per a qualsevol imatge, és lògic pensar que hem de buscar el mínim de la funció de Loss en funció dels paràmetres W i b.

Hi ha diferents maneres d'actualizar els valors de W i b en relació a la funció de Loss. Els algorismes encarregats d'aquesta tasca s'anomenen optimitzadors. En el nostre cas, utilitzarem com a exemple el descens de gradient estocàstic.

Definició 3.26. *Donada una funció de Loss L associada a una xarxa neuronal, SGD o descens de gradient estocàstic és un mètode iteratiu que busca el mínim de L en funció del gradient d'aquesta respecte d'un paràmetre W amb la fórmula:*

$$W(t) = W(t-1) - \alpha \cdot (\frac{\partial L}{\partial W});$$

on t és la iteració, i α un paràmetre que defineix la mida de les passes de cada iteració anomenat learning rate.

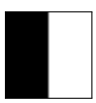
Calcular el gradient de la funció de Loss per a cada paràmetre amb la fórmula no serà viable, ja que a la pràctica les xarxes neuronals tindran moltes capes. La solució a aquest problema és el mètode recursiu anomenat backpropagation.

3.1.3 Backpropagation

Aquest mètode recursiu fa ús de la regla de la cadena per a calcular els gradients de tots els paràmetres al llarg del graf computacional. A continuació entrarem en més detall per a veure en què consisteix. Esquemàticament, backpropagation consisteix en 3 fases:

- Avaluar la funció de Loss L
- Calcular el gradient de L respecte tots els elements de W i b
- Modificar els elements de W i b proporcionalment amb el seu gradient

Prenem la xarxa de dues capes $f_{W,b,\phi}$ de l'exemple 3.19. Anomenarem $Z_i = W_i \cdot A_i + b_i$, amb $A_0 = x$ i $A_i = \phi(Z_i)$. L'estructura de la xarxa és de la forma:


Imatge x

$$\xrightarrow{W_1 \cdot A_0} \begin{pmatrix} 2 \\ -2 \\ -3 \end{pmatrix} \xrightarrow{+b_1} \underset{Z_1}{\begin{pmatrix} 4 \\ -2 \\ -2 \end{pmatrix}} \xrightarrow{\phi(Z_1)} \underset{A_1}{\begin{pmatrix} 4 \\ 0 \\ 0 \end{pmatrix}} \xrightarrow{W_2 \cdot A_1} \begin{pmatrix} -4 \\ 12 \end{pmatrix} \xrightarrow{+b_2} \underset{Z_2}{\begin{pmatrix} -2 \\ 9 \end{pmatrix}} \xrightarrow{L_{H_l}(Z_2, I)} 12$$

El nostre objectiu és calcular el gradient de L_{H_l} en funció de W_1, W_2, b_1, b_2. Per la regla de la cadena, tenim:

$$\frac{\partial L_{H_l}}{\partial W_2} = \frac{\partial L_{H_l}}{\partial Z_2} \cdot \frac{\partial Z_2}{\partial W_2}; \qquad \frac{\partial L_{H_l}}{\partial b_2} = \frac{\partial L_{H_l}}{\partial Z_2} \cdot \frac{\partial Z_2}{\partial b_2};$$

$$\frac{\partial L_{H_l}}{\partial W_1} = \frac{\partial L_{H_l}}{\partial Z_2} \cdot \frac{\partial Z_2}{\partial A_1} \cdot \frac{\partial A_1}{\partial Z_1} \cdot \frac{\partial Z_1}{\partial W_1}; \qquad \frac{\partial L_{H_l}}{\partial b_1} = \frac{\partial L_{H_l}}{\partial Z_2} \cdot \frac{\partial Z_2}{\partial A_1} \cdot \frac{\partial A_1}{\partial Z_1} \cdot \frac{\partial Z_1}{\partial b_1}; \tag{3.1}$$

Aquí és on sorgeix el primer inconvenient. Estem intentant derivar funcions de Loss i d'activació, però hem vist prèviament que no són necessàriament derivables. A la pràctica, aquestes funcions seran derivables $\forall x \in \mathbb{R} \setminus C$, on C serà un conjunt de punts finit. Per exemple, la funció d'activació ReLU no és derivable en $x = 0$. En aquests casos, es defineix el valor de la derivada "artificialment" com 0, 0.5 o 1, per exemple. Cal destacar que a la pràctica aquesta mena de valors no s'assoliran pràcticament mai.

Sortejat l'obstacle que presenta la derivabilitat en tots els punts de les funcions usades, podem definir la forma en que s'actualitzaran els valors de W_i i b_i amb el mètode de SGD. Denotarem per $W_i(t)$ i $b_i(t)$ els valors de W_i i b_i, respectivament, en el pas t de backpropagation. Així, ens queda:

$$W_i(t) = W_i(t-1) - \alpha \cdot (\frac{\partial L_{H_l}}{\partial W_i}); \qquad b_i(t) = b_i(t-1) - \alpha \cdot (\frac{\partial L_{H_l}}{\partial b_i})$$

Anem a veure un exemple de càlcul d'una iteració del mètode backpropagation amb SGD per a la xarxa.

Exemple 3.27. Considerem la xarxa $f_{W,b,\phi}$ amb els valors següents:


Imatge "i" 2×2

$$\iff \underset{x\text{: Vector imatge}}{\begin{pmatrix}1 & 0 & 1 & 0\end{pmatrix}}; \qquad \alpha = 1; \qquad \phi = ReLU; \qquad C = \{I, L\}$$

$$W_1 = \begin{pmatrix}1 & -2 & 3 & -1\\ 0 & 2 & -3 & 1\\ 3 & 3 & 1 & -4\end{pmatrix}; \quad W_2 = \begin{pmatrix}-1 & 3 & -1\\ 3 & -1 & -2\end{pmatrix}; \quad b_1 = \begin{pmatrix}2\\0\\1\end{pmatrix}; \quad b_2 = \begin{pmatrix}2\\-3\end{pmatrix};$$

$$Z_1 = W_1 \cdot x^T + b_1 = \begin{pmatrix}4\\-2\\-3\end{pmatrix}; \quad A_1 = \phi(Z_1) = \begin{pmatrix}4\\0\\0\end{pmatrix}; \quad Z_2 = W_1 \cdot A_1 + b_2 = \begin{pmatrix}-2\\9\end{pmatrix};$$

Sigui $v = Z_2$ el vector predicció. Calculem el valor de pèrdua al classificar l'imatge x amb la funció SVM de l'exemple 3.24:

$$L_{H_l}(v, c_x) = \sum_{i|c_i \neq c_x} \max(0, v_i - v_x + 1); \qquad v_x = -2; \qquad c_x = I;$$

$$L_{H_l}(v, I) = \sum_{i|c_i \neq I} \max(0, v_i - (-2) + 1) = \max(0, 9 - (-2) + 1) = 9 + 2 + 1 = 12$$

$$Z_i = \begin{pmatrix}z_1^i\\z_2^i\end{pmatrix}; \qquad L_{H_l}(Z_i, I) = z_2^i - z_1^i + 1$$

Prosseguim a calcular una iteració de backpropagation amb l'objectiu de millorar el resultat de la xarxa. Volem calcular, doncs:

$$W_i(t) = W_i(t-1) - \alpha \cdot (\frac{\partial L_{H_l}}{\partial W_i}); \qquad b_i(t) = b_i(t-1) - \alpha \cdot (\frac{\partial L_{H_l}}{\partial b_i});$$

per $i = \{1, 2\}$. Així, necessitem trobar el valor dels gradients. Recordem que en el cas de la funció ReLU, podem considerar la seva derivada excepte en el punt zero. Tenim:

$$\frac{\partial L_{H_l}}{\partial Z_2} = \begin{pmatrix} -1 & 1 \end{pmatrix}; \qquad \frac{\partial Z_2}{\partial W_2} = A_1 = \begin{pmatrix} 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 \end{pmatrix}; \qquad \frac{\partial Z_2}{\partial b_2} = Id;$$

$$\frac{\partial A_1}{\partial Z_1} = Id; \qquad \frac{\partial Z_2}{\partial A_1} = \begin{pmatrix} -1 & 3 & -1 \\ 3 & -1 & -2 \end{pmatrix}; \qquad \frac{\partial Z_1}{\partial b_1} = Id;$$

$$\frac{\partial Z_1}{\partial W_1} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix};$$

De (3.1), obtenim:

$$\frac{\partial L_{H_l}}{\partial W_2} = \begin{pmatrix} -4 & 0 & 0 \\ 4 & 0 & 0 \end{pmatrix}; \qquad \frac{\partial L_{H_l}}{\partial b_2} = \begin{pmatrix} -1 \\ 1 \end{pmatrix};$$

$$\frac{\partial L_{H_l}}{\partial W_1} = \begin{pmatrix} 4 & 0 & 4 & 0 \\ -4 & 0 & -4 & 0 \\ -1 & 0 & -1 & 0 \end{pmatrix}; \qquad \frac{\partial L_{H_l}}{\partial b_1} = \begin{pmatrix} 4 \\ -4 \\ -1 \end{pmatrix};$$

Per tant, les noves W_i i b_i seran:

$$W_1(t) = \begin{pmatrix} 1 & -2 & 3 & -1 \\ 0 & 2 & -3 & 1 \\ 3 & 3 & 1 & -4 \end{pmatrix} - \begin{pmatrix} 4 & 0 & 4 & 0 \\ -4 & 0 & -4 & 0 \\ -1 & 0 & -1 & 0 \end{pmatrix} = \begin{pmatrix} -3 & -2 & -1 & -1 \\ 4 & 2 & 1 & 1 \\ 4 & 3 & 2 & -4 \end{pmatrix};$$

$$W_2(t) = \begin{pmatrix} -1 & 3 & -1 \\ 3 & -1 & -2 \end{pmatrix} - \begin{pmatrix} -4 & 0 & 0 \\ 4 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 3 & 3 & -1 \\ -1 & -1 & -2 \end{pmatrix};$$

$$b_1(t) = \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix} - \begin{pmatrix} 4 \\ -4 \\ -1 \end{pmatrix} = \begin{pmatrix} -2 \\ 5 \\ 1 \end{pmatrix}; \qquad b_2(t) = \begin{pmatrix} 2 \\ -3 \end{pmatrix} - \begin{pmatrix} -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 3 \\ -4 \end{pmatrix};$$

Amb tot això, ja hem actualitzat tots els valors de la xarxa $f_{W,b,\phi}$. Anem a comprovar ara si els nous valors ens donen un millor resultat al avaluar l'imatge de la lletra "i".


Imatge x

$$\xrightarrow{W_1 \cdot A_0} \begin{pmatrix} -4 \\ 5 \\ 6 \end{pmatrix} \xrightarrow{+b_1} \underset{Z_1}{\begin{pmatrix} -6 \\ 10 \\ 7 \end{pmatrix}} \xrightarrow{\phi(Z_1)} \underset{A_1}{\begin{pmatrix} 0 \\ 10 \\ 7 \end{pmatrix}} \xrightarrow{W_2 \cdot A_1} \begin{pmatrix} 23 \\ -24 \end{pmatrix} \xrightarrow{+b_2} \underset{Z_2}{\begin{pmatrix} 26 \\ -28 \end{pmatrix}} \xrightarrow{L(Z_2, I)} 0$$

Com podem veure, la xarxa ha après a identificar l'imatge gràcies al algorisme SGD. Cal mencionar que si volem executar l'algorisme per a cada imatge que tinguem en el set d'imatges d'entrenament, pot resultar molt costós. Quan es té un dataset molt gran, el que es sol fer és calcular la Loss d'un subconjunt de n imatges i optimitzar aleshores.

Aquests petits subconjunts d'imatges es diuen batches. Així, si tenim un dataset amb 100 imatges, si usem una mida de batch de 10, haurem de fer els càlculs 10 vegades enlloc de 100.

3.2 CNN

Si bé les xarxes neuronals com les que hem introduït resulten molt útils per a certes tasques, tenen certes mancances quan tractem amb imatges. És per aquest motiu que sorgeixen les xarxes neuronals convolucionals (o CNN). A continuació explicarem en què consisteixen de manera senzilla, tot definint alguns conceptes bàsics.

Recordem que en les xarxes neuronals senzilles com les que hem vist abans, partíem d'una imatge. Aquesta sol ser representada per 3 canals (RGB) tot i que per a simplificar l'explicació hem suposat que les imatges que teníem eren de només un canal. Així doncs, podem dir que el que fan les xarxes neuronals és, intuïtivament, relacionar els píxels d'una imatge.

El que es pretén amb les xarxes convolucionals és afegir un nivell més de complexitat/profunditat en el sentit que, enlloc de relacionar els píxels d'un canal d'una imatge, primer obtindrem múltiples canals d'aquesta, de manera que tinguem més informació per a relacionar. Com s'obtenen aquests canals extra? Mitjançant les convolucions. Anem a definir que són. Usarem abans unes definicions prèvies:

Definició 3.28. *Anomenarem imatge de n canals (de dimensió $h \times w$) al tensor format per n matrius imatge de mida $h \times w$. La denotarem per:*

$$I^{n\times(h\times w)} = (I_1, I_2, ..., I_n)$$

amb $I_i \in \mathcal{M}_{h\times w}(\mathbb{R})$ de la forma

$$I_i = \begin{pmatrix} x^i_{11} & x^i_{12} & \dots \\ \vdots & \ddots & \\ x^i_{h1} & & x^i_{hw} \end{pmatrix}, \quad i \in \{1, 2, ..., n\}$$

Definició 3.29. *Anomenarem filtre o kernel a un tensor amb estructura idèntica a una imatge de n canals. Un filtre K associat a una imatge de n canals (de mida $h \times w$) serà de la forma:*

$$K^{n\times(h_2\times w_2)} = (K_1, K_2, ..., K_n)$$

amb

$$\begin{pmatrix} k^i_{11} & k^i_{12} & \dots \\ \vdots & \ddots & \\ k^i_{h_2 1} & & k^i_{h_2 w_2} \end{pmatrix}, \quad i \in \{1, 2, ..., n\}, \quad h_2 < h, w_2 < w;$$

Veiem doncs com un filtre sempre tindrà la mateixa profunditat que la imatge on es vol aplicar. Però que vol dir aplicar un filtre? Aquí entra la definició de convolució d'una imatge:

3.2.1 Convolució

Definició 3.30. *Anomenarem convolució a l'operació entre una imatge (de n canals) $I^{n\times(h_1\times w_1)}$ i un filtre associat $K^{n\times(h_2\times w_2)}$ que denotarem per $\otimes$, depenent de s, p, definida com:*

$$I^{n\times((h_1+2\cdot p)\times(w_1+2\cdot p))} \otimes K^{n\times(h_2\times w_2)} = \hat{I}^{n\times(h_3\times w_3)}$$

amb

$$h_3 = \frac{h_1 + 2\cdot p - h_2}{s} + 1; \qquad w_3 = \frac{w_1 + 2\cdot p - w_2}{s} + 1;$$

El resultat del producte queda

$$\hat{I}^{n\times(h_3\times w_3)} = (\hat{I}_1, \hat{I}_2, ...\hat{I}_n); \quad \hat{I}_q = \begin{pmatrix} a^q_{11} & a^q_{12} & \dots \\ \vdots & \ddots & \\ a^q_{h_3 1} & & a^q_{h_3 w_3} \end{pmatrix}; \quad q \in \{1, 2, ..., n\}$$

amb els elements a_{ij} *de* $\hat{I}_q$ *definits com*

$$a^q_{ij} = \sum_{\alpha=1}^{h_2} \sum_{\beta=1}^{w_2} x^q_{\hat{i}\hat{j}} \cdot k^q_{\alpha\beta}$$

amb

$$\hat{i} = (i-1)\cdot s + \alpha; \quad \hat{j} = (j-1)\cdot s + \beta;$$

Els paràmetres s *i* p *s'anomenen stride i padding, respectivament. El padding consisteix en augmentar la dimensió de les matrius del tensor* I *"rodejant-les" amb files i columnes de valors arbitraris (per exemple, se solen utilitzar zeros) de manera que quedi com* $I^{n\times((h_1+2\cdot p)\times(w_1+2\cdot p))}$*. El valor de* s *(o stride), controla com es mou el filtre o kernel sobre la imatge on volem convolucionar. Els valors de* h_2, w_2, s, p *han de complir que* $h_3, w_3 \in \mathbb{Z}$*. El padding s'aplica abans de fer la transformació a partir del stride.*

Anem a veure un exemple de convolució. Per a simplificar els càlculs, considerarem una imatge d'un sol canal. Aplicarem una convolució amb padding 1 ($p = 1$) i stride 2 ($s = 2$).

Exemple 3.31. Siguin $p = 1$, $s = 2$ i siguin $I^{1\times(4\times4)}$ una imatge d'un canal, i $K^{1\times(2\times2)}$ un kernel associat a aquesta tals que:

$$I^{1\times(4\times4)} = \begin{pmatrix} 1 & 1 & 2 & 0 \\ 0 & 1 & 3 & 1 \\ 1 & 1 & 0 & 3 \\ 2 & 1 & 2 & 1 \end{pmatrix} \xrightarrow{padding=1} I^{1\times((4+p)\times(4+p))} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 2 & 0 & 0 \\ 0 & 0 & 1 & 3 & 1 & 0 \\ 0 & 1 & 1 & 0 & 3 & 0 \\ 0 & 2 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix};$$

$$K^{1\times(2\times2)} = \begin{pmatrix} 2 & 5 \\ 0 & 3 \end{pmatrix};$$

Anem a calcular la convolució:

$$a_{11} = x_{11}\cdot k_{11} + x_{12}\cdot k_{12} + x_{21}\cdot k_{21} + x_{22}\cdot k_{22} = 0\cdot 2 + 0\cdot 5 + 0\cdot 0 + 1\cdot 3 = 03$$
$$a_{12} = x_{13}\cdot k_{11} + x_{14}\cdot k_{12} + x_{23}\cdot k_{21} + x_{24}\cdot k_{22} = 0\cdot 2 + 0\cdot 5 + 1\cdot 0 + 2\cdot 3 = 06$$
$$a_{13} = x_{15}\cdot k_{11} + x_{16}\cdot k_{12} + x_{25}\cdot k_{21} + x_{26}\cdot k_{22} = 0\cdot 2 + 0\cdot 5 + 0\cdot 0 + 0\cdot 3 = 00$$
$$a_{21} = x_{31}\cdot k_{11} + x_{32}\cdot k_{12} + x_{41}\cdot k_{21} + x_{42}\cdot k_{22} = 0\cdot 2 + 0\cdot 5 + 0\cdot 0 + 1\cdot 3 = 03$$
$$a_{22} = x_{33}\cdot k_{11} + x_{34}\cdot k_{12} + x_{43}\cdot k_{21} + x_{44}\cdot k_{22} = 1\cdot 2 + 3\cdot 5 + 1\cdot 0 + 0\cdot 3 = 15$$
$$a_{23} = x_{35}\cdot k_{11} + x_{36}\cdot k_{12} + x_{45}\cdot k_{21} + x_{46}\cdot k_{22} = 1\cdot 2 + 0\cdot 5 + 3\cdot 0 + 0\cdot 3 = 02$$
$$a_{31} = x_{51}\cdot k_{11} + x_{52}\cdot k_{12} + x_{61}\cdot k_{21} + x_{62}\cdot k_{22} = 0\cdot 2 + 2\cdot 5 + 0\cdot 0 + 0\cdot 3 = 10$$
$$a_{32} = x_{53}\cdot k_{11} + x_{54}\cdot k_{12} + x_{63}\cdot k_{21} + x_{64}\cdot k_{22} = 1\cdot 2 + 2\cdot 5 + 0\cdot 0 + 0\cdot 3 = 12$$
$$a_{33} = x_{55}\cdot k_{11} + x_{56}\cdot k_{12} + x_{65}\cdot k_{21} + x_{66}\cdot k_{22} = 1\cdot 2 + 0\cdot 5 + 0\cdot 0 + 0\cdot 3 = 02$$

$$I^{(6\times6)} \otimes K^{(2\times2)} = \hat{I}^{n\times(3\times3)} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 3 & 6 & 0 \\ 3 & 15 & 2 \\ 10 & 12 & 2 \end{pmatrix}$$

3.2.2 Pooling

Un cop vista l'operació de convolució, només ens falta explicar què és una funció de pooling i ja tindrem les eines necessàries per a construir una xarxa convolucional senzilla.

Definició 3.32. *Anomenarem funció de pooling a una funció que, donada unua matriu $M \in \mathcal{M}_{h\times w}(\mathbb{R})$, en disminueix la mida de forma no lineal.*

Exemple 3.33. La funció de pooling més usada s'anomena MaxPooling. Consisteix en subdividir la matriu inicial en rectangles i prendre el valor màxim de cada regió. Donada una imatge de n canals $I^{n\times(h_1\times w_1)}$ el resultat d'aplicar MaxPooling amb un filtre de $n \times (h_2 \times w_2)$, padding p i stride s serà:

$$MaxPool(I^{n\times((h_1+2\cdot p)\times(w_1+2\cdot p))}) = \hat{I}^{n\times(h_3\times w_3)}$$

amb

$$h_3 = \frac{h_1 + 2\cdot p - h_2}{s} + 1; \qquad w_3 = \frac{w_1 + 2\cdot p - w_2}{s} + 1;$$

El resultat del MaxPooling queda

$$\hat{I}^{n\times(h_3\times w_3)} = (\hat{I}_1, \hat{I}_2, ...\hat{I}_n); \quad \hat{I}_p = \begin{pmatrix} a^p_{11} & a^p_{12} & \dots \\ \vdots & \ddots & \\ a^p_{h_3 1} & & a^p_{h_3 w_3} \end{pmatrix}, \quad p \in \{1, 2, ..., n\}$$

amb els elements a_{ij} de $\hat{I}_p$ definits com

$$a^p_{ij} = \max_{\substack{1\le\alpha\le h_2\\ 1\le\beta\le w_2}} x^p_{\hat{i}\hat{j}}$$

amb

$$\hat{i} = (i-1)\cdot s + \alpha; \quad \hat{j} = (j-1)\cdot s + \beta;$$

Per exemple, si tenim una matriu M de la forma:

$$M = \begin{pmatrix} 15 & 2 & 3 & 1 \\ 12 & 2 & 2 & 1 \\ 6 & 2 & 5 & 1 \\ 12 & 3 & 2 & 2 \end{pmatrix}$$

el resultat d'aplicar MaxPooling amb un filtre de $1 \times (2 \times 2)$, padding 1 i stride 2 serà:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 15 & 2 & 3 & 1 & 0 \\ 0 & 12 & 2 & 2 & 1 & 0 \\ 0 & 6 & 2 & 5 & 1 & 0 \\ 0 & 12 & 3 & 2 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \Longrightarrow \begin{pmatrix} 15 & 3 & 1 \\ 12 & 5 & 1 \\ 12 & 3 & 2 \end{pmatrix}$$

Amb tot el que hem comentat, ja podem definir l'estructura d'una CNN (xarxa neuronal convolucional), que dividirem en 3 subestructures més petites. Donada una imatge d'entrada de n canals I_n:

- E, la més petita, consistirà en aplicar una convolució i una funció d'activació: $I_n \xrightarrow{conv} \hat{I_n} \xrightarrow{act} Z_1$
- P, la concatenació de n_1 estructures com l'anterior, seguit d'una capa de pooling: $I_n \xrightarrow{E_1} Z_1 \xrightarrow{E_2} ... \xrightarrow{E_{n_1}} Z_{n_1} \xrightarrow{pool} M_1$
- C, la concatenació de n_2 estructures com l'anterior, seguit d'una xarxa neuronal senzilla (per exemple una $f_{W,b}$): $I_n \xrightarrow{P_1} M_1 \xrightarrow{P_2} ... \xrightarrow{P_{n_2}} M_{n_2} \xrightarrow{f_{W,b}} Out$

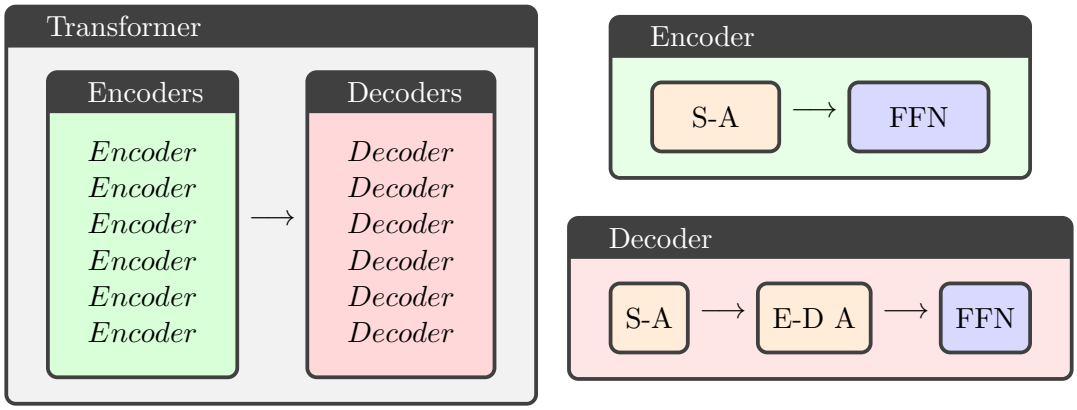
3.3 Transformers

Un cop hem introduït els conceptes que necessitavem sobre les xarxes neuronals convolucionals, ja tenim el material necessari per entendre el funcionament dels Transformers, una arquitectura que va aparèixer el 2017 en l'article *Attention is all you need* [1]. Apareixen per a solucionar d'una vegada per totes els problemes que suposa la computació seqüencial dels models de l'estat de l'art d'aleshores (RNNs, LSTMs, ...). A diferència de tot el que s'estava fent, els Transformers encaraven el problema de la traducció de frases prescindint completament de recurrències i convolucions.

En l'article es presenta un diseny més senzill basat exclusivament en el mecanisme d'Attention, paral·lelitzable i que necessita menys temps per a ser entrenat. A continuació explicarem el seu funcionament, acostant-nos poc a poc, començant a molt alt nivell.

Definició 3.34. *Podem entendre el transformer com una funció que rep una frase en un idioma i retorna la seva traducció a un altre idioma. La qualitat que el diferencia d'altres models de traducció, és el mecanisme de Attention, que veurem a continuació.*

Anem a veure com és l'estructura interna del Transformer. Està format per dues parts diferenciades. Encoder i Decoder, que contenen 6 encoders i 6 decoders, respectivament.



Amb "S-A" per Self-Attention, "FFN" una xarxa neuronal senzilla, i "E-D A" per Encoder-Decoder Attention. Volem veure en que consisteix un bloc de Self-Attention.

Definició 3.35. *Donada una matriu d'entrada $X \in \mathcal{M}_{n_0 \times m_0}(\mathbb{R})$, i 3 matrius de pesos $W_q, W_k, W_v \in \mathcal{M}_{m_0 \times n_1}(\mathbb{R})$, denotarem per Q, K, V (Query, Key, Value), respectivament, a:*

$$Q = X \cdot W_q; \quad K = X \cdot W_k; \quad V = X \cdot W_v;$$

Així, el resultat d'aplicar Self-Attention a X mitjançant W_q, W_k, W_v serà $Z \in \mathcal{M}_{n_0 \times n_1}(\mathbb{R})$ definida com:

$$Z = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{n_1}}\right) \cdot V$$

on softmax és una funció que comprimeix i envia tots els valors a l'interval $[0, 1]$.

Observació 3.36. Realment, en cada bloc de Self-Attention, es calculen 8 Z's diferents, $Z_1, Z_2, ...Z_8$ utilitzant 8 ternes diferents $(W_{q_i}, W_{k_i}, W_{v_i}), i \in \{1, 2, .., 8\}$. Aquest fenomen és conegut com multi-headed attention. Per a obtenir la Z final, es concatenen les Z_i i es multiplica el resultat amb una matriu de pesos extra W_0.

Intuïtivament, podem considerar que el que està fent el mecanisme de Self-Attention és, per cada paraula de la frase que vol traduïr, mirar quanta relació té amb la resta de paraules de la frase.

Però com convertim una frase d'entrada en una matriu X? Convertint cada paraula a un vector mitjançant un algorisme de embedding [2]. A més, a cada vector representant una paraula, li sumarem un vector anomenat postional encoding [3]. Aquest afegit té l'objectiu de guardar informació sobre la localització de cada paraula a la frase. Tenim que cada fila de la matriu Z representa el resultat d'aplicar Self-Attention a una paraula diferent. Per tant, el nombre de files de la matriu correspon al nombre de paraules de la frase a traduïr.

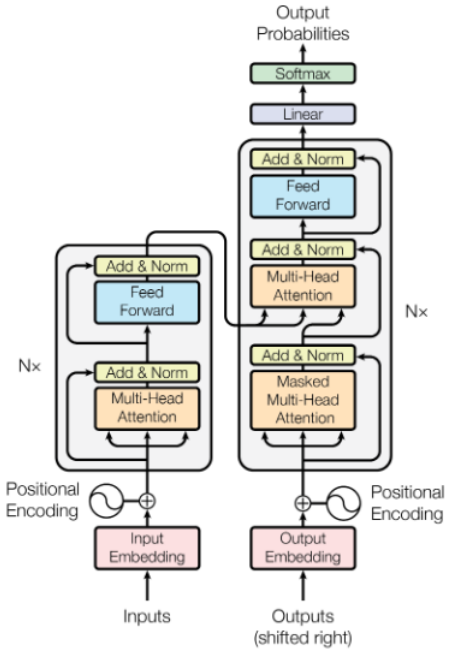
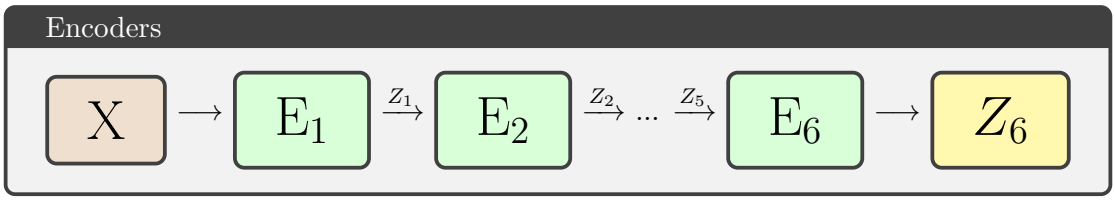


Figura 5: Estructura del transformer obtinguda de l'article [1]

Abans de passar les files de Z paral·lelament per una xarxa neuronal senzilla, es durà a terme una Layer Normalization [4] (normalitzar els gradients) de la suma de l'entrada del bloc de Self-Attention, X, i la sortida Z (Add & Norm). Aquest procés es realitza per conservar més informació de cada paraula. El següent pas consisteix a passar cada fila de Z (paral·lelament) per una xarxa neuronal senzilla. El funcionament d'aquest tipus de xarxes està explicat en apartats anteriors. Els 6 encoders s'apliquen successivament, de manera que la sortida de E_i és l'entrada del E_{i+1}. Així doncs, l'estructura de la part d'encoders del Transformer serà:



Els decoders funcionent pràcticament igual que els encoders. Les úniques diferències són:

- El conjunt de decoders té un funcionament recursiu: El primer decoder comença amb un input nul i l'output de l'últim decoder és la primera paraula traduïda. Aquesta es converteix en l'input del primer decoder de la segona iteració, que retorna com a sortida la segona paraula traduïda. Aquest procés es repeteix fins que es retorna un valor que identifica el final de la frase.
- En el Encoder-Decoder Attention, les matrius V i K provenen del producte de Z_6 per W_{vi}, W_{ki}, respectivament, mentre que la matriu Q és el resultat del producte del output del bloc de self-attention anterior (del mateix decoder) per W_{qi}.
- Per a obtenir una paraula del vector que retorna cada iteració dels decoders, aquest passa per una xarxa neuronal senzilla i s'aplica un softmax final.

Amb això donem per introduït el funcionament del Transformer. Referim el lector a l'article original[1] per a més detalls. En aquest treball volem aplicar l'estructura dels transformers per a detectar i classificar objectes d'una imatge. Aquí és on apareix DETR.

3.4 DETR

Detection Transformer, o DETR, és una arquitectura basada en els Transformers [1] que s'utilitza per a detectar i classificar objectes en imatges. Es va presentar al maig de 2020 en l'article End-to-End Object Detection with Transformers [5]. Encaren el problema de detecció d'objectes com a un problema de predicció de conjunts. El detector d'imatges ens donarà la classe de cada objecte detectat, juntament amb una bounding box (capsa que enquadra l'objecte en l'imatge mitjançant les coordenades de la cantonada superior esquerra, i l'altura i l'amplada de la capsa). A continuació estudiarem l'estructura principal del DETR.

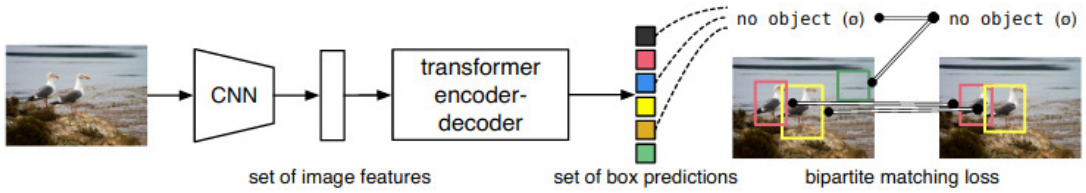


Figura 6: Imatge obtinguda de l'article original [5]. DETR prediu (paral·lelament) el conjunt de deteccions combinant una típica CNN amb l'arquitectura de Transformers. Cada predicció s'emparella amb una bounding box diferent del ground truth (conjunt de bounding boxes correctes). Les prediccions sense match seran de la classse "no object" (∅).

Com podem veure en la figura 6, el funcionament de DETR és el següent:

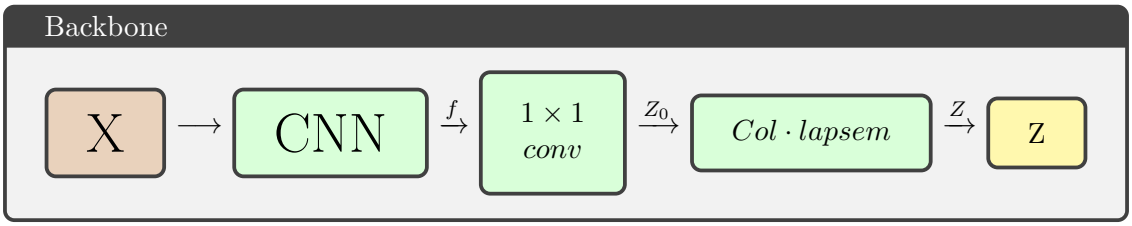
- Passem una imatge per una CNN per a obtenir-ne un conjunt de features (una matriu)
- Aquesta matriu passarà pel transformer (que segueix l'estructura de l'original).
- L'algorisme retorna un conjunt de N prediccions amb el següent format: {classe (probabilitat), bounding box}. N és un hiperparàmetre del model que haurà de ser superior al màxim d'objectes que poden aparèixer en les imatges.

Explicarem a continuació amb una mica més de detall l'estructura de les diferents parts del DETR. Començarem pel backbone, una CNN.

3.4.1 Backbone

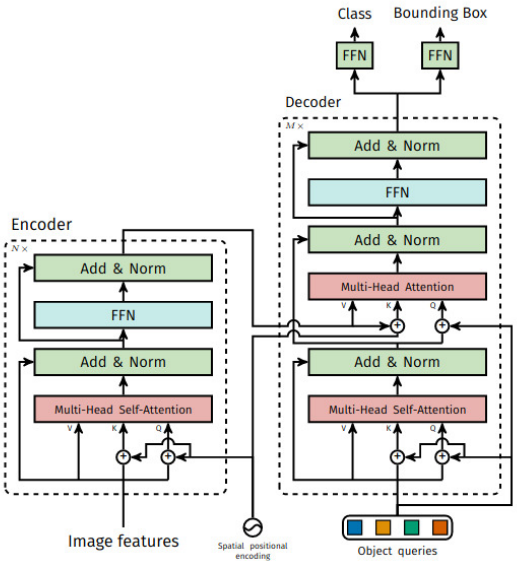
El backbone del DETR és una CNN que rep una imatge de 3 canals, $x \in \mathbb{R}^{3\times H_0\times W_0}$ i retorna una imatge de C canals, $I \in \mathbb{R}^{C\times H\times W}$. Concretament, la CNN és una ResNet50, i els valors que prenen els paràmetres són $C = 2048$, $H = \frac{H_0}{32}$, $W = \frac{W_0}{32}$. A continuació s'aplica una convolució de 1×1 per a disminuir el nombre de canals al valor que necessita el Transformer, d.

Tenim doncs una imatge de d canals, $Z_0 \in \mathbb{R}^{d\times H\times W}$, però l'arquitectura del transformer, com hem comentat abans, està preparada per a rebre matrius. És per aquest motiu que es col·lapsa la dimensió de l'imatge de d canals de manera que ens queda finalment una matriu $Z \in \mathbb{R}^{d\times H\cdot W}$. Esquemàticament, tenim:



3.4.2 Transformer Encoder-Decoder

L'arquitectura del transformer en el DETR segueix la mateixa estructura del transformer original. La única diferència és que en el DETR, el decoder fa les N prediccions paral·lelament. A continuació adjuntem una imatge de la seva estructura obtinguda de l'article original [5]:



A destacar només tenim el significat de les "Object queries". Són un conjunt de N "jutges" cada un dels quals s'especialitza durant l'entrenament en "mirar" a certs sectors de l'imatge, i de buscar objectes d'un cert tamany. Aquesta "especialització" es pot veure gràficament en la figura 7. El valor de N és 100 per defecte.

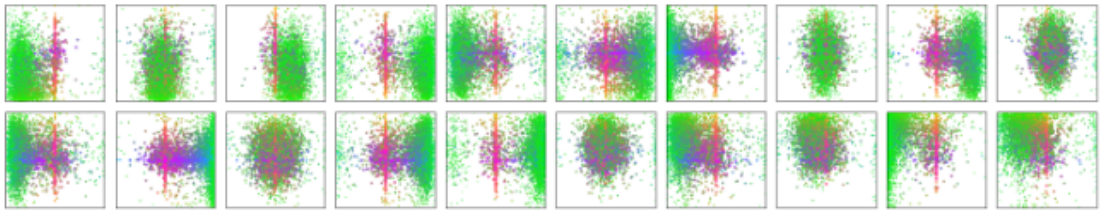


Figura 7: Imatge obtinguda de l'article original [5]. Cada capsa representa el conjunt de prediccions que ha fet una querie diferent (20 en total) sobre el total d'imatges del dataset COCO2017 [17]. Cada predicció és un punt de color. Les coordenades del punt dins cada capsa són les coordenades de la predicció a l'imatge corresponent. El color dels punts indica el tamany de la predicció de bounding box (verd=petites, vermell=amples, blau=altes)

3.4.3 Loss

A continuació explicarem la funció de Loss que utilitza DETR. Donades N prediccions, el primer pas és assignar cada una a una bounding box diferent del ground truth. Veurem com troben el millor emparellament. Abans, però, donarem unes definicions prèvies.

Definició 3.37. *Anomenarem y el conjunt ground truth, i $\hat{y} = \{\hat{y}_i\}_{i=1}^N$ el conjunt de N prediccions. Considerem que el conjunt y té N elements (els objectes de cada imatge + no object detections). Cada element de y és de la forma $y_i = (c_i, b_i)$ on c_i és la classe de l'objecte (pot ser $\emptyset$) i $b_i \in [0,1]^4$ és un vector que definex la bounding box.*

Definició 3.38. *Donada una predicció amb índex $\sigma(i)$, definim la probabilitat de la classe c_i com $\hat{p}_{\sigma(i)}(c_i)$. Definim també la predicció de bounding box com $\hat{b}_{\sigma(i)}$.*

Definició 3.39. *Donades $b_i, \hat{b}_{\sigma(i)}$, definim $\mathcal{L}_{box}$ com:*

$$\mathcal{L}_{box}(b_i, \hat{b}_{\sigma(i)}) = \lambda_{iou}\mathcal{L}_{iou}(b_i, \hat{b}_{\sigma(i)}) + \lambda_{L1}\|b_i - \hat{b}_{\sigma(i)}\|_1$$

on $\lambda_{iou}, \lambda_{L1}$ són hiperparàmetres i $\mathcal{L}_{iou}$ és la IoU (Intersection over Union) loss [7].

Definició 3.40. *Donades $y_i, \hat{y}_{\sigma(i)}$, definim $\mathcal{L}_{match}$ com:*

$$\mathcal{L}_{match}(y_i, \hat{y}_{\sigma(i)}) = -\mathbb{1}_{\{c_i \neq \emptyset\}}\hat{p}_{\sigma(i)}(c_i) + \mathbb{1}_{\{c_i \neq \emptyset\}}\mathcal{L}_{box}(b_i, \hat{b}_{\sigma(i)})$$

on $\mathbb{1}$ és la funció característica.

Amb això, ja podem veure en què consisteix el que ells anomenen bipartite matching. Per a trobar el millor emparellament, busquen la permutació de N elements de $\sigma \in \mathfrak{S}_N$ amb menor cost:

$$\hat{\sigma} = \arg\min_{\sigma \in \mathfrak{S}_N} \sum_i^N \mathcal{L}_{match}(y_i, \hat{y}_{\sigma(i)})$$

Un cop obtinguda la millor permutació $\hat{\sigma}$, utilitzen la Hungarian Loss per a totes les parelles, definida com:

Definició 3.41. *Donat un conjunt de N prediccions y, i un conjunt ground truth $\hat{y}$, definim $\mathcal{L}_{Hungarian}$ com:*

$$\mathcal{L}_{Hungarian}(y, \hat{y}) = \sum_{i=1}^{N} [-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbb{1}_{\{c_i \neq \emptyset\}} \mathcal{L}_{box}(b_i, \hat{b}_{\hat{\sigma}(i)})]$$

És el valor de la Hungarian Loss el que utilitza el seu optimitzador per a entrenar el model. Cal mencionar que DETR utilitza l'optimitzador adaptatiu AdamW [8].

3.5 SVD

Una de les preguntes que ens hem plantejat és com evolucionen les matrius de pesos del model durant l'entrenament, i si en podem explorar aquesta evolució a través de la seva descomposició en valors singulars (SVD) [18]. A continuació donem la definició del mètode SVD.

Definició 3.42. *Sigui $A \in \mathcal{M}_{m\times n}(\mathbb{R})$ una matriu real. Una descomposició en valors singulars de A és una factorització del tipus:*

$$A = U\Sigma V^T$$

amb $U \in \mathcal{M}_{m\times m}(\mathbb{R})$, $V \in \mathcal{M}_{m\times n}(\mathbb{R})$ ortogonals i $\Sigma \in \mathcal{M}_{m\times n}(\mathbb{R})$ matriu diagonal formada amb els valors singulars de A a la diagonal, ordenats de major a menor:

$$\Sigma = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \end{pmatrix}$$

Observació 3.43. Hi ha un teorema d'existència de la descomposició en valors singulars per tota matriu $A \in \mathcal{M}_{m\times n}(\mathbb{R})$. A més, els valors singulars $\lambda_1, \lambda_2, ..., \lambda_n$ són únics, positius, i a més el nombre de valors propis diferents de zero coincideix amb el rang de la matriu A. Es pot veure que els valors singulars diferents de zero de la matriu A són les arrels quadrades dels valors propis de $A^T \cdot A$ i $A \cdot A^T$.

A la part pràctica, aplicarem aquesta descomposició a les matrius de pesos W_q, W_k, W_v del transformer de DETR per a observar-ne els valors singulars, i eliminar-ne un percentatge dels que tenen menor valor per a veure'n la incidència en l'entrenament. Aquest procés el detallarem a la secció d'implementació del treball.

4 Implementació dels Transformers per la Detecció de Jugadors en imatges

4.1 Planificació

Al començar el treball, el meu objectiu principal era poder aplicar un algorisme de visió artificial per a detectar jugadors de futbol en imatges (fixem-nos que des del principi el projecte es va plantejar el problema com de detecció en imatges i no com tracking en vídeos). Partia, però, d'un coneixement molt limitat en quant a machine learning, xarxes neuronals, etc, ja que durant la carrera només s'hi dóna una petita ullada. Aquest fet va marcar la planificació del treball.

Així doncs, vam acordar amb la tutora d'informàtica, a principis d'octubre, que necessitaria un parell de mesos per a obtenir tota la teoria necessària per a poder desenvolupar el treball. Durant aquests mesos m'encarregaria també de buscar un dataset amb imatges de jugadors de futbol i les seves respectives anotacions. A més, havia d'anar preparant el setup del meu ordinador per a poder realitzar un futur entrenament.

Així doncs, l'objectiu era poder començar després de les vacances de Nadal amb la part pràctica, tenint ja adquirida una bona base teòrica i tenint a punt totes les eines necessàries. El primer que havia d'obtenir era un model de Baseline, uns primers resultats de detecció de jugadors de futbol amb DETR. Un cop obtinguda, els passos a seguir eren comparar-la amb d'altres mètodes, com Yolo, i finalment, utilitzar el temps fins a 1 mes abans d'entregar la memòria (temps que havíem acordat per a la realització d'aquesta) per a intentar millorar els resultats, analitzar els errors, i buscar possibles modificacions.

4.2 Base teòrica

Per a entrar en el món del machine learning i les xarxes neuronals, la tutora d'informàtica em va recomanar un curs de la universitat de Standford anomenat "CS231n: Convolutional Neural Networks for Visual Recognition" [15]. Un cop adquirida tota aquesta base, vaig llegir els articles del transformer [1] i DETR [5]. Amb tot això ja tenia tota la teoria per a poder realitzar la part pràctica. A més, però, durant la realització de la part pràctica vaig haver de llegir diferents articles de modificacions del DETR, detallats a la bibliografia, per a buscar possibles millores del mètode.

4.3 Obtenció del dataset

Mentre anava adquirint tota la base teòrica, se'm va encarregar de buscar un dataset de jugadors de futbol. Va resultar molt complicat ja que n'hi ha molt pocs. A continuació explicaré els 3 que vaig trobar, dels quals finalment només vam utilitzar un degut a les raons que s'explicaran tot seguit:

- **Alfheim** [10]: Dataset amb imatges capturades de 3 partits complets en l'Alfheim Stadium (estadi del club Tromsø IL de Noruega). Les coordenades dels jugadors estan obtingudes mitjançant dispositius GPS. A part, el vídeo està capturat amb una càmera situada al centre d'un dels laterals de l'estadi. Per a relacionar la posició dels jugadors en el camp de futbol (pla XY) amb la posició en les imatges (pla $\hat{XY}$), és necessari realitzar una homografia. En l'article expliquen però que el

Figura 8: Imatges del dataset LCCNN. Les imatges estan generades a partir de les anotacions del dataset amb l'arxiu de codi main.py adjuntat. Com es pot veure, es donen falsos negatius en situacions en les que els jugadors adopten postures diferents de les normals.

marge d'error en les dades fa que no sigui possible obtenir unes dades de qualitat mitjançant l'homografia. Així doncs, malgrat ser un dataset molt gran, per al nostre problema no el podiem utilitzar.

- **LCCNN** [11]: Dataset amb fragments d'imatges de televisió de 2 partits de futbol de la MLS (Major League Soccer). Són 2 vídeos bastant curts, un total de, aproximadament, 2000 imatges. Les anotacions del dataset són generades automàticament mijtançant una light cascaded CNN. Aquest fet comporta que el dataset no és precís. Després d'analitzar les imatges del dataset, vam decidir que no aportaven al nostre dataset degut als errors que hi havia, i a que les deteccions que feia bé eren redundants amb les del dataset D'Orazio[12]. En la figura 8 veiem un exemple de com és el dataset LCCNN.

- **D'Orazio** [12]: Fragments de 2 minuts de 6 càmeres d'un partit de la lliga italiana. Les anotacions estan fetes de manera manual, fet que comporta una alta precisió. Dels 6 vídeos de 2 minuts, se'n obtenen aproximadament 15000 imatges amb anotacions. Aquest és el dataset que hem utilitzat per a l'entrenament i el test dels diferents mètodes. En la figura 9 podem veure una mostra del dataset.

Com que DETR s'entrena amb datasets amb el format de COCO [17], he hagut de modificar les anotacions del dataset D'Orazio per a que fossin compatibles. El script que he fet per a aquesta funció està a l'arxiu main.py adjuntat al treball. A més, com que el dataset proporcionava vídeos enlloc d'imatges, vaig haver d'extreure'n prèviament els frames a partir dels vídeos i quadrar el número de frame extret amb el número de frame de les anotacions. De les aproximadament 15000 imatges que té el dataset, n'hem utilitzat 12500 per a entrenar, i 2500 per al test (5 càmeres per a entrenar, i 1 per avaluar). Així va quedar el dataset preparat per a ser entrenat amb format COCO.

4.4 DETR

Abans de començar a entrenar amb DETR, vaig haver de llegir l'article i entendre bé el seu funcionament. Per a l'entrenament, em vaig baixar el codi que proporcionen al github. Teníem dues opcions: Entrenar desde zero amb el nostre dataset, o fer finetunning amb un

Figura 9: Imatges del dataset D'Orazio. Les imatges estan generades a partir de les anotacions del dataset amb l'arxiu de codi main.py adjuntat. Podem veure que hi ha 3 càmeres disposades a cada lateral del camp, emparellades frontalment 2 a 2.

checkpoint proporcionat per ells. En l'article expliquen que el DETR necessita un dataset molt gran per a ser entrenat desde zero, així que ens hem decantat pel finetunning.

Tot i això, com a experiment, vam posar a entrenar el model desde zero amb diferents optimitzadors per a veure que realment la mida del dataset era insuficient per a que el model convergís. Per a poder visualtizar com anaven variant els valors d'avaluació al entrenar, hem utilitzat la llibreria weights and biases. Després de realitzar el finetunning durant 100 epochs, vam prendre els resultats com a Base-line.

El següent pas, abans de provar altres mètodes, ha estat analitzar el funcionament del model entrenat. Hem agafat 100 imatges en les que el model no tenia una precisió del 100% i hem analitzat els diferents errors. Un dels motius d'error més recurrent era la doble detecció de jugadors o pilota. Tenint present aquest fet, hem estudiat posar un threshold en la similitud (IoU) entre dues bounding box per a eliminar algunes d'aquestes dobles deteccions. Hem calculat els 3 valors màxims de IoU en el ground truth (anotacions d'entrenament) i els hem comparat amb els valors de IoU entre dobles deteccions.

4.5 Yolo

Per a comparar els resultats del DETR en detecció de jugadors i pilota, hem entrenat també diferents versions de Yolo. Per a entrenar Yolo, hem usat mmdetection, un repositori equipat amb moltes eines per a poder entrenar i avaluar diferents arquitectures.

4.6 Deformable-DETR

Al març d'aquest 2021, va sortir una modificació [13] del DETR que presentava el que ells anomenen Deformable-DETR. Mostraven una modificació de l'arquitectura original basada en les deformable convolutions [14]. En l'article s'explica que amb aquesta modificació, el model convergeix 5 vegades més ràpid, i que assoleix el mateix nivell de resultats. Aquest fet el podem veure a la figura 10. Així doncs, utilitzant el repositori mmdetection, hem entrenat el model amb el nostre dataset de futbol per a comparar-lo amb el DETR original.

4.7 SVD-DETR

Per a veure com canvia el model durant l'entrenament, hem accedit als valors propis de les matrius de pesos i n'hem monitoritzat els canvis. Finalment, hem usat la descomposició en valors singulars de les matrius per a introduïr una modificació en el DETR original. A cada epoch, hem agafat el model entrenat, i hem fet el següent:

- Hem obtingut les matrius de pesos W_q, W_k, W_v de tots els encoders i decoders dels transformers.
- Hem fet la descomposició en valors propis d'aquestes matrius $W^i_{q,k,v} = U\Sigma V^T$.
- Hem modificat la matriu Σ en cada cas per tal d'eliminar el 10% dels valors propis amb menys valor.
- Hem tornat a calcular la matriu $W^i_{q,k,v}$ usant la nova Σ. Així doncs, hem modificat totes les matrius de pesos en relació als valors propis d'aquestes.

Amb això, buscavem veure quina influència tenen aquells valors propis amb menys valor, si realment estaven aportant en l'entrenament o només generaven soroll. Així doncs, hem visualitzat l'entrenament amb aquesta petita modificació basada en SVD i l'hem comparat amb l'entrenament del model original.

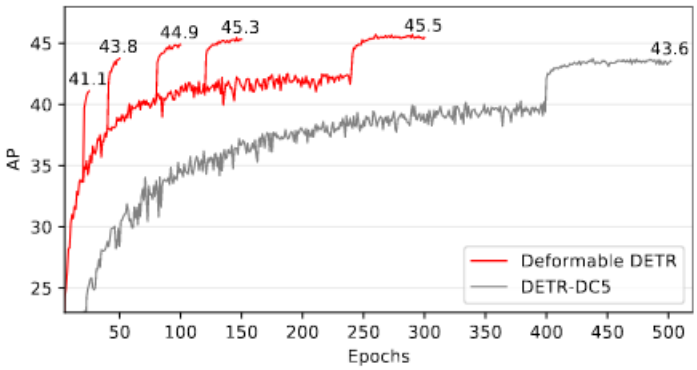


Figura 10: Gràfica extreta de l'article Deformable-DETR [13] on es pot veure el menor temps de convergència del Deformable-DETR respecte el DETR original.

5 Validació del mètode de detecció de jugadors amb transformers

A continuació detallarem tots els resultats que hem anat obtenint durant la realització del treball. El primer experiment que vam dur a terme va ser el de comparar diferents optimitzadors per a entrenar desde zero el DETR, amb l'objectiu d'escollir-ne un per a l'entrenament amb finetunning i, a més, comprovar si realment el dataset no era prou gran.

Figura 11: Comparació dels optimitzadors ASGD, AdamW i Adamax en el dataset de futbol D'Orazio entrenant DETR desde zero.

Els optimitzadors que vam comparar són ASGD, AdamW i Adamax. Com podem comprovar a la figura 11, en els 3 casos obtenim una gran diferència entre la loss de train i de test. Aquest fenomen s'anomena overfitting. A més, al no haver-hi prou imatges en el dataset, l'entrenament s'estanca en una loss molt alta (degut a que troba un mínim local i no pot sortir d'allà). D'aquests resultats obtenim que canviant l'optimitzador no es pot solventar el problema que implica el tenir un dataset reduït. Així doncs, deixem l'optimitzador AdamW per defecte i fem finetunning a partir d'un checkpoint.

Figura 12: La primera gràfica mostra com evoluciona la train loss i la test loss al llarg de les 100 epochs de finetunnning del DETR amb el dataset D'Orazio. En la segona gràfica es mostra com canvia el learning rate.

En la figura 12 podem veure com evoluciona el finetunning de DETR amb el dataset de futbol. Veiem com després de reduïr el learning rate per primer cop es produeix un salt de qualitat, a partir del qual el model arriba ja pràcticament a un mínim local. Els posteriors descensos del learning rate serveixen per a estabilitzar la test loss i així obtenir un model amb resultats més compactes (veiem que la train loss i la test loss coincideixen).

Un cop obtinguda una base-line, hem analitzat el funcionament del model en el nostre dataset. En la figura 13 podem veure el funcionament del DETR abans i després del finetunning en imatges del dataset, i en imatges que no són del dataset. Com es pot observar, amb el finetunning s'eliminen una gran quantitat de falsos positius, i també disminueix, més lleugerament, el nombre de falsos negatius. La comparació de precisió entre el model amb finetunning i sense es pot veure a la figura 23.

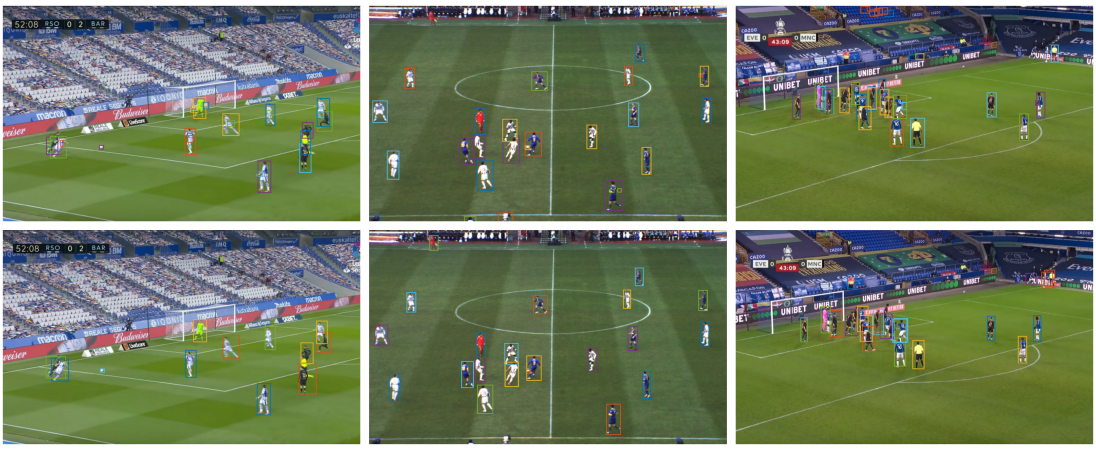


Figura 13: Diferències entre el checkpoint del DETR (fila d'adalt) i el model amb finetuning (fila d'abaix). La columna del mig representa una imatge del conjunt d'entrenament, mentre que les altres dues són imatges extretes de partits per televisió.

Abans de comparar els resultats del DETR amb YOLO i amb el Deformable-DETR, vam analitzar els errors que cometia el model. Com podem veure en la distribució dels errors en la figura 14, la meitat dels errors es produeixen al detectar falsament una pilota. Aquest fet és degut a que tenim molt poques imatges en el dataset en les que aparegui una pilota, així que si ja de per si tenim limitacions per la mida del dataset, en el cas de la pilota encara s'accentuen més.

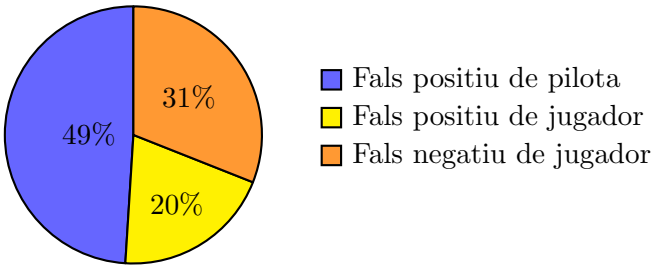


Figura 14: Distribució dels errors de predicció analitzats en 100 imatges del validation set. Els falsos positius de jugador són tots per doble detecció.

En la figura 15 veiem la distribució concreta de cada error. Com podem observar, la doble detecció de la pilota i dels jugadors és el principal motiu d'error del model (54%). Per aquest motiu ens sorgeix la possibilitat d'estudiar la similitud entre les bounding box del ground truth (mitjançant IoU) per a aplicar un treshold i així reduïr aquest error.

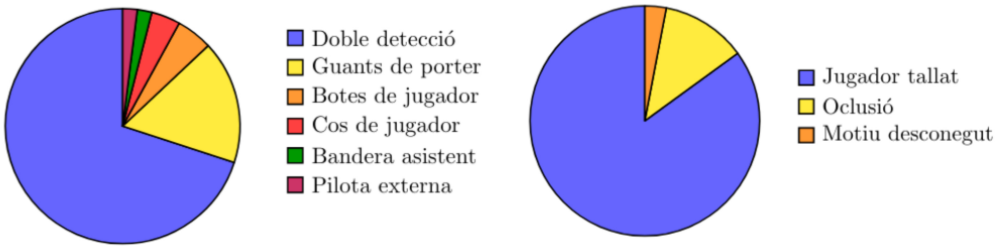


Figura 15: A l'esquerra, la distribució dels motius de falsos positius de pilota. A la dreta, la distibució dels motius de falsos negatius de jugadors.

Figura 16: A l'esquerra veiem una doble detecció amb un alt valor de IoU entre les dues boundig box (0.91). A la dreta, una doble detecció amb un baix valor de IoU (0.5).

Agregant un petit script vam obtenir que els 3 valors màxims de IoU en les anotacions de les imatges d'entrenament eren $0.8897, 0.8834, 0.8784$ (prenem 3 valors per assegurar-nos que no hi hagi un valor molt allunyat de la resta). Podríem aplicar un threshold en un IoU de 0.9, però ens arriscariem a tenir falsos negatius (ja que els valors màxims obtinguts són només pel nostre dataset reduït, i no seria gens extrany trobar valors una mica més elevats en ground truth d'altres imatges). Una altre opció és la d'aplicar un threshold en 0.95, assegurant-nos de no perdre cap detecció correcte. Els números que hem obtingut són aquests:

- En total, el model entrenat fa una doble detecció (ja sigui de pilota, o jugador) aproximadament 260 vegades en el total de les 2600 imatges de test.
- Si apliquem un threshold de 0.95, ens estalviem 17 falsos positius.
- Si apliquem un threshold de 0.90, ens estalviem 70 falsos positius.

Així doncs, veiem que és una opció vàlida per a millorar els resultats (disminuïm el rati de doble detecció entre un 6% i un 27%), però que necessitariem d'un dataset més extens per a obtenir una millora òptima. En la figura 16, veiem un exemple de cas en que ens podria servir el threshold, i un altre exemple en el que no.

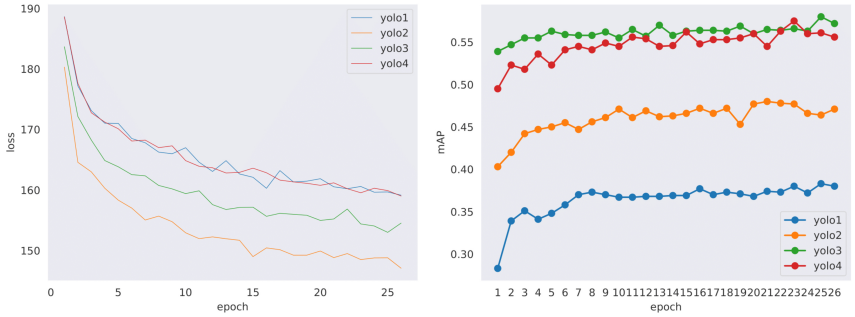


Figura 17: Evolució de la loss (primer gràfic) i la mAP (segon gràfic) durant 25 epochs de finetunning amb diferents checkpoints de yolo. Yolo1, Yolo2 i Yolo3 estan entrenats amb escala d'imatges 320, 416 i 608, respectivament, mentre que Yolo4 està entrenat amb precisió mixta i escala d'imatges 608.

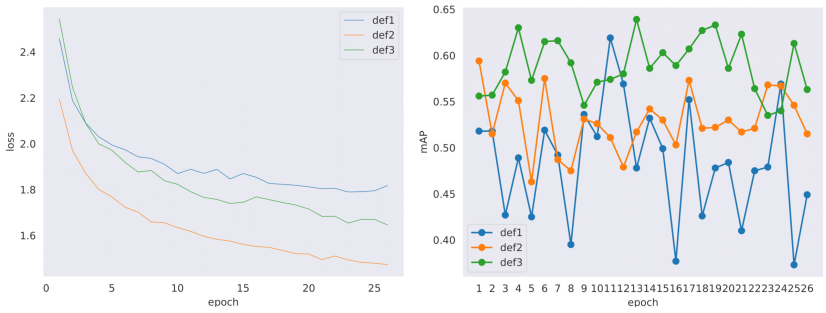


Figura 18: Evolució de la loss (primer gràfic) i la mAP (segon gràfic) durant 25 epochs de finetunning amb diferents checkpoints de Deformable-DETR. Def1 és el model original. Def2 té incorporat un mecanisme iteratiu de refinament de les bounding box. Def3 és una variant inspirada en els detectors d'objectes de dues fases.

Hem entrenat diferents versions de Yolo i Deformable-DETR amb el nostre dataset usant mmdetection. En la figura 17 en podem veure l'evolució de la loss i la mAP en les diferents versions de Yolo, mentre que a la figura 18 tenim l'evolució de la loss i la mAP de les variacions del Deformable-DETR. Les mètriques comparatives amb els altres models les podem veure a la figura 23.

A part de fer una avaluació de les mètriques dels diferents models, hem analitzat gràficament com es comporten en les imatges del nostre dataset. A la figura 19 podem veure el rendiment del Yolo1 abans i després del finetunning. Com podem observar, millora significativament però segueix tenint dificultat en algunes deteccions (per exemple, en postures no habituals dels jugadors).

A la figura 20 veiem el funcionament del Deformable-DETR original abans i després del finetunning. Veiem que millora respecte del Yolo, clarament, però segueix tenint problemes amb falsos positius de pilota (degut al fet que hem explicat prèviament). En general hem observat que el finetuning elimina molts falsos positius, donat que el dataset COCO té 80 classes, i el nostre dataset només en té dues. En el cas dels jugadors, com que els models estan entrenats prèviament per a detectar persones, s'aconsegueixen uns resultats molt bons, mentre que amb la pilota, tot i haver una classe "sports ball" en el dataset COCO, els resultats són clarament inferiors.

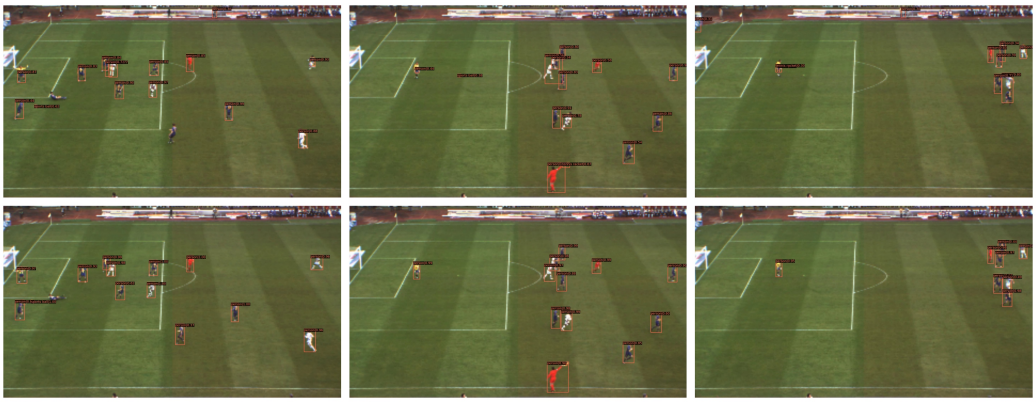


Figura 19: Comparació del rendiment de Yolo1 amb finetunning (fila d'abaix) i sense (fila d'adalt) en 3 imatges de test.

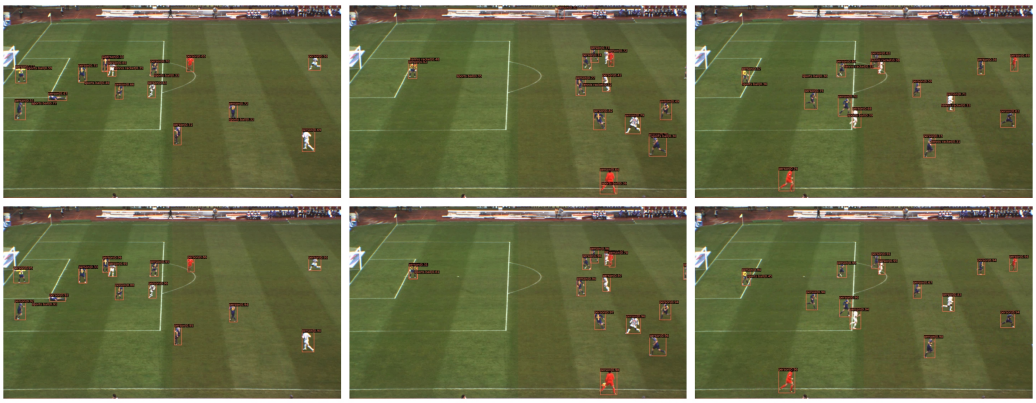


Figura 20: Comparació del rendiment de Def1 amb finetunning (fila d'abaix) i sense (fila d'adalt) en 3 imatges de test.

Un cop hem analitzat el funcionament dels diferents models de Yolo, Deformable-DETR i DETR amb el nostre dataset, hem volgut analitzar com canvia el model de DETR durant l'entrenament. Per a poder-ho visualitzar, hem fet ús de la descomposició en valors singulars de les matrius de pesos del Transformer de DETR. Com es pot observar en la figura 21, els valors singulars de les matrius de pesos decreixen de forma bastant homogènia, però durant l'entrenament augmenten les diferències entre aquests. Per aquest motiu, ens sorgeix la idea d'eliminar el 10% amb menor valor per a veure si estan aportant al model o si realment estan generant soroll.

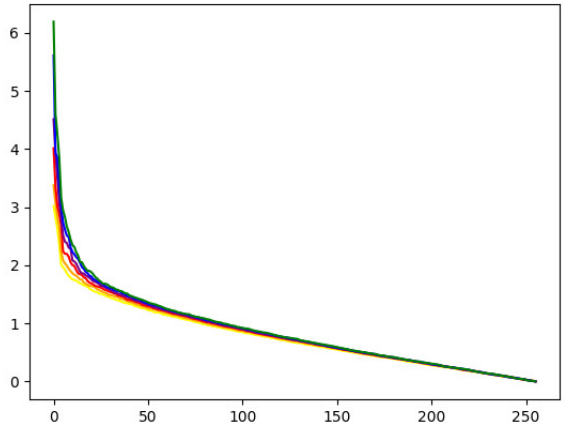


Figura 21: Comparació dels 250 valors singulars d'una matrius de pesos (W_k) del primer Encoder de DETR. Seguint l'escala cromàtica començant pel Groc, cada canvi de color representa 5 epochs d'entrenament. Així, tenim: Groc = Inici, Taronja = 5 epochs, Vermell = 10 epochs, Lila = 15 epochs, Blau = 20 epochs, Verd = 25 epochs.

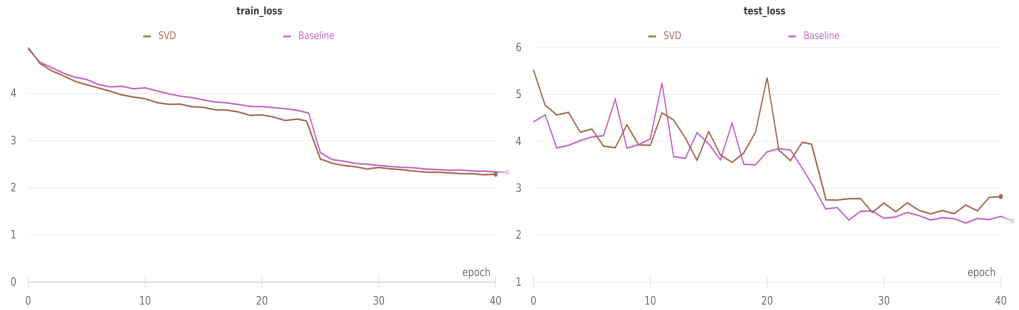


Figura 22: Comparació de la baseline amb l'experiment de SVD. A l'esquerra, la train loss. A la dreta, la test loss.

Així doncs, hem entrenat el model durant 40 epochs modificant, a cada epoch, les matrius de pesos del DETR. Els resulats de l'experiment els podem visualitzar gràficament a la figura 22. S'observa que el valor de la train loss millora respecte la baseline, seguint la mateixa evolució. En canvi, la test loss fluctua molt més i acaba sent pitjor que en la baseline. Per tant, el model amb l'experiment SVD obté una diferència més gran entre la train loss i la test loss. Aquest fenomen és conegut com a Overfitting. La comparació de mètriques amb els altres models està a la figura 23. A la figura 24 podem veure un exemple de com funciona l'experiment amb diferents imatges. Amb això donem per finalitzada la secció on hem mostrat tots els resultats obtinguts durant la realització d'aquest treball.

Model	AP	$AP_{0.5}$	$AP_{0.75}$	AP_s	AP_m	AP_l	AR	AR_s	AR_m	AR_l
DETR	0.348	0.777	0.284	0.167	0.528	0.578	0.460	0.287	0.630	0.662
DETR f	0.563	0.902	0.624	0.400	0.723	0.761	**0.693**	**0.601**	0.782	0.813
DEF1	0.333	0.699	0.363	0.060	0.602	0.657	0.423	0.129	0.714	0.748
DEF1 f	0.449	0.902	0.427	0.249	0.644	0.722	0.550	0.370	0.726	0.785
DEF2	0.313	0.539	0.366	0.017	0.609	0.675	0.398	0.064	0.729	0.750
DEF2 f	0.515	0.952	0.472	0.303	0.724	0.766	0.578	0.364	0.790	0.817
DEF3	0.329	0.577	0.385	0.025	0.633	0.691	0.417	0.095	0.737	0.766
DEF3 f	0.563	**0.971**	0.511	0.393	**0.730**	**0.780**	0.623	0.446	**0.797**	**0.825**
YOLO1	0.128	0.333	0.075	0.007	0.234	0.504	0.235	0.089	0.363	0.584
YOLO1 f	0.380	0.764	0.359	0.185	0.566	0.686	0.475	0.293	0.648	0.747
YOLO2	0.183	0.433	0.143	0.020	0.322	0.623	0.304	0.104	0.488	0.687
YOLO2 f	0.471	0.857	0.452	0.314	0.619	0.701	0.552	0.409	0.691	0.749
YOLO3	0.274	0.534	0.283	0.027	0.522	0.619	0.366	0.119	0.608	0.679
YOLO3 f	**0.572**	0.924	**0.648**	**0.463**	0.680	0.703	0.631	0.523	0.736	0.760
YOLO4	0.281	0.529	0.300	0.019	0.544	0.573	0.378	0.133	0.623	0.639
YOLO4 f	0.556	0.903	0.610	0.424	0.688	0.702	0.632	0.520	0.744	0.758
SVD f	0.412	0.804	0.427	0.169	0.646	0.742	0.563	0.402	0.717	0.792

Figura 23: Comparació dels resultats després del finetunning (indicat amb "f", 25 epochs per Yolo i Def; 40 epochs per SVD, 100 epochs per DETR) i sense finetunning dels diferents models de Yolo, DETR (SVD f i DETR f parteixen de DETR) i Deformable-DETR. AP vol dir la mitjana dels Average Precision amb valors de threshold entre 0.5 i 0.95, amb pas de 0.05; $AP_{0.5}$ és Average Precision amb threshold 0.5; $AP_{0.75}$ és Average Precision amb threshold 0.75; AP_s és Average Precision de les bounding box de mida petita (pilota). AP_m és Average Precision de les bounding box de mida mitjana. AP_l és Average Precision de les bounding box de mida gran. AR és Average Recall. AR_s és Average Recall de les bounding box de mida petita. AR_m és Average Precision de les bounding box de mida mitjana. AR_l és Average Precision de les bounding box de mida gran.

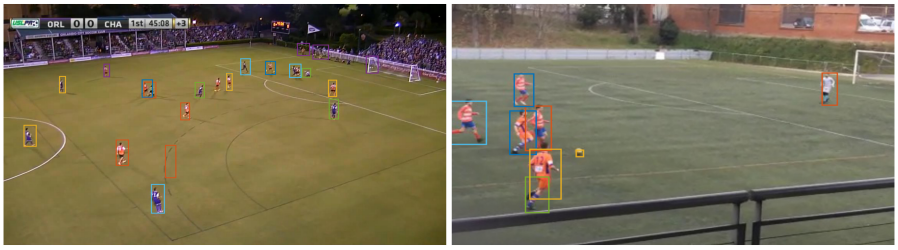


Figura 24: Exemples de funcionament del model amb la variació de SVD.

6 Conclusions i treball futur

A l'inici del treball, em vaig marcar els objectius d'adquirir una base teòrica de les branques de machine learning i xarxes neuronals, i utilitzar un model (en aquest cas, DETR) per a detectar jugadors de futbol en imatges. A més, voliem comparar els resultats i introduïr possibles millores.

Pel que fa al meu coneixement teòric, m'agradaria dir que estic molt satisfet amb el resultat. Després de fer els cursos recomanats per la tutora, i buscar informació en blogs i vídeos, he assolit la base que desitjava. Així, he estat capaç de llegir i entendre articles científics de les branques del machine learning i les xarxes neuronals en detecció d'objectes, o d'explicar el funcionament de les xarxes neuronals convolucionals en un nivell per a un lector no especialitzat. I és que com diuen, la millor manera d'aprendre un tema és ensenyant-lo.

Per la banda d'aplicar transformers en imatges de futbol, també he aconseguit l'objectiu marcat. He aconseguit trobar un dataset útil d'imatges de futbol i amb aquest he pogut entrenar el DETR. En aquest sentit també he après com funcionen els Transformers, que són una arquitectura molt nova i amb moltes possibilitats. He vist, però, que els resultats obtinguts, malgrat ser raonablement bons, no són prou precisos com per a obtenir informació de qualitat (a nivell d'ús per a clubs de futbol) a través de la detecció automàtica de jugadors amb Transformers.

He aconseguit també comparar els resultats obtinguts del DETR amb altres mètodes, com Yolo i Deformable-DETR. De tota manera, al ser un dataset tant reduït, els diferents models han tingut les mateixes dificultats. En el cas del SVD, ha estat molt interessant veure com evoluciona per dins un model com DETR, però l'experiment no ha pogut millorar-ne el rendiment. Una millora que si que he pogut aplicar al DETR és el càlcul d'un threshold per a la doble detecció d'imatges, que podria haver estat més precisat en el cas de tenir un dataset més gran.

En quant a línies a seguir per a treball futur, hi ha varies opcions. Una opció molt clara que milloraria el rendiment seria fer manualment un dataset més extens amb imatges de partits de futbol, amb diferents perspectives, accions, etc.

Una altra línia futura seria considerar el problema de tracking de jugadors (usant vídeos i no imatges). En aquest treball, hem decidit estudiar només el cas de detecció en imatges per diferents motius: El primer és que el que solen fer els algorismes de seguiment de jugadors és detectar-los en imatges i, després, aplicar algun mètode que en relacioni les diferents deteccions.

L'altre motiu és que el problema de tracking d'objectes es pot considerar com un subproblema del de detecció d'objectes, i crèiem més oportú encarar el problema més general per a obtenir uns coneixements més troncals. Amb això comentat, pot ser molt interessant estudiar alguna manera de que el DETR utilitzi la informació aportada pels frames anteriors (i posteriors, segons la necessitat de fer les deteccions en directe o post-partit) per a la detecció dels objectes.

Així doncs, valoro com a molt positiva l'experiència viscuda fent aquest treball, en el que he après molt i he aconseguit els meus objectius inicials.

Referències

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser,L., Polosukhin, I.: Attention is all you need. In: NeurIPS (2017)

[2] Jaron Collis: Glossary of Deep Learning: Word Embedding. In: Medium (2017)

[3] Amirhossein Kazemnejad: Transformer Architecture: The Positional Encoding. In: Kazemnejad's blog (2019)

[4] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, Tie-Yan Liu: On Layer Normalization in the transformer architecture.
`arXiv:2002.04745v2 [cs.LG]` , juny de 2020.

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko: End-to-End Object Detection with Transformers. `arXiv:2005.12872v3 [cs.CV]` , maig de 2020.

[6] Jay Alammar: The illustrated Transformer. In: jalammar.github.io (2018)

[7] Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S.: Generalized intersection over union. In: CVPR (2019).

[8] Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: ICLR (2017).

[9] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollar, P.,Zitnick, C.L.: Microsoft COCO: Common objects in context. In: ECCV (2014).

[10] Svein Arne Pettersen, Dag Johansen, Håvard Johansen, Vegard Berg-Johansen Vamsidhar Reddy Gaddam, Asgeir Mortensen, Ragnar Langseth, Carsten Griwodz, Håkon Kvale Stensland, Pål Halvorsen: Soccer video and player position dataset. `http://home.ifi.uio.no/paalh/dataset/alfheim/`, 2013

[11] K. Lu, J. Chen, J. J. Little, H. He: Light Cascaded CNNs for Player Detection. `arXiv:1709.10230v1 [cs.CV]` , sestembre de 2017.

[12] T. D'Orazio, M. Leo, N. Mosca, P. Spagnolo, andP. Mazzeo. A semi-automatic system for ground truth generation of soccer video sequences. In Proc of AVSS, pages 559–564, 2009.

[13] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, Jifeng Dai: Deformable DETR: Deformable Transformers for End-to-End Object Detection. `arXiv:2010.04159v4 [cs.CV]` , març de 2021.

[14] ifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In ICCV, 2017.

[15] Standford University: CS231n: Convolutional Neural Networks for Visual Recognition. `http://cs231n.stanford.edu/2018/syllabus.html`

[16] SkillCorner, `https://www.skillcorner.com/`

[17] Object detection on COCO in paperswithcode.com/sota/object-detection-on-coco

[18] Gunther Hammerlin, Karl-Heinz Hoffmann: Numerical Mathematics. Springer (1991).