



TREBALL DE FI DE GRAU

SISTEMA AUTOMATITZAT PER A LA CREACIÓ DE PLANTILLES ÒPTIMES A FC 24

Josep Gabiel Fornes Reynes

Grau d'Enginyeria Informàtica

Escola Politècnica Superior

Any acadèmic 2024-25

SISTEMA AUTOMATITZAT PER A LA CREACIÓ DE PLANTILLES ÒPTIMES A FC 24

Josep Gabiel Fornes Reynes

Treball de Fi de Grau

Escola Politècnica Superior

Universitat de les Illes Balears

Any acadèmic 2024-25

Paraules clau del treball: algoritme genètic, optimització de plantilles

Tutor: Miquel Miró Nicolau, Javier Varona Gómez

Autoritz la Universitat a incloure aquest treball en el repositori
institucional per consultar-lo en accés obert i difondre'l en línia, amb
finalitats exclusivament acadèmiques i d'investigació

Autor/a	Tutor/a
Sí <input checked="" type="checkbox"/>	No <input type="checkbox"/>
Sí <input checked="" type="checkbox"/>	No <input type="checkbox"/>

Gràcies a tots els professors que m'han ajudat a arribar a aquesta fita dels meus estudis i, especialment, als meus companys que m'han acompanyat durant tot aquest trajecte.

SUMARI

Sumari	iii
Índex de figures	v
Índex de taules	vi
Acrònims	vii
Resum	ix
1 Introducció	1
2 Tecnologies i eines emprades	3
2.1 Python	3
2.1.1 OpenCV (cv2)	3
2.1.2 PIL (Pillow, ImageGrab)	3
2.1.3 Ultralitycs YOLO (ultralytics.YOLO)	4
2.1.4 easyOCR	4
2.1.5 PyMySQL	4
2.1.6 Levenshtein	4
2.1.7 Llibreries de suport	4
2.2 Entorn Conda	5
2.3 PyCharm	5
2.4 MySQL	5
2.5 RStudio	5
2.6 Git i GitHub	6
2.7 Label Studio	6
2.8 Google Colab	6
3 Introducció al Ultimate Team	9
3.1 Context històric i evolució	9
3.2 Fonaments del mode Ultimate Team	9
3.2.1 Cartes i classificació	9
3.2.2 Economia, monedes i paquets	9
3.2.3 Química i sinergies	10
3.2.4 Evolucions	10
3.3 Les <i>Squad Building Challenges</i> (Squad Building Challenges (SBC))	10
3.3.1 Què és una SBC?	10

3.3.2	Tipologia	10
3.3.3	Impacte al mercat	10
3.4	Relevància per al Treball de Fi de Grau	11
3.4.1	Problema d'optimització de plantilles	11
3.4.2	Objectiu del projecte	11
4	Obtenció de jugadors	13
4.1	Automatització de captures de pantalla	13
4.2	Retalls dels elements importants i obtenció de dades	14
4.2.1	Fase Inicial	15
4.2.2	Fase definitiva	21
5	Generació de plantilles	31
5.0.1	Algorisme genètic: descripció general	31
5.1	Definició de requisits per a totes les SBC	32
5.2	Algoritme evolutiu	34
5.2.1	Paràmetres bàsics	34
5.2.2	Entrades	34
5.2.3	Població inicial	34
5.2.4	Càlcul d'informació de l'equip	35
5.2.5	Funció fitness	37
5.2.6	Creuament	40
5.2.7	Mutació	41
6	Resultats i discussió	43
7	Conclusions	45
A	Apèndix	47
	Bibliografia	49

ÍNDEX DE FIGURES

4.1	Exemple de pantalla amb les cartes del club.	14
4.2	Indicacions dels elements de la carta.	16
4.3	Recopilació d'algunes de les imatges utilitzades per entrenar YOLO	22
4.4	Resultats del test.	23
4.5	Recopilació d'algunes de les deteccions de les imatges de la Figura 4.3	24
4.6	Recopilació d'algunes de les imatges utilitzades per entrenar el segon model de YOLO	25
4.7	Resultats del test.	26
4.8	Recopilació d'algunes de les deteccions de les imatges de la Figura 4.6	27

ÍNDEX DE TAULES

4.1	Constants de la carta (Amplitud i alçada)	15
4.2	Coordenades de les cartes dins la captura de pantalla.	16
4.3	Coordenades relatives dins de cada carta per a l'extracció dels diferents elements.	17
4.4	Exemples dels retalls dels diferents elements de cada carta.	17
4.5	Columnes principals del <i>EA FC 24 FUT Players Dataset</i>	18
4.6	Columnes principals del <i>EA Sports FC 24 Complete Player Dataset</i>	18
4.7	Estructura de la classe Jugador	20
4.8	Tendències principals observades a les corbes d'entrenament	23
4.9	Resum de l'entrenament mostrat a la Fig. 4.7	26
4.10	Columnes principals de la base de dades procedent de <i>FUTBIN</i>	28
4.11	Estructura de la classe Jugador	29
5.1	Puntuació de la química	36
5.2	Bonificacions addicionals de cartes especials	36

ACRÒNIMS

SBC Squad Building Challenges

OCR Optical Character Recognition

JSON JavaScript Object Notation

PIL Python Imaging Library

YOLO You Only Look Once

IDE entorn de desenvolupament integrat

GPU unitats de processament gràfic

UT Ultimate Team

TFG Treball de Final de Grau

RESUM

Ultimate Team és un dels modes de joc més populars del videojoc FC 24. En aquest mode, els usuaris poden crear i gestionar el seu propi equip de futbol utilitzant cartes de jugadors reals, que es poden obtenir mitjançant sobres, el mercat de transferències o completant desafiaments. Cada carta representa un jugador amb atributs i estadístiques específiques, i l'objectiu principal és construir una plantilla competitiva tenint en compte factors com la química entre jugadors, la formació tàctica, la nacionalitat i el club d'origen.

Aquest mode fomenta l'estratègia i la gestió esportiva, ja que els usuaris han d'equilibrar el rendiment competitiu amb la gestió econòmica per millorar progressivament la seva plantilla. Dins d'Ultimate Team, s'inclouen reptes anomenats SBC, que premien els jugadors per completar equips que compleixin una sèrie de requisits específics relacionats amb la mitjana global de l'equip, la química o la procedència dels jugadors. Cal saber que construir aquestes plantilles manualment pot resultar una tasca costosa i laboriosa, especialment si es busca minimitzar el cost total complint alhora totes les restriccions.

Amb l'objectiu de facilitar i automatitzar aquest procés, s'ha desenvolupat una aplicació que genera plantilles òptimes a partir de les cartes disponibles al club de l'usuari. Per assolir-ho, es capturen pantalles de la col·lecció de cartes, les quals es processen mitjançant tècniques de visió per computador per detectar i segmentar cada carta. Posteriorment, es fa servir un reconeixement òptic de caràcters (Optical Character Recognition (OCR)) per extreure'n els atributs textuais rellevants. Amb aquesta informació, es consulta una base de dades per identificar els jugadors i crear una base estructurada amb totes les cartes disponibles.

A continuació, el sistema utilitza un motor de generació automàtica de plantilles basat en un algorisme evolutiu, que selecciona els jugadors més adequats segons els requisits d'un SBC. Aquest motor optimitza la selecció considerant tant el cost com la posició i els atributs de cada jugador, assegurant que l'equip compleixi totes les condicions al menor cost possible.

El projecte ha estat validat amb dades reals i mostra aplicacions potencials tant per a jugadors interessats en optimitzar els seus recursos dins del joc com per a investigadors en l'àmbit de la intel·ligència artificial aplicada a entorns de simulació i videojocs.

CAPÍTOL



1

INTRODUCCIÓ

Els videojocs de simulació esportiva han experimentat un creixement notable en les darreres dècades, combinant la competició tradicional amb elements de gestió i estratègia. Dins d'aquest panorama, Ultimate Team s'ha consolidat com un dels modes de joc més populars del videojoc FC 24, atorgant als jugadors la possibilitat de crear i gestionar el seu propi equip de futbol utilitzant cartes virtuals de jugadors reals. Cada carta representa un jugador amb unes característiques i estadístiques específiques, i la configuració de l'equip exigeix tenir en compte múltiples factors, com ara la química entre jugadors, les nacionalitats, els clubs d'origen i la formació tàctica emprada.

Aquest entorn planteja un repte de gestió complexa per part dels usuaris, que no només han d'optimitzar el rendiment esportiu del seu equip, sinó que també han de gestionar els recursos econòmics de forma eficient. Dins d'Ultimate Team, els anomenats SBC constitueixen un mecanisme fonamental que incentiva els jugadors a construir equips que compleixin unes condicions molt específiques, a canvi de recompenses. Aquestes condicions poden incloure restriccions sobre la mitjana global de l'equip, la química interna, el nombre de jugadors d'una nacionalitat determinada o l'affiliació a certs clubs. A més, un aspecte crucial a tenir en compte és que, un cop completat un SBC, tots els jugadors que han format part de la plantilla desapareixen del club de l'usuari. Aquesta característica incrementa la necessitat de construir plantilles que siguin òptimes, minimitzant el cost i l'impacte en la col·lecció personal de cartes.

El problema que es pretén resoldre en aquest treball és la dificultat que suposa, per a l'usuari, la generació manual d'equips òptims dins d'Ultimate Team, en un entorn amb múltiples restriccions i una elevada variabilitat de cartes disponibles. Aquesta problemàtica no només té implicacions dins del joc en si, sinó que constitueix un cas d'estudi interessant dins dels àmbits de la visió per computador, el reconeixement de patrons i l'optimització algorísmica. A nivell tecnològic, abordar aquest problema implica desafiaments relacionats amb l'extracció automàtica d'informació visual, la identificació robusta de dades textuales i la presa de decisions basada en múltiples criteris.

Per donar resposta a aquesta problemàtica, s'ha desenvolupat una aplicació que

1. INTRODUCCIÓ

automatitza tot el procés d'obtenció i processament de dades de les cartes disponibles al club de l'usuari, així com la generació de plantilles òptimes segons els requisits d'un SBC. El sistema implementa un flux de treball que comença amb la captura automàtica de pantalles del joc, continua amb la segmentació de les cartes mitjançant tècniques de visió per computador, i segueix amb l'extracció de dades textuales a partir de reconeixement òptic de caràcters (OCR). Amb la informació obtinguda, el sistema es connecta a una base de dades externa que conté informació detallada sobre totes les cartes disponibles al joc, amb l'objectiu d'enriquir les dades recollides i completar la identificació de cada carta del club. A partir d'aquesta informació, es construeix un arxiu en format JavaScript Object Notation (JSON) que recull de manera estructurada totes les característiques rellevants de les cartes disponibles. Posteriorment, es fa ús de tècniques d'intel·ligència artificial, concretament d'algorismes evolutius, per desenvolupar un motor de generació automàtica de plantilles. Aquests algorismes, inspirats en els mecanismes de selecció natural, permeten seleccionar de manera òptima els jugadors que millor compleixen els requisits dels reptes, minimitzant el cost total de la plantilla i preservant, en la mesura del possible, els actius més valuosos del club.

La memòria es divideix en diversos capítols. En primer lloc, es contextualitza l'entorn d'Ultimate Team i s'aprofundeix en la descripció dels SBC i les seves característiques. A continuació, s'exposa el desenvolupament del sistema, detallant les tècniques de captura, segmentació i reconeixement de dades emprades. Posteriorment, es descriu el disseny del motor d'optimització basat en intel·ligència artificial i s'analitzen els resultats obtinguts mitjançant casos pràctics. Finalment, es presenten les conclusions del treball, s'avaluen les possibles millores futures i es discuteixen les aplicacions potencials d'aquesta metodologia en altres entorns similars.

CAPÍTOL

2

TECNOLOGIES I EINES EMPRADES

En el desenvolupament d'aquest projecte s'han utilitzat diverses tecnologies i eines, cadascuna seleccionada per les seves característiques específiques i la seva adaptabilitat als requisits del treball. A continuació, es detalla cadascuna d'elles, així com la seva funció i aportació al projecte.

2.1 Python

Python ha estat el llenguatge principal de programació emprat en aquest treball. La seva elecció es justifica per la seva sintaxi senzilla i llegible, la gran disponibilitat de biblioteques especialitzades, la comunitat activa i la facilitat d'integració amb altres tecnologies. Python ha estat utilitzat per desenvolupar totes les fases del flux de treball, des de la captura de pantalles fins al processament d'imatges, reconeixement de text i generació automàtica de plantilles.

Les principals biblioteques utilitzades han estat:

2.1.1 OpenCV (cv2)

OpenCV (Open Source Computer Vision Library) és una llibreria oberta i extensible enfocada al tractament i anàlisi d'imatges i vídeos. S'ha utilitzat per detectar, segmentar i manipular cartes dins de captures de pantalla, identificar regions d'interès específiques i preparar les imatges per a l'extracció posterior d'informació textual.

2.1.2 PIL (Pillow, ImageGrab)

Pillow és una bifurcació moderna de la llibreria Python Imaging Library (PIL), destinada a la manipulació d'imatges. S'ha emprat concretament la funció ImageGrab per realitzar captures automàtiques de la pantalla, permetent generar una col·lecció d'imatges directament des del videojoc sense intervenció manual.

2.1.3 Ultralytics YOLO (ultralytics.YOLO)

You Only Look Once (YOLO) és una de les arquitectures més populars per a la detecció d'objectes en imatges en temps real. La implementació de YOLO proporcionada per Ultralytics permet detectar de manera ràpida i precisa les cartes dins de les captures de pantalla, localitzant-ne les posicions exactes. Això ha permès automatitzar el retall de cartes i preparar-les per a processament OCR posterior.

2.1.4 easyOCR

easyOCR és una llibreria de reconeixement òptic de caràcters (OCR) compatible amb múltiples idiomes i dissenyada per extreure text directament d'imatges. En aquest projecte s'ha emprat per extreure automàticament informació clau de cada carta, com el nom o les característiques numèriques del jugador, proporcionant una capa de comprensió semàntica a partir de dades visuals.

2.1.5 PyMySQL

PyMySQL és una llibreria que permet la connexió i operació amb bases de dades MySQL des de Python, sense necessitat d'instal·lar connectors nadius externs. S'ha utilitzat per consultar una base de dades amb informació detallada sobre les cartes disponibles en el joc. Aquesta connexió ha estat fonamental per validar i completar les dades obtingudes mitjançant visió per computador i OCR, assegurant la seva integritat i precisió.

2.1.6 Levenshtein

La llibreria Levenshtein proporciona funcionalitats per calcular la distància d'edició entre dues cadenes de text, mesurant el nombre mínim d'operacions necessàries per transformar una cadena en una altra. S'ha utilitzat en aquest projecte per comparar noms obtinguts mitjançant l'OCR amb els noms dels diferents registres de la base de dades. Això ens permet calcular la distància de Levenshtein entre cadenes i seleccionar automàticament el registre amb el nom més proper, assegurant així una identificació precisa.

2.1.7 Llibreries de suport

- **os**: per gestionar rutes de fitxers, crear carpetes automàticament i manipular fitxers dins del sistema operatiu.
- **glob**: per buscar llistats de fitxers seguint patrons dins de directoris.
- **datetime**: per crear marques de temps (timestamps) i registrar esdeveniments en el procés de captura i processament.
- **time**: per controlar intervals de temps en l'execució del codi, especialment en la captura automàtica d'imatges.
- **random**: per generar equips inicials aleatoris i operacions genètiques (crossover i mutació) en l'algorisme evolutiu.

- **re**: per la manipulació i validació de cadenes de text mitjançant expressions regulars.
- **json**: per estructurar i emmagatzemar la informació de les cartes en fitxers JSON, facilitant també tasques de depuració.
- **winsound**: per emetre avisos acústics quan es realitza una captura de pantalla, indicant a l'usuari que pot avançar a la següent pantalla.

2.2 Entorn Conda

Per a la gestió de les dependències i la configuració dels paquets utilitzats en el projecte, s'ha optat per la creació d'un entorn virtual Conda. Això ha garantit la compatibilitat entre les diferents versions de biblioteques i ha facilitat la reproduïbilitat del projecte en diferents sistemes.

A més, Conda ha simplificat la instal·lació de paquets que tenen dependències complexes, com OpenCV i Ultralight YOLO, assegurant un entorn estable i controlat per al desenvolupament.

2.3 PyCharm

Per a la programació en Python, s'ha utilitzat l'entorn de desenvolupament integrat (IDE) PyCharm. Aquest IDE proporciona eines avançades d'edició de codi, depuració, control de versions i integració amb entorns virtuals. La integració nativa amb Conda ha facilitat la gestió dels paquets i entorns de manera eficient.

La facilitat de configuració d'estructures de projecte, així com la disponibilitat d'eines d'autocompletat de codi i navegació ràpida, han incrementat notablement la productivitat en el desenvolupament.

2.4 MySQL

La base de dades MySQL ha estat utilitzada per emmagatzemar informació estructurada sobre les cartes de jugadors del videojoc FC 24. En aquesta base de dades s'han registrat atributs com el nom del jugador, puntuació general, posició, club, lliga, nacionalitat, versió de la carta, preu, cama bona, estrelles de cama dolenta, estrelles de skills, estadístiques individuals i gènere del jugador/a.

La utilització de MySQL ha permès realitzar consultes eficients per identificar ràpidament les cartes corresponents a les imatges capturades, enriquint així la informació parcial obtinguda per visió per computador i OCR.

2.5 RStudio

S'ha emprat RStudio per al preprocessament de dades dins de la base de dades, específicament:

- Eliminació de registres innecessaris.

2. TECNOLOGIES I EINES EMPRADES

- Creació d'una columna específica per al gènere del jugador/a.
- Correcció de noms de clubs incorrectes o inconsistents.

Aquest treball de neteja de dades ha estat fonamental per garantir la consistència i la qualitat de la informació emmagatzemada.

2.6 Git i GitHub

Per al control de versions i la gestió del codi font s'han utilitzat Git i GitHub. Aquestes eines han permès:

- Fer un seguiment acurat de l'evolució del projecte.
- Gestionar branques de desenvolupament separades per funcionalitats.
- Mantenir còpies de seguretat regulars del codi.
- Facilitar la documentació i l'organització del projecte.

GitHub ha servit també com a plataforma per a la presentació final del projecte, afavorint la traçabilitat i transparència del desenvolupament.

2.7 Label Studio

Per a la creació del conjunt de dades anotades, s'ha utilitzat Label Studio. Aquesta eina ha permès definir de manera manual les regions d'interès dins de les captures de pantalla, establint les anotacions necessàries per entrenar correctament el model YOLO.

2.8 Google Colab

L'entrenament del model s'ha realitzat utilitzant Google Colaboratory (Colab), aprofitant la disponibilitat d'unitats de processament gràfic (GPU) per accelerar el procés.

S'ha elaborat un quadern Jupyter específic, que conté totes les fases de l'entrenament:

- Configuració de l'entorn i les biblioteques necessàries.
- Definició de l'estructura del conjunt de dades segons els estàndards de YOLO.
- Inicialització d'un model YOLOv8 preentrenat.
- Ajust dels paràmetres d'entrenament (epochs, learning rate, augmentació de dades).
- Entrenament del model i validació dels resultats.

Un cop finalitzat l'entrenament, el model optimitzat s'ha exportat i integrat dins de l'aplicació principal, permetent la detecció automàtica de cartes durant el processament.

Conclusió

Aquesta combinació de tecnologies ha permès abordar amb èxit totes les fases del projecte: des del tractament de dades visuals, la gestió de bases de dades, el desenvolupament d'algorismes d'intel·ligència artificial, fins a l'anàlisi i validació dels resultats.

INTRODUCCIÓ AL ULTIMATE TEAM

3.1 Context històric i evolució

Els orígens Ultimate Team (UT) es remunten a *FIFA 09*, quan EA Sports va llançar un mode basat en cartes que permetia als usuaris confeccionar un equip propi per competir contra la IA o altres jugadors en línia. En pocs anys UT es va convertir en la principal font d'ingressos de la franquícia: el model de monetització basat en paquets (packs) de cartes i microtransaccions va consolidar-se fins a generar més de 1.600 M USD anuals (dades internes d'EA, 2024). L'any 2023 la sèrie va abandonar la marca *FIFA* i es va reestrenar com *EA SPORTS FC*, mantenint UT—ara anomenat **Football Ultimate Team (FUT)**—com a pilar central.

3.2 Fonaments del mode Ultimate Team

3.2.1 Cartes i classificació

Cada carta representa un futbolista real amb atributs numèrics (ritme, xut, passada, defensa, físic, etc.) i una valoració global (OVR). Les cartes bàsicament es classifiquen en bronze, plata i or, i es complementen amb versions especials que apareixen durant la temporada—*Team of the Week (IF)*, *Team of the Season, Icons, Heroes*, etc. Les cartes *Icon* homenatgen llegendes històriques, mentre que les *Hero* premien jugadors icònics de clubs o lligues determinades, però amb un nivell competitiu lleugerament inferior al de les *Icons*.

3.2.2 Economia, monedes i paquets

La moneda principal (coins) s'obté disputant partits i venent cartes al mercat de transferències; els *FC Points*, en canvi, s'adquireixen amb diners reals i serveixen per comprar paquets de manera immediata. EA introduceix *Price Ranges* per limitar la inflació i l'activitat il·lícita en el mercat. Aquest model ha generat polèmica: diversos reguladors

3. INTRODUCCIÓ AL ULTIMATE TEAM

europeus han investigat si els paquets constitueixen formes de joc d'atzar il·legal (casos als Països Baixos i Bèlgica, 2019–2022).

3.2.3 Química i sinergies

El rendiment d'una plantilla no depèn només de l'OVR: el sistema de química recompença les combinacions de lliga, club i nacionalitat, de manera que escollir jugadors adients esdevé un problema d'optimització amb restriccions.

3.2.4 Evolucions

A EA FC 24 s'ha incorporat la mecànica d'*Evolutions*, que permet millorar estadístiques concretes d'un jugador després de completar reptes definits. Aquesta característica amplia l'espai de decisions estratègiques del jugador i afegeix un nou nivell de profunditat a la gestió de plantilla.

3.3 Les *Squad Building Challenges* (SBC)

3.3.1 Què és una SBC?

Una *Squad Building Challenge* és un repte fora del terreny de joc que sol·licita elaborar una plantilla amb requisits específics—màxims de valoració global, química mínima, nacionalitats o lligues concretes, jugadors únics, etc.—i, un cop presentada, els futbolistes s'esborren del club a canvi d'una recompensa (paquet, carta especial, objecte cosmètic...).

3.3.2 Tipologia

- **SBC Bàsiques i Fundacionals:** pensades per a usuaris novells, amb requisits senzills i recompenses bàsiques.
- **SBC Híbrid (*Hybrid Leagues, Hybrid Nations*):** demanden combinacions complexes de lligues i nacionalitats.
- **SBC de Jugador o Icona:** exigeixen plantilles d'alt nivell (87–90 OVR) i ofereixen una carta premium (Icon, Hero o jugador promocional).
- **SBC Repetibles vs. Úniques:** les primeres es poden completar diverses vegades abans de la data de caducitat; les segones, només una.
- **SBC En Viu (Live) i de Temporada:** lligades a esdeveniments reals (Black Friday, FUT Birthday) i desapareixen en pocs dies.

3.3.3 Impacte al mercat

Quan EA publica una SBC exigent, la demanda de cartes “fòra de meta” però amb mitjana alta (82–88 OVR) puja en qüestió de minuts, fet que provoca pics de preu que els inversors aprofiten per especular. Aquest fenòmen—coneugut com a *SBC fodder boom*—és clau per entendre la volatilitat del mercat FUT.

3.4 Relevància per al Treball de Fi de Grau

3.4.1 Problema d'optimització de plantilles

Resoldre una SBC es pot modelar com un problema de satisfacció de restriccions amb múltiples objectius: minimitzar el cost (coins) i complir simultàniament requisits de valoració, química i raritat. El nombre de combinacions creix exponencialment amb la grandària de la base de dades de jugadors (més de 20.000 cartes actives en un cicle anual).

3.4.2 Objectiu del projecte

El present TFG proposa la **generació automàtica de plantilles òptimes per a SBC** utilitzant exclusivament els jugadors disponibles al club, i aplicant tècniques d'intel·ligència artificial —principalment un algorisme genètic combinat amb heurístiques de química i valoració— per:

1. **Minimitzar el cost marginal en monedes**, evitant la compra de noves cartes i priorititzant la utilització de jugadors intransferibles del club.
2. **Complir al 100 % els requisits de cada SBC** (valoració mitjana, química, nacionals, lligues, rareses, etc.) amb la menor pèrdua de valor futur; és a dir, sense sacrificar cartes “meta” que l’usuari tingui a l’equip competitiu.
3. **Proporcionar múltiples solucions alternatives** perquè el jugador pugui triar segons la seva estratègia (conservar mitjanes altes, liquidar excedents, maximitzar recompenses, etc.).

Aquest enfocament integra coneixement del mercat FUT, modelatge de restriccions i optimització multiobjectiu sobre grafos de jugadors, aportant valor tant acadèmic com pràctic.

Conclusions

Ultimate Team no és només un mode arcade: és un ecosistema econòmic complex on milers de jugadors interactuen diàriament amb un mercat virtual regulat per EA. Les SBC, per la seva naturalesa formalitzable, constitueixen un laboratori ideal per aplicar tècniques d'optimització. El fet de limitar-nos als recursos interns del club converteix la generació de plantilles en un problema de satisfacció de restriccions amb forta dependència temporal del mercat. Aquest Treball de Final de Grau (TFG) demostra que, mitjançant algoritmes evolutius i heurístiques específiques de FUT, és possible automatitzar la construcció de plantilles que minimitzin el cost i maximitzin l’aprofitament d’actius, oferint alhora flexibilitat estratègica a l’usuari final.

OBTENCIÓ DE JUGADORS

Com s'ha comentat anteriorment, un dels principals reptes d'aquest projecte és l'extracció de les cartes del club del jugador dins del mode de joc *Ultimate Team*. Aquest mode no proporciona cap mecanisme directe per obtenir les dades de les cartes adquirides mitjançant sobres, recompenses o el mercat de transferències.

Davant aquesta limitació, es va dur a terme una investigació extensa que confirmà la impossibilitat de recuperar aquesta informació de manera programàtica o senzilla. Això ens dugué a pensar altres alternatives. Una alternativa era seleccionar les cartes manualment una a una, opció que quedà descartada immediatament a causa de la complexitat i cost de temps innecessari. Una altre opció era realitzar un mètode més automàtic, en el que l'usuari hagues de realitzar la menor feina possible. Així que es va optar per aquesta darrera opció, però encara no sabiem la manera. La clau del projecte era trobar alguna base de dades amb tota la informació de totes les cartes del joc, ja que això ens facilitaria la funcionalitat. Una vagada vam saber que es podia obtenir una base de dades sòlida, sols ens faltava com podríem conectar les cartes del nostre propi club amb la base de dades. La solució es va basar en la captura d'imatges directament des del joc per poder extreure informació i conectar-la amb la base de dades. No obstant això, aquest procés manual resultava molt costós i repetitiu per a l'usuari, per aquest motiu, es va desenvolupar un programa per automatitzar el procés de les captures.

4.1 Automatització de captures de pantalla

En un primer intent per automatitzar completament aquest procés, es desenvolupà una versió inicial del programa utilitzant les llibreries ImageGrab i PyAutoGUI. L'objectiu era capturar imatges i, simultàniament, navegar automàticament per les pàgines del joc mitjançant simulació de pulsacions de teclat. Malauradament, el joc bloqueja les interaccions generades per PyAutoGUI, impedint el control del teclat i, per tant, la navegació dins la interfície del joc.

Davant aquesta limitació, es reformulà l'estrategia, centrant-se en automatitzar

4. OBTENCIÓ DE JUGADORS

exclusivament les captures de pantalla, ajudant en gran part a l'usuari. Així, es desenvolupà un script en Python que realitza captures automàtiques a intervals regulars, emprant la llibreria ImageGrab de PIL. A més, s'incorporà un senyal acústic mitjançant winsound per notificar a l'usuari quan la captura ha estat realitzada, permetent així avançar manualment a la següent pàgina de cartes.



Figura 4.1: Exemple de pantalla amb les cartes del club.

D'aquesta manera, ja hem obtingut les captures on podem trobar totes les cartes del nostre club, però encara queda obtenir la informació de cada carta de forma separada.

4.2 Retalls dels elements importants i obtenció de dades

Una vegada arribats en aquest punt, la idea principal consistia aaprofitar els elements visibles de cada carta —com ara la valoració global, el nom del jugador, la posició principal, les estadístiques, la nacionalitat, la lliga, el club i el tipus de carta— per vincular aquesta informació amb la base de dades del sistema, des d'on es poden obtenir dades addicionals com el preu actual de la carta, les posicions alternatives o el gènere del jugador, aspectes especialment rellevants per a la realització de l'algorisme genètic.

L'extracció d'alguns d'aquests atributs, com la nacionalitat, la lliga, el club o la versió de la carta, suposava una dificultat considerable, ja que la forma més eficient d'obtenir-los era mitjançant tècniques de visió per computador. No obstant això, la creació d'un conjunt de dades adequat per entrenar un model de detecció com YOLO resultava una tasca força complexa i laboriosa.

En canvi, l'extracció de la resta d'elements —com el nom, la posició o les estadístiques numèriques— era relativament senzilla. Mitjançant el retall de zones específiques de les cartes i l'ús de tècniques de reconeixement òptic de caràcters (OCR), era possible obtenir aquests valors de manera fiable i automatitzar la seva vinculació amb la base de dades.

Inicialment, l'objectiu era capturar el màxim nombre possible d'elements mitjançant retalls previs de les cartes i aplicar-hi OCR per extreure les dades corresponents.

4.2. Retalls dels elements importants i obtenció de dades

Així que, aquest procés va tenir dues fases, la fase inicial i la fase definitiva, on en aquesta darrera és una forma més optimitzada de la primera fase.

4.2.1 Fase Inicial

Ens vam adonar que les cartes es mostraven sempre amb una disposició fixa: cinc a la fila superior i cinc a la inferior. Aquesta estructura consistent permetia determinar amb precisió la posició de cada carta a la pantalla, de manera que la solució més simple i eficient per generar els retalls era fer-ho de forma estàtica, aprofitant aquestes coordenades fixes.

Retalls per pixels

Un cop obtingudes totes les captures de pantalla, emmagatzemades dins d'una carpeta específica, es va desenvolupar un programa encarregat de realitzar retalls en les posicions concretes corresponents a cada carta mostrada. L'objectiu d'aquest procés era extreure, de forma automatitzada, totes les cartes visibles en cada captura i guardar-les en una nova carpeta organitzada.

Per dur a terme els retalls amb precisió, primerament es van definir les dimensions de cada carta en píxels, concretament l'amplada i l'alçada. Aquestes constants es mostren en la següent taula:

Constants	Valor
Amplada (W)	339
Alçada (H)	450

Taula 4.1: Constants de la carta (Amplitud i alçada)

Amb aquestes dimensions com a referència, es van calcular les coordenades (X, Y) de cadascuna de les deu cartes mostrades en pantalla. Com ja hem comentat anteriorment, la disposició seguia un patró fix amb cinc cartes en la fila superior i cinc en la inferior, totes equidistants entre elles.

Les coordenades de tall s'estableixen en la taula següent, utilitzant una fórmula paramètrica basada en la posició relativa horitzontal (X) i una altura comuna per fila (Y):

4. OBTENCIÓ DE JUGADORS

Carta	X	Y
1	760	175
2	760 + 339	175
3	760 + (2 × 339)	175
4	760 + (3 × 339)	175
5	760 + (4 × 339)	175
6	760	689
7	760 + 339	689
8	760 + (2 × 339)	689
9	760 + (3 × 339)	689
10	760 + (4 × 339)	689

Taula 4.2: Coordenades de les cartes dins la captura de pantalla.

Aquest enfocament basat en retalls estàtics permet extreure de manera fiable totes les cartes de cada captura, assegurant una consistència espacial que simplifica considerablement el procés de lectura i processament d'imatges posterior.

Un cop realitzat el retall de cada carta individual, es procedeix a l'extracció específica dels elements d'interès. Atès que la base de dades no inclou informació relativa a les estadístiques detallades dels jugadors (com el ritme, xut, passada, etc.), no és necessari obtenir retalls d'aquests valors concrets.

Tot i que inicialment es van implementar retalls per a la lliga, la nacionalitat i el club, aquestes dades no es van acabar utilitzant, ja que no es disposa d'un sistema robust de reconeixement d'imatges per identificar correctament aquests elements gràfics. Per tant, encara que realitzem els retalls de les lligues, nacionalitats i clubs, el sistema es centrarà únicament en l'extracció de dues informacions fonamentals: la puntuació global (valoració mitjana del jugador) i el nom.

La Figura 4.2 mostra la localització gràfica d'aquests elements dins d'una carta típica:



Figura 4.2: Indicacions dels elements de la carta.

Les coordenades relatives (en píxels) dins de cada carta per a cadascun dels elements retallats es recullen a la següent taula:

4.2. Retalls dels elements importants i obtenció de dades

Element	X	Y	Amplada (W)	Alçada (H)
Puntuació	41	76	55	40
Nom del jugador	26	303	285	40
Lliga	152	383	40	40
Nacionalitat	112	390	40	30
Club	190	385	40	40

Taula 4.3: Coordenades relatives dins de cada carta per a l'extracció dels diferents elements.

A continuació es mostren exemples visuals dels retalls obtinguts per a cada element:

Element	Exemple d'imatge
Puntuació	
Nom del jugador	
Lliga	
Nacionalitat	
Club	

Taula 4.4: Exemples dels retalls dels diferents elements de cada carta.

Un cop obtinguts aquests retalls, s'aplica un sistema de reconeixement òptic de caràcters (OCR) per extreure automàticament el nom del jugador i la seva puntuació. Aquesta informació es la que ens permetrà vincular la carta a la base de dades per facilitar la generació de plantilles per a SBC de manera automàtica i eficient.

Bases de dades utilitzades

Abans d'iniciar el procés d'extracció d'informació és imprescindible conèixer l'estructura dels conjunts de dades disponibles i les transformacions prèvies que cal aplicar-hi. En aquest treball s'han seleccionat dos datasets complementaris, allotjats a la plataforma Kaggle, que cobreixen necessitats informatives diferents.

4. OBTENCIÓ DE JUGADORS

1. EA FC 24 FUT Players Dataset Aquest conjunt de dades (*EA FC 24 FUT Players Dataset*) conté una entrada per a cada carta disponible al mode Ultimate Team i inclou informació econòmica de mercat, versió de la carta i estadístiques bàsiques del jugador:

Columna	Descripció
Name	Nom complet del jugador
Ratings	Valoració global (OVR)
Position	Posició principal (ST, CM, CB, ...)
Version	Tipus de carta (Gold, TOTW, FUT Birthday, ...)
Price	Preu actual a PlayStation (0 si no és present al mercat)
SKI	Habilitats tècniques (0 – 5 estrelles)
WF	Cama menys hàbil (0 – 5 estrelles)
WR	Ritme de treball Atac/Defensa (Low, Medium, High)
PAC-PHY	Estadístiques bàsiques: PAC, SHO, PAS, DRI, DEF, PHY
Body	Alçada i tipus de cos
Popularity	Índex d'ús del jugador al joc
BS, IGS	Estadístiques base i en joc, respectivament

Taula 4.5: Columnes principals del *EA FC 24 FUT Players Dataset*

2. EA Sports FC 24 Complete Player Dataset Per completar els atributs faltants es recorre al dataset *EA Sports FC 24 Complete Player Dataset*, que agrega la trajectòria de jugadors (homes i dones) des de *FIFA 15* fins a *EA Sports FC 24*. Només s'han importat els fitxers corresponents a la temporada 2024, on trobem 109 columnes; les més rellevants són:

Columna	Descripció
fifa_version	Versió del joc (<i>EA Sports FC 24</i>)
short_name	Nom curt del jugador
long_name	Nom complet del jugador
club_name	Club al qual pertany
league_name	Lliga on juga el club
nationality_name	Nacionalitat del jugador

Taula 4.6: Columnes principals del *EA Sports FC 24 Complete Player Dataset*

A causa de la magnitud del *Complete Player Dataset* i del fet que inclou registres de totes les edicions del joc —des de *FIFA 15* fins a *EA Sports FC 24*— es va dur a terme un pas previ de depuració amb RStudio. En aquesta fase es va filtrar el conjunt de dades per conservar únicament els jugadors corresponents a l'edició 2024; concretament, es van seleccionar els registres en què la columna `fifa_version` pren el valor 24. D'aquesta manera s'assegura que la informació disponible sigui coherent amb la temporada que es modela en aquest treball.

3. Complementarietat dels datasets El *FUT Players Dataset* aporta informació crucial sobre el valor de mercat i les diferents versions de carta, mentre que el *Complete*

4.2. Retalls dels elements importants i obtenció de dades

Player Dataset proporciona metades esportives —club, lliga i nacionalitat— imprescindibles per satisfer els requisits de química d'una *SBC*. En conseqüència, tots dos fitxers es fan servir per poder obtenir els atributs necessaris. El procediment d'integració i depuració es descriu detalladament a la Secció següent.

Extracció d'informació

Un cop definida l'estructura de les bases de dades, cal descriure el procediment que vincula cada carta capturada amb els seus atributs. El procés es divideix en quatre fases: (i) lectura d'imatges amb OCR, (ii) cerca al *FUT Players Dataset*, (iii) obtenció de metades al *Complete Player Dataset* i (iv) construcció de l'objecte Jugador.

i. Lectura OCR Per a cada carta es disposa de dos retalls —nom i valoració global— que es processen amb EasyOCR. El valor numèric es converteix a enter; si la conversió falla, la carta es descarta.

ii. Identificació al *FUT Players Dataset* Ens vam adonar que el nom obtingut *in situ* sovint no coincideix literalment amb el registre de la taula *players*. Per això:

1. es descompon el nom en paraules i es construeix una consulta LIKE que exigeix la presència de totes les paraules i la mateixa valoració;
2. si la consulta retorna diversos candidats, se selecciona aquell amb menor distància de *Levenshtein* respecte del nom original;
3. si no hi ha coincidències, s'intenta una normalització bàsica (*p. ex.* reemplaçar “ii” per “ü”) i es repeteix el procés. Això es produeix per una errada del OCR, ja que detecta com a “ii” les lletres “ü”, per tant es realitzen aquests canvis.

El resultat inclou, entre d'altres, les posicions jugables, el tipus de carta i el preu en el mercat de PlayStation.

iii. Obtenció de club, lliga i nacionalitat Amb el nom confirmat es fa una segona consulta al *Complete Player Dataset*. L'ordre de recerca és:

1. taula `male_players`, columna `long_name`;
2. taula `female_players`, columna `long_name`;
3. taula `male_players`, columna `short_name`;
4. taula `female_players`, columna `short_name`.

En cada pas s'utilitza la mateixa estratègia de coincidència parcial i selecció per distància de *Levenshtein*. D'aquesta consulta s'extrauen la nacionalitat, la lliga i el club.

4. OBTENCIÓ DE JUGADORS

Algorisme de Levenshtein La *distància de Levenshtein* (1) mesura la similitud entre dues cadenes x i y com el nombre mínim d'operacions —inserció, eliminació o substitució d'un caràcter— necessàries per transformar x en y . S'aplica una recursió de programació dinàmica sobre una matriu D de dimensions $(m+1) \times (n+1)$. El cost temporal és $O(mn)$ i l'espacial $O(mn)$, tot i que es pot reduir la memòria a $O(\min\{m, n\})$.

En aquest projecte s'utilitza per triar el jugador de la base de dades amb nom més proper quan l'OCR no retorna una coincidència exacta.

iv. Construcció de l'objecte Jugador Ambdós blocs d'informació es combinen en una instància de la classe Jugador, definida amb els atributs principals que es mostren a la Taula 4.7. Cada instància s'afegeix a una llista i, finalment, es serialitza a JSON per alimentar l'algorisme genètic de generació de plantilles.

Atribut	Descripció
nombre	Nom del jugador
puntuacion	Valoració global
nacionalidad	Nacionalitat (segons <i>Complete Player Dataset</i>)
liga	Lliga on juga el club (segons <i>Complete Player Dataset</i>)
club	Club d'adscripció (segons <i>Complete Player Dataset</i>)
positions	Posicions jugables de la carta
card_type	Versió de la carta (Gold, TOTW, ...)
price	Preu de mercat a PlayStation ($k \rightarrow$ milers)
image	Nom del fitxer de la imatge de la carta

Taula 4.7: Estructura de la classe Jugador

Aplicat a totes les cartes del club, el procés genera un arxiu jugadores.json que recull tota la informació necessària per a les fases posteriors de l'estudi.

Exemple d'un objecte Jugador

```
[
  {
    "nombre": "Harry Kane",
    "puntuacion": 90,
    "nacionalidad": "England",
    "liga": "Bundesliga",
    "club": "FC Bayern München",
    "positions": "ST,CF",
    "card_type": "normal Mostly Lengthy",
    "price": 57500.0,
    "image": "card_1.png"
  }
]
```

4.2.2 Fase definitiva

El procediment inicial presentava dues limitacions crítiques. D'una banda, els retalls estàtics fallaven quan la resolució del monitor variava i només permetien capturar cartes des d'una ubicació molt concreta de la interfície. De l'altra, l'obtenció de la informació completa dels jugadors exigia un nombre excessiu de consultes a la base de dades, fet que penalitzava el rendiment global.

Per resoldre aquests problemes es va redissenyar íntegrament els processos anteriors:

- a) **Retalls dinàmics** mitjançant tècniques de visió per computador, substituint els retalls per coordenades fixes.
- b) **Unificació de la informació** en una única base de dades que conté tots els camps necessaris, evitant consultes creuades.
- c) **Optimització de la cerca de jugadors**, reduint al mínim el nombre de consultes i millorant la latència del sistema.

Retalls amb tècniques de visió per computador

La metodologia de retall estàtic resultava poc flexible, ja que depenia d'una resolució i d'un emplaçament fix de les cartes. Per solucionar-ho ens vam proposar desenvolupar un mètode capaç de localitzar les cartes en pantalla amb qualsevol resolució i disposició. Aquesta necessitat ens va portar a considerar tècniques de visió per computador i, concretament, a adoptar el detector d'objectes YOLO.

YOLO (You Only Look Once) YOLO és una família de xarxes convolucionals per a detecció d'objectes en temps real. A diferència dels enfocaments basats en dues etapes (p. ex. R-CNN), YOLO tracta la imatge d'entrada com una sola unitat (“only look once”) i, en una passada, divideix la imatge en cel·les que prediuen tant la classe com la capsula delimitadora dels objectes. Això li permet aconseguir velocitats superiors als 30 FPS amb una precisió competitiva, la qual cosa el fa especialment adequat per detectar cada carta —independentment de la resolució o el layout— abans d'aplicar els retalls i l'OCR corresponents.

Els pesos públics de YOLO estan entrenats principalment amb el conjunt de dades COCO, que conté 80 classes genèriques i no inclou cartes d'Ultimate Team. Per aquest motiu cal *afinar* (fine-tune) el model amb un conjunt d'exemples etiquetats específics. Entrenament per a detecció de cartes completes (classe única *card*) i entrenament per a detecció d'elements interns de la carta (classes *overall*, *name*, *pace*, *shooting*, *defending*, ...).

El flux proposat és, per tant, bifàsic: (1) detectar totes les cartes de la imatge i retallar-les; (2) aplicar el segon model sobre cada retall per localitzar els elements interns i, finalment, (3) executar l'OCR només on cal.

Aquesta estratègia separa clarament el problema macroscòpic (delimitar cartes) del microscòpic (identificar components), redueix la complexitat de cada model i millora la robustesa del sistema davant canvis de resolució o de disposició de la interfície.

4. OBTENCIÓ DE JUGADORS

A continuació us explicarem com es va realitzar l'entrenament per a la detecció de cartes:

- **Entrenament Cartes:** primer de tot, cal fer una recopilació d'imatges que ens servirà per entrenar el model. Algunes de les imatges elegides son d'aquest tipus:

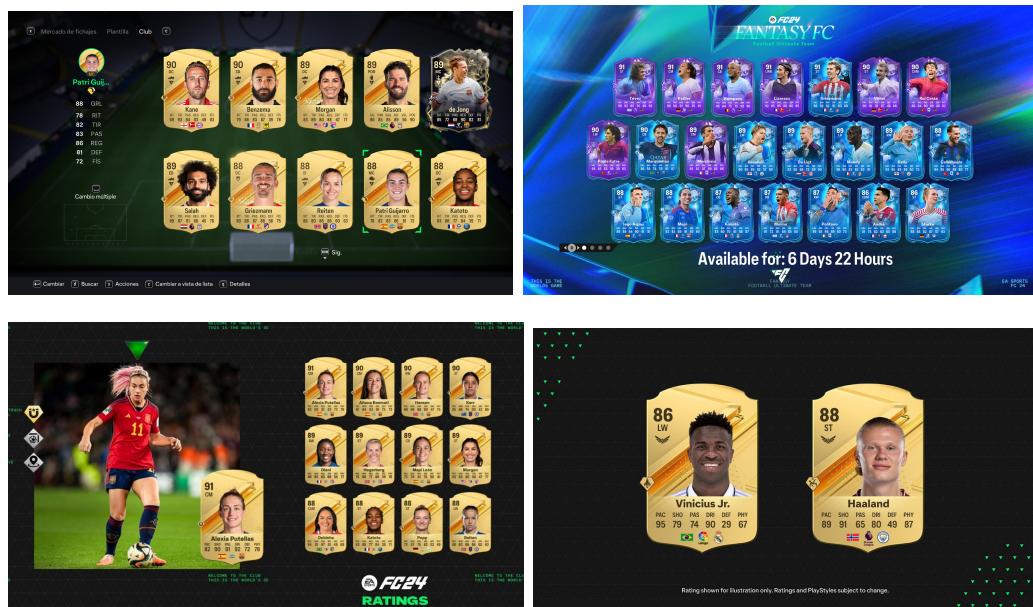


Figura 4.3: Recopilació d'algunes de les imatges utilitzades per entrenar YOLO

Un cop recopilades entre 50 i 100 imatges representatives, cal anotar la ubicació de cada carta. Aquesta tasca es fa amb *Label Studio*: es defineix una única classe, card, i es dibuixen les *bounding boxes* corresponents a totes les cartes de cada imatge. Tot i que el procés és laboriós—cal etiquetar manualment cada fotografia—és imprescindible per entrenar el detector.

Quan l'anotació és completa, s'exporta el projecte en format YOLO (les imatges i els fitxers .txt amb les etiquetes). A continuació:

- Es puja l'arxiu exportat a *Google Colab*.
- Es divideix el conjunt en carpetes `images/train`, `images/val` i, si cal, `images/test`.
- Es crea el fitxer `data.yaml`, indicant les rutes a les carpetes de `train/val` i el nombre de classes (`nc: 1; names: [card]`).
- Finalment, s'executen les ordres `yolo train` i `yolo val` per entrenar i avaluar el model.

A continuació podem veure els resultats d'aquest entrenament:

4.2. Retalls dels elements importants i obtenció de dades

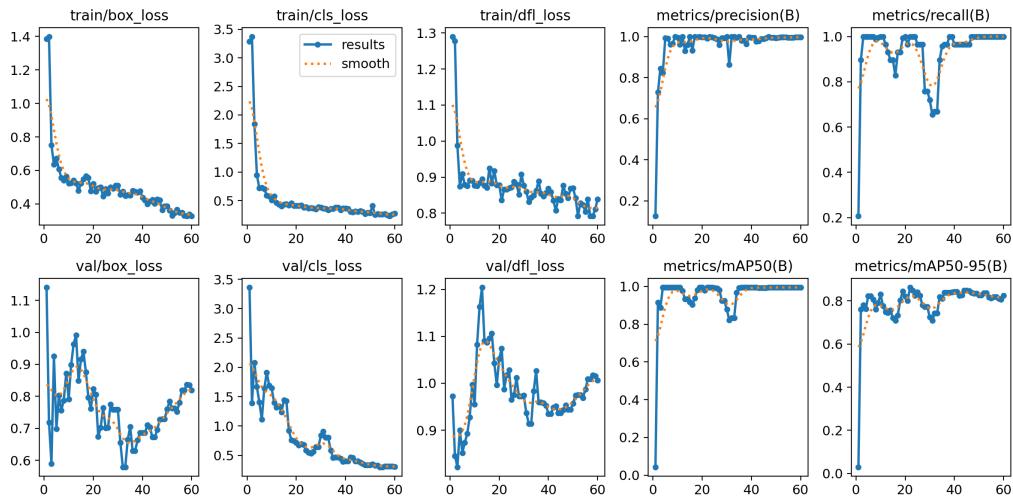


Figura 4.4: Resultats del test.

La Figura 4.4 mostra l’evolució de les pèrdues (*losses*) i mètriques durant 60 èpoques d’entrenament. A continuació se’n presenta una síntesi i es discuteixen les implicacions.

Quadre	Tendència observada
train/{box, dfl}_loss	cls, Descens ràpid fins a l’ <i>epoch</i> 10, després disminució suau; no hi ha símptomes d’ <i>underfitting</i> .
val/{box, dfl}_loss	cls, Volatilitat inicial; mínim entorn de l’ <i>epoch</i> 25–30; lleugera pujada final ⇒ inici d’ <i>overfitting</i> .
precision(B)	≥ 0.95 a partir de l’ <i>epoch</i> 10; es manté estable ≈ 1.0 .
recall(B)	Oscil·lacions 0.70–0.95 fins l’ <i>epoch</i> 35; estable en 0.98–1.00 al tram final.
mAP50(B)	Puja a > 0.90 molt aviat; recupera ràpidament una caiguda puntual (<i>epoch</i> 30); finalitza a 0.99.
mAP50–95(B)	Es manté entre 0.75–0.85; tanca entorn de 0.80.

Taula 4.8: Tendències principals observades a les corbes d’entrenament

- **Convergència:** les pèrdues de *training* mostren una disminució consistent ⇒ aprenentatge estable.
- **Sobre-ajustament (overfitting):** l’augment progressiu de les pèrdues de validació a partir de l’*epoch* 40 indica que el model comença a memoritzar el conjunt d’entrenament.
- **Precisió vs. revocació:** precisió elevada (≈ 1.0) amb revocació lleugerament inferior al començament; la revocació s’estabilitza a 0.98–1.00 en les darreres èpoques.
- **mAP:** *mAP@0.5* quasi perfecte (0.99); *mAP@0.5–0.95* al voltant de 0.80 ⇒ la localització fina encara pot millorar.

4. OBTENCIÓ DE JUGADORS

1. **Aturar l'entrenament abans** (*early stopping*) cap a l'*epoch* 40 per reduir l'*overfitting*.
2. **Augmentació de dades:** rotacions $\leq 5^\circ$, variacions d'HSV i mosaic per millorar generalització.
3. **Millorar mAP50–95:** entrenar amb resolució superior (*imgsz*=1024) i aplicar augmentació *copy-paste*.
4. **Robustesa:** afegir captures amb il·luminació diferent i cartes parcialment ocultes.
5. **Exportar el millor pes** (segons *val/mAP50–95*) a ONNX o TensorRT per a inferència en producció.

El model assolix **precisió** ≈ 1.0 i **mAP50** ≈ 0.99 , fet que el fa molt fiable per detectar cartes. Resta marge de millora en la localització estricta (mAP50–95) i en la reducció de l'*overfitting*. Amb les accions proposades, s'espera aconseguir un rendiment òptim per a l'aplicació final.

Algunes de les deteccions son les següents:

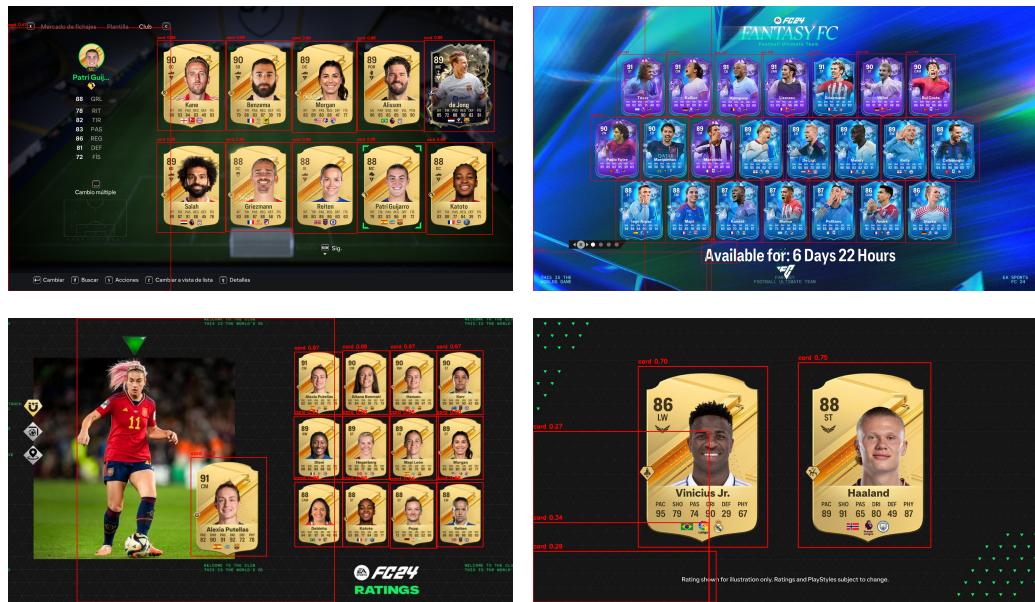


Figura 4.5: Recopilació d'algunes de les deteccions de les imatges de la Figura 4.3

Es pot observar que el model detecta correctament les cartes, si bé apareixen algunes deteccions espúries amb una confiança clarament inferior. Per descartar-les, podem fixar un llindar mínim de probabilitat; concretament, només es retallaran aquells objectes amb una confiança $\geq 0,70$, atès que totes les cartes vàlides superen aquest valor.

- **Entrenament Elements:** el procés és el mateix, ja que es realitza una recopilació d'imatges per a poder entrenar el model, es seleccionen les bounding boxes de cada imatge, s'exporten els resultats i s'entrena el model.

4.2. Retalls dels elements importants i obtenció de dades

Les imatges seleccionades son les deteccions que s'han realitzat amb el model anterior. Així que s'han recopilat totes les cartes de les imatges anteriors per a poder obtenir un dataset complet. Algunes de les imatges son aquestes:

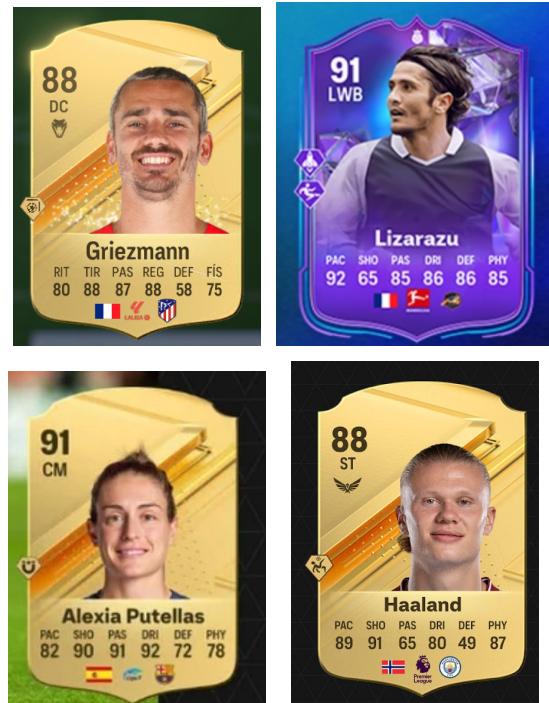


Figura 4.6: Recopilació d'algunes de les imatges utilitzades per entrenar el segon model de YOLO

Com ja hem comentat anteriorment, el procés és el mateix, quan ja hem fet la recopilació de les imatges anteriors, cal anotar la ubicació de cada element en la carta mitjançant *Label Studio*. Les classes que s'han implementat son *Defending*, *Dribbling*, *Name*, *Overall*, *Pace*, *Passing*, *Phisicallity*, *Position* and *Shooting*. Una vegada hem indicat totes les bounding boxes, extreim els resultats i realitzam el següen procés:

- Es puja l'arxiu exportat a *Google Colab*.
- Es divideix el conjunt en carpetes `images/train`, `images/val` i, si cal, `images/test`.
- Es crea el fitxer `data.yaml`, indicant les rutes a les carpetes de `train/val` i el nombre de classes (nc: 9; names: [Defending, Dribbling, Name, Overall, Pace, Passing, Phisicallity, Position and Shooting]).
- Finalment, s'executen les ordres `yolo train` i `yolo val` per entrenar i avaluar el model.

A copntinuació podem veure els resultats d'aquest entrenament:

4. OBTENCIÓ DE JUGADORS

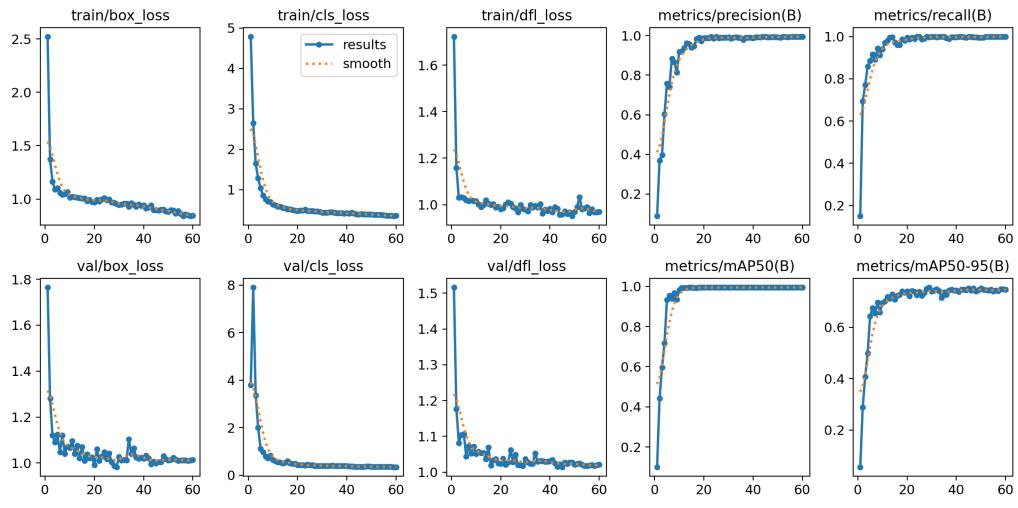


Figura 4.7: Resultats del test.

La Figura 4.7 mostra corbes de pèrdua i mètriques durant 60 èpoques. Les tendències principals s'han sintetitzat a la Taula 4.9.

Quadre	Tendència observada
train/{box, cls, dfl}_loss	Descens molt ràpid (èpoques 0–5) i estabilització entorn de 0.8–0.9.
val/{box, cls, dfl}_loss	Corbes paral·leles a les de <i>train</i> ; no hi ha repunt final ⇒ sense <i>overfitting</i> .
precision(B)	S'enfila a ≥ 0.95 en menys de 10 èpoques i assoleix 1.00.
recall(B)	Puja a 0.97–0.99 i s'estabilitza.
mAP50(B)	Arriba a 0.99 molt aviat i es manté.
mAP50–95(B)	Convergència lenta fins a 0.70–0.73.

Taula 4.9: Resum de l'entrenament mostrat a la Fig. 4.7

- **Convergència estable:** pèrdues de *train* i *val* disminueixen de forma semblant ⇒ bon ajustament.
- **Precisió vs. revocació:** precisió pràcticament perfecta (≈ 1.0) i revocació > 0.97 .
- **mAP:** $mAP@0.5 \ 0.99$; $mAP@0.5–0.95 \ 0.72 \Rightarrow$ millorable la localització fina.
 1. **Augmentació avançada:** rotacions $\leq 5^\circ$, *copy-paste*, variacions d'HSV i mosaic per refinar les capses.
 2. **Resolució superior:** entrenar amb `imgsz=1024` per millorar la qualitat del *bounding box*.
 3. **Auto-anchors:** recalcular ancoratges si la relació d'aspecte de les cartes és específica.
 4. **Exemples difícils:** afegir cartes parcialment tapades o amb il·luminació adversa.

4.2. Retalls dels elements importants i obtenció de dades

En conjunt, el model presenta **precisió i revocació** $\simeq 1.0$ i **mAP50** $\simeq 0.99$, la qual cosa el fa molt fiable per detectar els elements de les cartes. El marge de millora se centra en la localització estricta ($map@0.5-0.95$) i en l'afinament de caps per a tasques d'OCR de camp petit.

Aquí podem veure un exemple de les deteccions realitzades:

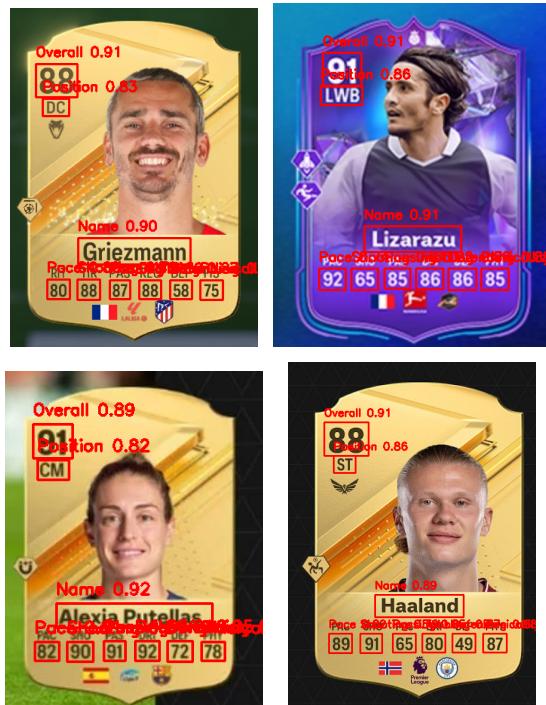


Figura 4.8: Recopilació d'algunes de les deteccions de les imatges de la Figura 4.6

Podem veure una bona detecció dels elements de la carta, la qual ens permet fer els retalls per aplicarli l'OCR

Base de dades

Tal com s'ha comentat, un dels canvis necessaris per optimitzar les cerques fou substituir la base de dades inicial per una font més completa i actual. A aquest efecte es va emprar la base de dades de *FUTBIN*, pàgina especialitzada en *EA FC 24*—i, especialment, en el mode Ultimate Team—que es manté actualitzada diàriament i conté tots els camps rellevants.

Abans d'integrar-la, es va dur a terme un preprocessament orientat a distingir jugadors i jugadores:

- Es va identificar la llista de lligues femenines presents al joc: *Liga F, Barclays WSL, NWSL, GPFBL* i *D1 Arkema*.
- Es va afegir una nova columna gender.
 - Si el jugador pertanyia a alguna de les lligues anteriors $\Rightarrow F$.

4. OBTENCIÓ DE JUGADORS

- Si la lliga era *Icon* \Rightarrow U (gènere indeterminat).
 - En qualsevol altre cas \Rightarrow M.
- c) Per diferenciar clubs amb secció masculina i femenina es va afegir la lletra F al final del nom del club femení (excepte per a la lliga *Hero*, on no és necessari).

Amb aquest preprocessament, la base de dades queda coherent amb els requisits del projecte i preparada per a consultes eficients.

L'estructura final de la base de dades és aquesta:

Columna	Descripció
name	Nom complet del jugador/a
rating	Valoració global (<i>OVR</i>)
position	Posició principal (<i>ST, CM, CB...</i>)
version	Tipus de carta (<i>Gold, TOTW, FUT Birthday...</i>)
price	Preu actual a PlayStation (0 si no és present al mercat)
club	Club d'adscripció (amb sufix F si és equip femení)
nationality	Nacionalitat del jugador/a
league	Lliga on juga el club
skills	Habilitats tècniques (0–5 estrelles)
weak_foot	Cama menys hàbil (0–5 estrelles)
foot	Cama dominant (<i>Right/Left</i>)
pace-physicality	Estadístiques bàsiques: <i>pace, shooting, passing, dribbling, defending, physicality</i>
body	Alçada i tipus de cos
gender	Gènere: M (male), F (female) o U (unisex)

Taula 4.10: Columnes principals de la base de dades procedent de *FUTBIN*

Extracció d'informació

Es processa cada captura mitjançant el **model 1** (detector de cartes). Sobre cada retall resultant s'aplica el **model 2** (detector d'elements interns) per localitzar els components de la carta. Un cop obtinguts els retalls de *name, overall, position, pace, shooting, passing, dribbling, defending* i *physicality*, s'executa EasyOCR.

En ocasions l'OCR retorna text buit; això no és crític, ja que amb la resta de camps és possible identificar el jugador. El flux de consulta és:

- a) Cerca primària a la base *FUTBIN* pel parell {*name, rating*}.
- b) Si no hi ha coincidència, es realitza una cerca secundària utilitzant les estadístiques (*pace, shooting,...*) per reduir l'espai de cerca.

Cal tenir present una casuística habitual: un jugador pot començar la temporada en un club i, després d'un traspàs a mitja campanya, rebre una nova carta amb les mateixes estadístiques però associada al nou equip. Això fa que la base de dades contingui dues cartes idèntiques en valors, però amb clubs diferents, i la cerca pot

4.2. Retalls dels elements importants i obtenció de dades

retornar diversos resultats. Actualment l'algoritme selecciona la primera coincidència; en versions futures es permetrà a l'usuari triar manualment la carta correcta quan es detectin duplicats.

Un altre escenari especial són les cartes *evolucionades*: a partir d'una carta bàsica, el joc permet millorar-la mitjançant reptes puntuals, i aquestes versions no figuren a la base de dades. En aquests casos, es conserven els valors obtinguts via OCR i s'ofereix a l'usuari la possibilitat de completar els camps restants. La incidència és baixa, perquè el nombre de cartes susceptibles d'evolució és limitat.

La informació consolidada es desa en instàncies de la classe Jugador:

Atribut	Descripció
name	Nom complet del jugador
rating	Valoració global (OVR)
nationality	Nacionalitat
league	Lliga on juga el club
club	Club d'adscripció
position	Posicions jugables (ST, CF...)
version	Tipus de carta (Gold, TOTW, ...)
price	Preu de mercat a PlayStation (monedes)
pace – physicality	Estadístiques bàsiques de la carta
foot	Cama dominant (<i>Right/Left</i>)
weak_foot	Cama menys hàbil (0–5)
skills	Estrelles d'habilitats (0–5)
body	Alçada i tipus de cos
gender	M, F o U
image	Nom del fitxer de la imatge de la carta

Taula 4.11: Estructura de la classe Jugador

Totes les instàncies es serialitzen en un fitxer jugadores.json, que serveix d'entrada per a l'algorisme genètic.

Exemple d'un objecte Jugador

```
[  
  {  
    "name": "Harry Kane",  
    "rating": 90,  
    "nationality": "England",  
    "league": "Bundesliga",  
    "club": "FC Bayern München",  
    "position": "ST,CF",  
    "version": "normal",  
    "price": 10750,  
    "pace": 69,  
    "shooting": 93,  
    "passing": 84,
```

4. OBTENCIÓ DE JUGADORS

```
        "dribbling": 83,  
        "defending": 49,  
        "phisicality": 83,  
        "foot": "Right",  
        "weakFoot": 5,  
        "skills": 3,  
        "body": "188cm | 6'2 | Mostly Lengthy",  
        "gender": "M",  
        "image": "card_3.png"  
    }  
]
```

Aquesta fase definitiva ens ha permès disposar de totes les dades necessàries per implementar l'algorisme genètic.

GENERACIÓ DE PLANTILLES

Un cop disposam del conjunt complet de cartes extretes i etiquetades, podem abordar la generació automàtica de plantilles. Cada SBC imposa un seguit de requisits—mitjana d'equip mínima o màxima, química global, nombre de nacionalitats, clubs concrets o versions de carta específiques—que la plantilla ha de satisfer de manera simultània. Definits aquests condicionants, iniciam el procés d'optimització mitjançant l'algorisme genètic, amb l'objectiu de trobar la combinació de cartes que compleixi tots els criteris al menor cost possible.

Primer de tot cal saber que és un algoritme genètic:

5.0.1 Algorisme genètic: descripció general

Un **algorisme genètic** (AG) és una tècnica d'optimització inspirada en els principis de l'evolució biològica. Parteix d'una *població* de solicions candidates i, mitjançant operadors de selecció, creuament i mutació, en genera successives generacions amb l'objectiu de maximitzar (o minimitzar) una *funció de fitness*. El flux bàsic és:

1. **Inicialització** Es crea una població inicial P_0 amb N individus generats aleatoriament o a partir d'una heurística.
2. **Avaluació de fitness** Cada individu i s'avalua amb una funció $f(i)$ que mesura la qualitat de la solució (p. ex. cost total de la plantilla i compliment dels requisits de l'SBC).
3. **Selecció** Es trien pares per reproduir-se, afavorint els individus amb millor fitness. Mètodes habituals: *roulette-wheel*, *tournament* o *rank*.
4. **Creuament (crossover)** A partir de dos pares seleccionats es crea una o més descendències combinant els seus gens (p. ex. intercanvi de subconjunts de cartes).

5. GENERACIÓ DE PLANTILLES

5. **Mutació** Amb una petita probabilitat p_{mut} s'altera aleatoriament part del genotip de l'individu (substituir una carta per una altra) per introduir diversitat i evitar òptims locals.
6. **Reemplaçament** Es construeix la nova població P_{t+1} a partir dels fills (i, opcionalment, d'alguns individus elit de P_t).
7. **Criteri d'aturada** El procés es repeteix fins que s'assoleix un nombre màxim de generacions G o no hi ha millora significativa en el fitness.

Paràmetres clau

- Tamany de població N
- Probabilitat de creuament p_{cx} i de mutació p_{mut}
- Percentatge d'*elitisme* (individus amb millor fitness que passen directament a la següent generació)
- Funció de fitness, que en el nostre cas penalitza el cost i els incompliments de requisits

Els AG són especialment útils quan l'espai de cerca és gran i la funció objectiu és no lineal o conté múltiples òptims locals, com ocorre en la generació de plantilles òptimes per a les SBC.

Primer de tot, cal fer una definició de com s'estructurarán els SBC:

5.1 Definició de requisits per a totes les SBC

Per gestionar de manera estructurada els requisits de cada SBC hem adoptat un fitxer JSON. Cada objecte descriu un repte concret i conté:

- **Metadades:** nom del repte, formació, llista de posicions i nom de la imatge.
- **Requisits numèrics:** mitjana d'equip i química (mínima i/o màxima).
- **Requisits qualitatius:** nombre mínim o màxim de nacionalitats, clubs i versions de carta.

A continuació es mostra un exemple abreujat:

```
{  
  "teams": [  
    {  
      "name": "[name_team]",  
      "formation": "4-4-2",  
      "image": "[image_name]",  
      "positions": [  
        "ST", "ST", "LM", "CM", "CM", "RM", "LB", "CB", "CB", "RB", "GK"  
      ],  
    },  
  ]},
```

5.1. Definició de requisits per a totes les SBC

```
"requirements": {
    "average": {
        "min": 88,
        "max": 90
    },
    "chemistry": {
        "min": 10,
        "max": 20
    },
    "nationalities": {
        "min": [
            {
                "name": "Spain",
                "number": 1
            }
        ],
        "max": [
            {
                "name": "France",
                "number": 4
            }
        ]
    },
    "clubs": {
        "min": [
            {
                "name": "FC Barcelona",
                "number": 1
            }
        ]
    },
    "versions": {
        "min": [
            {
                "name": "Thunderstruck",
                "number": 1
            }
        ]
    }
}
```

5.2 Algoritme evolutiu

A continuació es detalla l'algoritme evolutiu, estructurat de la manera següent:

1. Obtenció dels requisits i de la població inicial (vegeu Secció 5.2.3).
2. Repetició durant `NUM_GENERATIONS` iteracions:
 - a) Avaluació de la població mitjançant el càlcul de la informació dels equips (Secció 5.2.4) i de la funció de *fitness* (Secció 5.2.5).
 - b) Aplicació de l'elitisme conservant els deu millors equips.
 - c) Bucle de `POPULATION_SIZE` iteracions per generar descendència:
 - i. Selecció de dos equips pares.
 - ii. Creuament per formar un fill (Secció 5.2.6).
 - iii. Mutació del fill (Secció 5.2.7).

5.2.1 Paràmetres bàsics

En primer lloc es declaren tres constants fonamentals:

- `NUM_GENERATIONS` – nombre d'iteracions evolutives;
- `POPULATION_SIZE` – mida de la població (equips simultanis);
- `MUTATION_RATE` – probabilitat de mutació per individu.

Els valors òptims d'aquests paràmetres s'analitzaran més endavant.

5.2.2 Entrades

A partir del fitxer `sbc.json` s'obtenen els requisits i restriccions de la SBC, a més de la informació addicional. Paral·lelament, es carrega `jugadores.json`, que conté tots els jugadors detectats al club.

5.2.3 Població inicial

Per generar la població inicial es creen equips aleatoris amb tants jugadors com posicions exigeix la SBC. L'algorisme selecciona, per a cada posició, un jugador aleatori del club sense repetir-lo dins del mateix equip (restricció d'unicitat). Es repeteix el procés fins a obtenir `POPULATION_SIZE` equips.

Una vegada obtenim la població inicial, hem de començar a avaluar-la i a generar-ne de nova. Aquesta és una de les tasques més complicades, ja que es necessita tenir coneixements sobre el funcionament dels equips en l'UT.

Quan s'avalua la població, el que feim nosaltres és recorrer tots els equips de la població, calculant la informació important de l'equip i llavors calculant la puntuació *fitness*, que es la que ens permetrà arribar a la millor solució.

5.2.4 Càlcul d'informació de l'equip

```
team_info = {
    "puntuations": {},
    "nationalities": {},
    "leagues": {},
    "clubs": {},
    "versions": {},
    "nationalities_chemistry": {},
    "leagues_chemistry": {},
    "clubs_chemistry": {},
    "overall": 0,
    "overall_rounded": 0,
    "team_price": 0,
    "team_chemistry": 0,
    "players_chemistry": [],
    "players": 0,
}
```

- **puntuations** [dictionary]: nombre de jugadors per cada valoració.
- **nationalities** [dictionary]: nombre de jugadors per cada nacionalitat.
- **leagues** [dictionary]: nombre de jugadors per cada lliga.
- **clubs** [dictionary]: nombre de jugadors per cada club.
- **versions** [dictionary]: nombre de jugadors per cada versió de carta.
- **nationalities_chemistry** [dictionary]: nombre de jugadors per cada nacionalitat que contribueixen a la química de l'equip.
- **leagues_chemistry** [dictionary]: nombre de jugadors per cada lliga que contribueixen a la química de l'equip.
- **clubs_chemistry** [dictionary]: nombre de jugadors per cada club que contribueixen a la química de l'equip.
- **overall** [integer]: mitjana total de l'equip.
- **overall_rounded** [integer]: mitjana total arrodonida de l'equip.
- **team_price** [integer]: preu total de l'equip.
- **team_chemistry** [integer]: química total de l'equip.
- **players_chemistry** [list]: química individual de cada jugador.
- **players** [integer]: nombre total de jugadors a l'equip.

5. GENERACIÓ DE PLANTILLES

A continuació es descriu com s'obté cada valor:

Els comptadors **puntuations**, **nationalities**, **leagues**, **clubs** i **versions** es construeixen recorrent l'equip: per a cada jugador, si el valor ja existeix al diccionari corresponent s'incrementa en una unitat; altrament, s'inicialitza a 1.

Pel que fa a la química, cal conèixer el funcionament específic del sistema a Ultimate Team, peça clau per avaluar la validesa de l'equip.

Química Cada jugador pot aportar entre 0 i 3 punts; la química màxima d'un XI és 33. Un jugador només pot sumar punts si ocupa una posició compatible amb la carta; en cas contrari, la seva química és 0 i no contribueix als llindars de club/lliga/nacionalitat.

La química es basa en tres atributs globals—club, lliga i nacionalitat—segons els llindars següents:

Categoría	+1 punt	+2 punts	+3 punts
Club	2 titulares	4 titulares	7 titulares
Lliga	3 titulares	5 titulares	8 titulares
Nacionalitat	2 titulares	5 titulares	8 titulares

Taula 5.1: Puntuació de la química

Punts clau:

- **Sense enllaços de posició:** la ubicació al camp és irrelevant; compta la presència dins del XI.
- **Compatibilitat de gènere:** homes i dones comparteixen química per nacionalitat i club, però no per lliga.

S'ha de tenir en compte que hi ha dos tipus de cartes especials que s'assignen distints tipus de química:

Tipus de carta	Química pròpia	Bonificació global
Icons	3/3 permanent	+2 a la nacionalitat, +1 a tota lliga del XI
Heroes	3/3 permanent	+2 a la lliga pròpia, +1 a la nacionalitat

Taula 5.2: Bonificacions addicionals de cartes especials

Amb aquestes regles es calculen els diccionaris de química `nationalities_chemistry`, `leagues_chemistry` i `clubs_chemistry`, així com el valor global `team_chemistry`.

Pel que fa a la mitjana, es clau saber com es calcula, ja que és un clau indicador de la valoració de la plantilla.

Mitjana El càlcul de la mitjana global d'un equip a *Ultimate Team* no és un simple promig aritmètic. Els desenvolupadors fan servir un mètode corregit que penalitza les cartes per sota de la mitjana i recompensa les que la superen.

Hi ha dos escenaris:

- **SBC** – es consideren només els 11 titulars ($n = 11$).
- **Plantilla completa** – s'inclouen els 7 suplents ($n = 18$).

El procediment és idèntic en ambdós casos; només varia n .

1. **Promig inicial** $M = \sum_{i=1}^n OVR_i / n$.

2. **Factor de correcció** Es calculen els punts extra

$$C = \frac{\sum_{OVR_i > M} (OVR_i - \lfloor M \rfloor)}{n}.$$

3. **Mitjana final** $Overall_{equip} = M + C$ i s'arrodoneix sempre a la baixa.

Exemple. Suposem 11 cartes:

$$OVR = \{84, 84, \underbrace{83, \dots, 83}_{8}, 80\}.$$

$$M = \frac{(84 \times 2) + (83 \times 8) + 80}{11} = 82.909,$$

$$\lfloor M \rfloor = 82,$$

$$C = \frac{(84 - 82) \times 2 + (83 - 82) \times 8}{11} = \frac{(2 \times 2) + (1 \times 8)}{11} = \frac{12}{11} = 1.091,$$

$$Overall_{equip} = 82.909 + 1.091 = 84.000 \text{ round_down } 84.$$

Per tant, la valoració oficial de l'equip és **84**. Qualsevol fracció decimal resultant sempre es trunca cap avall.

Per últim, les variables restants—`team_price`, `team_chemistry` i `players_chemistry`—es calculen, bàsicament, mitjançant sumes.

5.2.5 Funció fitness

Aquesta funció de *fitness* ha evolucionat en dues etapes distintes. En la fase inicial oferia resultats molt satisfactoris quan la SBC exigia una mitjana global elevada, però mostrava un rendiment pobre quan els requisits de mitjana eren baixos. En la fase final, es va fer una refactorització de tota la funció per a poder resoldre l'error anterior i millorar la puntuació *fitness*.

Per tant, ara ens toca exlicar les dues fases:

5. GENERACIÓ DE PLANTILLES

Fase inicial En la primera versió de l'algoritme es definiren dos objectius fonamentals:

- **Maximitzar el compliment dels requisits:** assolir exactament la mitjana global, la química i els llindars de clubs, lligues i nacionalitats que demana la SBC.
- **Minimitzar el cost sense malbaratar cartes valuoses:** prioritzar jugadors barats i reservar les cartes de mitjana alta per a reptes futurs amb requisits més estrictes.

La funció de *fitness* f es va interpretar de manera directa: com més gran és el valor retornat, millor és la plantilla. Per implementar-ho es feren servir dues variables:

- **requirement_score** (inicialment 0): puntuació acumulada dels requisits.
- **requirements_passed** (inicialment true): indicador de si l'equip compleix *tots* els requisits.

En aquesta primera versió la funció *fitness* s'interpretava de manera directa: com més gran és el valor retornat, millor és la plantilla.

Per a cada condició (*rating*, química, mínim de lligues, etc.) s'aplicava el mateix esquema:

- a) Si s'assoleix el requisit mínim, s'afegeix 1 a *requirement_score*. En cas contrari, s'hi suma la fracció

$$\text{requirement_score} += \frac{\text{valor de l'equip}}{\text{valor del requisit}},$$

i la variable *requirements_passed* es posa a *false*.

- b) Si l'equip supera el llindar màxim permès per al requisit, s'aplica una penalització de 0.5 a *requirement_score* i *requirements_passed* s'estableix igualment a *false*.

Quan la puntuació total quedava per sota d'un llindar intern, es feia una penalització addicional per descartar equips clarament deficientes.

Si, un cop evaluats tots els requisits, la variable *requirements_passed* continua sent *true*, s'afegeix a la puntuació global una bonificació inversament proporcional al cost de l'equip:

$$\text{requirement_score} += \frac{1}{1 + \text{team_price}}.$$

Així, dos equips que compleixen tots els requisits obtenen puntuacions diferents segons el cost: l'equip més barat suma més i, per tant, és preferit per l'algoritme evolutiu.

En síntesi, la funció de *fitness* es pot expressar com

$$f(\text{equip}) = \underbrace{\text{requirement_score}}_{\text{compliment dels requisits}} + \frac{\mu}{1 + \text{team_price}},$$

on $\mu > 0$ controla el pes de la component econòmica. L'algoritme evolutiu, per tant, prioritza les plantilles que satisfan tots els requisits amb el menor cost possible, penalitzant tant la manca com l'excés de recursos.

Els experiments amb aquest mètode van evidenciar una limitació important: en sumar un punt sencer quan l'equip assolia el llindar mínim de valoració global, una plantilla que sobrepassava àmpliament el requisit obtenia la mateixa puntuació que una que just el complia. Això resulta contradictori amb l'objectiu d'estalviar cartes de mitjana elevada; volem preservar-les per a reptes futurs.

En canvi, per als altres requisits (química, nacionalitats, clubs, etc.) no penalitzem l'excés: una química superior a la demandada pot aparèixer de forma natural en la millor combinació i, a diferència de la valoració, no encareix l'equip ni malbarata recursos.

A més, vam identificar altres inconvenients addicionals:

- i) La definició de la *fitness* “com més gran millor” dificulta la interpretació i el control de penalitzacions,
- ii) El cost de l'equip—variable essencial—no estava ponderat de manera adequada.
- iii) La penalització als equips que no compleixen els requisits era massa severa: alguns d'aquests equips podrien convertir-se en plantilles vàlides després d'una o dues mutacions.

Aquestes deficiències van motivar el redisseny complet de la funció de puntuació en la fase següent.

Fase final En aquesta versió definitiva la funció de *fitness* f mesura l'**error**: com més petita és la puntuació, millor és la plantilla. L'objectiu doble continua essent:

- **Minimitzar el cost:** prioritzar els jugadors més barats i reservar les cartes de mitjana alta per a futurs reptes amb requisits superiors.
- **Complir estrictament els requisits:** assolir la mitjana global, la química i els llindars de nacionalitats, clubs o versions de carta sense excedir-los innecessàriament.

Es defineixen tres variables:

- **requirement_score** (init 0) — error acumulat pels requisits.
- **unmet_requirements** (init 0) — nombre de requisits incomplerts.
- **PENALTY_PER_UNMET_REQUIREMENT** (0–1) — penalització base per requisit incomplert.

Aquesta fase resol les mancances de l'etapa anterior i millora el càclul de la puntuació de *fitness*.

a) Mitjana global mínima

Es penalitzen per igual el dèficit i l'excés:

$$\text{requirement_score} += \left| \frac{\text{valor de l'equip}}{\text{valor del requisit}} - 1 \right|.$$

Si el requisit no es compleix, s'incrementa **unmet_requirements**.

5. GENERACIÓ DE PLANTILLES

b) Resta de requisits

Només es penalitza quan *no* es compleixen:

$$\text{requirement_score} += \left\lfloor \frac{\text{valor de l'equip}}{\text{valor del requisit}} - 1 \right\rfloor, \quad \text{unmet_requirements} += 1.$$

Per exemple, si el requisit és una mitjana de 80 i s'obté 75, la divisió $75/80 = 0.9375$. Si s'obté 85, la divisió és 1.0625. Restant 1 i prenent el valor absolut, l'*error* és 0.0625 en ambdós casos, reflectint adequadament la distància al requisit, tant per defecte com per excés.

Un cop evaluats els requisits, es penalitza el preu de la plantilla normalitzant-lo entre 0 i 1. Cal conèixer el preu P i uns extrems P_{\min} i P_{\max} (definitos més endavant):

$$\text{requirement_score} += \frac{P - P_{\min}}{P_{\max} - P_{\min}}.$$

Una vegada es normalitza el preu, realitzam una petita penalització a les plantilles que no han superat algun requisit. La penalització es de la següent manera:

`requirement_score += unmet_requirements × PENALTY_PER_UNMET_REQUIREMENT.`

En síntesi, l'*error* total es pot expressar com

$$f(\text{plantilla}) = \underbrace{\frac{C_{\text{cost}}}{\text{preu normalitzat}}}_{\text{cost}} + \lambda \underbrace{\frac{\text{Prequisits}}{\text{penalitzacions}}}_{\text{penalitzacions}},$$

on $\lambda > 0$ pondera la severitat de les penalitzacions. L'algoritme evolutiu, per tant, cerca la plantilla de menor cost que satisfaci tots els requisits de la SBC sense excedir-los.

Per acabar la secció d'avaluació de les plantilles, cal destacar que els equips es classifiquen segons la puntuació de *fitness*. D'aquesta manera els millors apareixen al capdamunt i, mitjançant elitisme, es genera la nova població amb els deu primers equips.

A partir d'aquesta població inicial es generen tants descendents com indica la constant `POPULATION_SIZE`, aplicant successivament els operadors de creuament i mutació.

5.2.6 Creuament

Primer cal seleccionar dos pares. La selecció es fa triant aleatoriament dos equips de la nova població. Un cop escollits, el creuament s'aplica com segueix:

1. Es comprova la longitud de les dues plantilles per assegurar que coincideixen.
2. Es recorre cada posició de forma paral·lela.
3. Es genera un nombre aleatori $\in [0, 1]$.

Si és > 0.5 , s'intenta prendre el jugador del pare 1; si aquell jugador ja és present al fill, es passa al següent pas.

4. En cas contrari (o si el pas anterior falla), s'intenta afegir el jugador corresponent del pare 2, sempre que no sigui duplicat.
5. Si cap dels dos jugadors pot inserir-se, s'escull aleatoriament un jugador del club que encara no sigui a la plantilla i s'assigna a la posició.

Un cop completades totes les posicions s'obté la nova plantilla *fill*, barreja dels pares 1 i 2.

5.2.7 Mutació

Després del *crossover* s'aplica la mutació. Amb probabilitat MUTATION_RATE es decideix si la plantilla muta o no:

1. Si hi ha mutació, es tria una posició aleatòria.
2. Es selecciona un jugador del club que encara no formi part de la plantilla i se substitueix el jugador existent en aquella posició.
3. Si no hi ha mutació, l'equip roman sense canvis.

El valor de MUTATION_RATE s'analitzarà posteriorment mitjançant un estudi sistemàtic per determinar el seu valor òptim.

En finalitzar totes les iteracions, l'equip amb la puntuació més alta—situat a la primera posició de la població—es considera la millor solució obtinguda i s'exporta a un fitxer `best_team.json`, que conté la llista completa de jugadors, la informació agregada de l'equip i la seva puntuació de *fitness*.

CAPÍTOL

6

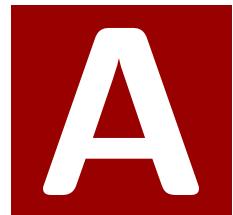
RESULTATS I DISCUSSIÓ

Parlar de l'algoritme evoilutiu: paràmetres num_generations population_size i mutation_rate. També de max_team_price, min_team_price.

CAPÍTOL
7

CONCLUSIONS

A P È N D I X



APÈNDIX

BIBLIOGRAFIA

- [1] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions and reversals,” *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710, 1966. 4.2.1