UNIVERSITAT POLITÈCNICA DE CATALUNYA

VISIÓ PER COMPUTADOR

Pràctica Final: Cartoon Network

Jordi Muñoz Florensa i Joan Carles Veny Martí

QT 2024-25

Índex

1 Introducció

Document per a representar el procés de disseny i implementació del projecte final de l'assignatura de Visió per Computador. L'objectiu d'aquesta pràctica és donada una imatge d'entrada, identificar la sèrie a la que pertany, en el nostre cas 10 series: Barrufets, Bob esponja, Gat i gos, Gumball, Hora d'aventures, Oliver i Benji, Pare de familia, Pokemon, SouthPark i Tom & jerry . En les properes seccions desglosarem el procés dut a terme per obtenir la versió final del codi, les idees intermitges i els problemes trobats fent la pràctica.

2 Tractament Previ

Donat que les dades inicials ens arribaven totes en una carpeta TRAIN i necessitarem posteriorment dividir-les en un conjunt de entrenament i un conjunt de test, varem realitzar un script "split_train_test"per tenir una estructura DATA/train/-nom_serie/* i DATA/test/nom_serie/*.

3 Característiques Classificatòries

Donat que estem davant d'un problema de classificació d'imatges en el que no tenim cap característica tabular més enllà dels n*m valors que representen el color de cada píxel de la imatge, hem utilitzat tècniques per extreure caracterísitques que ens permetin discriminar entre una serie o una altra.

3.1 Versió inicial

En la primera versió incial, donat que les imatges que teniem no donaven peu a una fàcil segmentació ni binarització, per tant algorismes com Otsu, Canny, Sobel, k-means... van ser descartats des d'un primer moment.

Així que vam decidir encarar el problema amb tècniques que fosin invariants a transformacions geomètriques ja fosin rotacions, translacions o escalats i que es guiessin principalment per l'estil de la sèrie ja que a simple vista podiem apreciar aquest estil i per tant tenien implícitament una gamma de colors bastant diferent entre elles.

Per fer-ho vam seguir els següents passos:

1. **Conversió de les imatges a espai de color RGB**: Primer de tot assegurem que totes les imatges tinguin tres canals de color, independentment del format original.
2. **Càlcul d'histogrames de color**: LLavors s'obté un histograma per a cada canal de color (roig, verd i blau), utilitzant 16 bins per canal.
3. **Normalització dels histogrames**: A continuació, er assegurar que els valors dels histogrames siguin comparables, es normalitzen dividint cada valor pel total de píxels de la imatge.

4. **Concatenació dels histogrames**: Finalment, els tres histogrames es concatenen per formar un vector de característiques de 48 valors (16 bins per canal).

A continuació podem observar el resultat dels Histogrames de color respecte cada sèrie, on es veu que cada una presenta un histograma ben diferenciat, com suposava la nostra hipòtesis, permetent així utilitzar aquesta carecterística com a classificador:

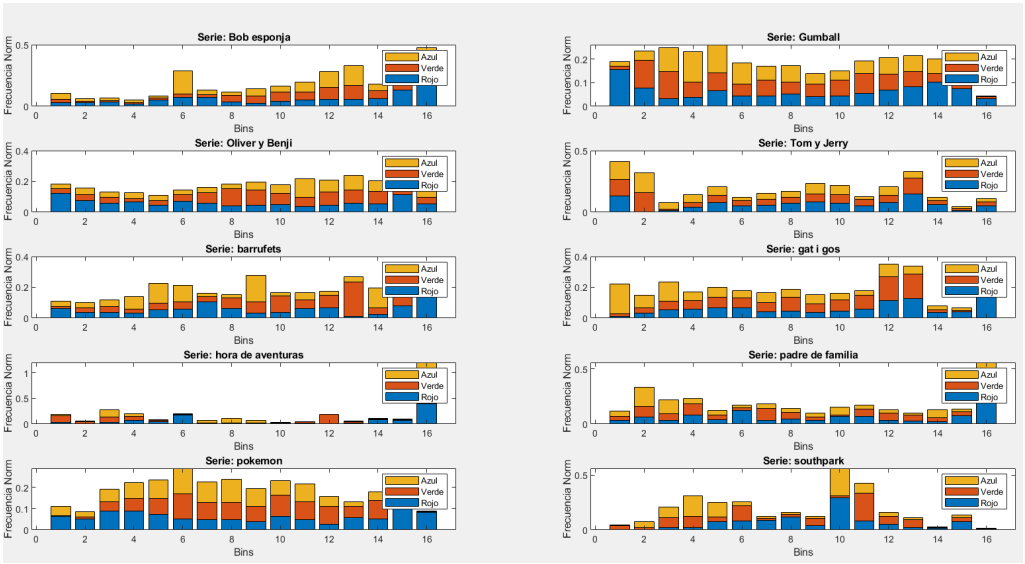

Figura 1: Histogrames de 16 bins de les series

Posteriorment vam entrenar un model d'ensamble d'arbres amb TreeBagger en el que vam obtenir una precisió del 92% i veiem com evolucionava l'algorisme en funció del nombre d'arbres que utilitzavem.

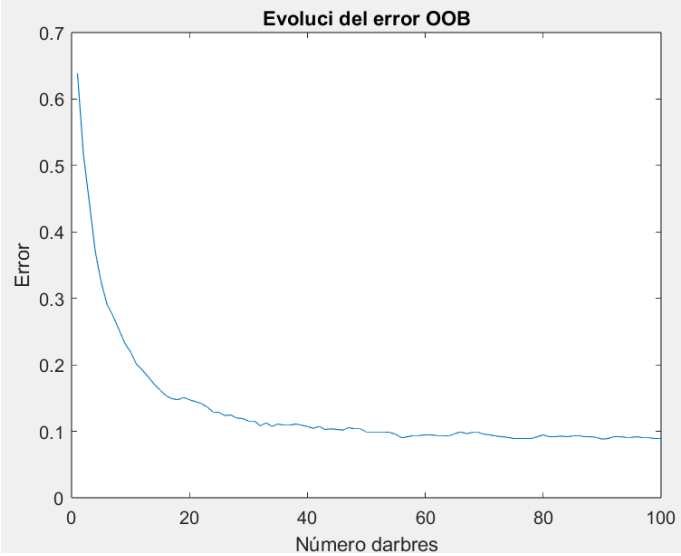

Figura 2: Evolució de l'error en funció del nombre d'abres

Així que despres d'aquesta primera versió vam decidir emprar 100 arbres pels models entrenats amb TreeBagger ja que el temps de computació no era molt alt i segurament en futures versions on hi hauran més característiques, un nombre més alt que 80 que podria ser el punt a partir del cual la funció d'error ja es bastant plana, no seria suficient.

3.2 Versions posteriors

Posteriorment durant el procés de pluja d'idees per millorar la primera versió vam decidir implementar noves estratègies que també fossin invariants a transformacions geomètriques i que ens puguessin donar un valor afegit al que ens donaven els histogrames de color en RGB. Així que vam pensar en les següents tècniques:

- **Oversampling**: Ja que com bé es sabut, davant de problemes de predicció ja siguin de regressió o classificació, les dades son la base dels resultats, si les dades no són suficients o bones no hi hauran bons resultats i com que nosaltres mancavem de la primera pena, vam pensar que incloure més imatges serie profitós.
- **SIFT**: Ja que podriem detectar punts clau que identifiquesin el personatge de la sèrie.
- **HOG**: Ja que son descriptors de característicàs utilitzats precisament per la detecció d'objectes, formes i entitats.
- **Histogrames en espai HSV**: Com que l'espari HSV té en compte la il·luminació, es va pensar que podria tenir millores respecte a l'RGB en els casos on hi hagi canvis significatius d'il·lumincació entre les imatges
- **Local Binary Patterns (LBP)**: Loacal Binary Patterns presenta certes millores respecta a la detecció de textures diferenciades, que en el cas de les diferents sèries amb diferents estils de dibuix i animació semblava adient utilitzarlo.

4 Classificadors

Donat que l'assignatura no es centra en models d'aprenentage autònom, en la pràctica no es demanava exprimir al màxim aquest camp, per tant hem decidit utilitzar 3 models els quals no han estat pre-entrenats amb cap conjunt de dades d'imatges com podria ser una xarxa neuronal com la GoogleNet en la que podriem haver fet fine-tuning. Pel que fa als classificadors, vam voler provar 3 models que no usen aprenentatge profund (com podria ser una CNN, que sabem que en imatges es dels models que millor funciona), que solen treballar bé amb dades del caràcter del que les disposem.

4.1 Random Forest:

El **Random Forest** és un algorisme d'aprenentatge supervisat que combina múltiples arbres de decisió per millorar la precisió i reduir el risc de sobreajustament. El Random Forest té diversos paràmetres ajustables, com ara el nombre d'arbres, el nombre màxim de característiques considerades a cada divisió, i la profunditat màxima dels arbres. Aquests paràmetres es poden ajustar per optimitzar el rendiment del model. A diferència dels arbres de decisió simples, que poden ajustar-se massa a les dades d'entrenament, el Random Forest aconsegueix mantenir un bon equilibri entre biaix i variància.

4.2 K-Nearest Neighbors (KNN):

K-Nearest Neighbors (KNN): El KNN és un dels classificadors més senzills i intuïtius. Funciona trobant els k veïns més propers a la mostra que es vol classificar i assignant-li la classe més comuna entre aquests veïns. La distància entre les mostres es calcula mitjançant una mètrica com la distància euclidiana. Tot i que és fàcil d'implementar, el KNN és molt sensible al soroll i pot veure's afectat per dades irrellevants o característiques no significatives. A més, el rendiment pot disminuir considerablement si el nombre de característiques és molt gran.

4.3 Support Vector Machines (SVM):

L'SVM és un algorisme potent que intenta trobar un hiperplà que separi les diferents classes de dades de manera òptima. Aquest hiperplà es selecciona de manera que maximitzi la distància entre les mostres més properes de classes diferents, anomenades *marges de separació*. L'SVM és molt efectiu en problemes de classificació lineal i pot adaptar-se a problemes no lineals mitjançant l'ús de *nuclis* (kernels). Encara i així presenta certs inconvenients, com que requereix un ajust precís dels seus paràmetres o pot ser computacionalment costós per a grants conjunts de dades.

5 Proves Realitzades i Resultats Obtinguts

Un cop explicades les característiques i els descriptors que es van platenjar per realitzar el projecte, en aquest apartat desglosarem les diferents proves realitzades per determinar el model final utilitzat. A més a més, incorporarem un redimensionament del tamany de la imatge sempre abans de les proves on forçarem a que les imatges que es tractin, tinguin una mida de 128*128 píxels.

5.1 Prova 1: Histogrames de color amb HSV

Per realitzar la evaluació de si relament el HSV millora la qualitat de les dades, entrenarem un model amb KNN, un amb RF y un amb SVM. Per tots tres realitzarem una breu epxloració d'hiperparàmetres i ens quedarem amb la combinació que ens doni millors resultats.

Per realitzar el canvi, simplement usarem la funció rgb2hsv on enlloc d'agafar els numBins valors per la component R, G i B del píxel agafarem el hue (H), la saturació (S) i la brillantor o value (V). A continuació podem veure els histogrames per cada sèrie y cada valor.

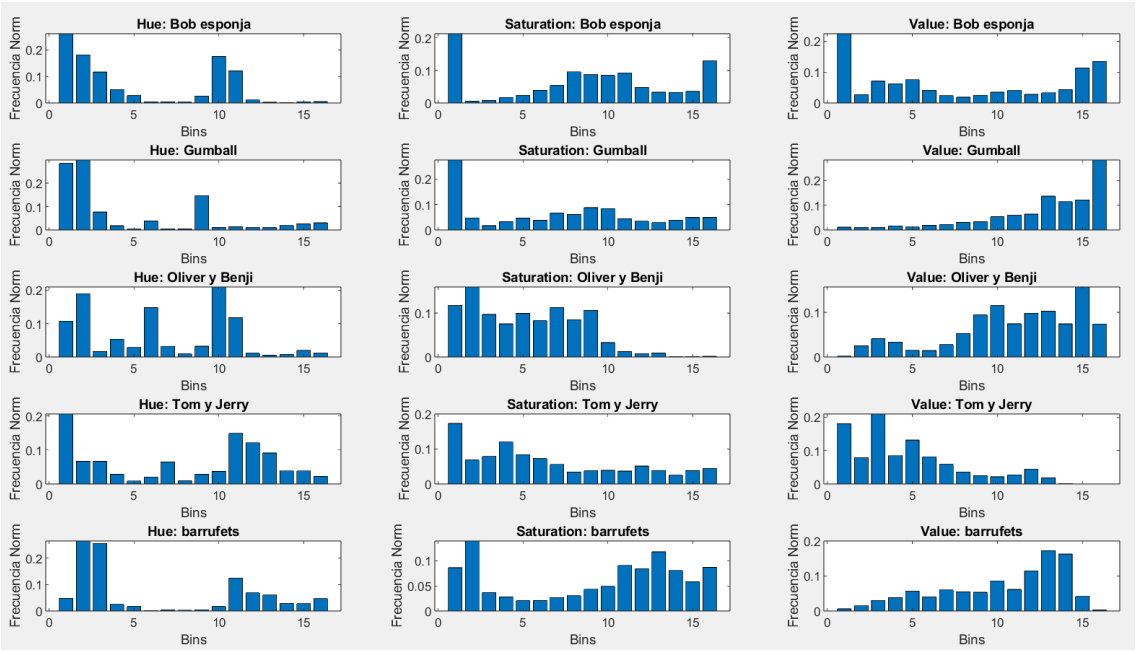

Figura 3: Histograma HSV

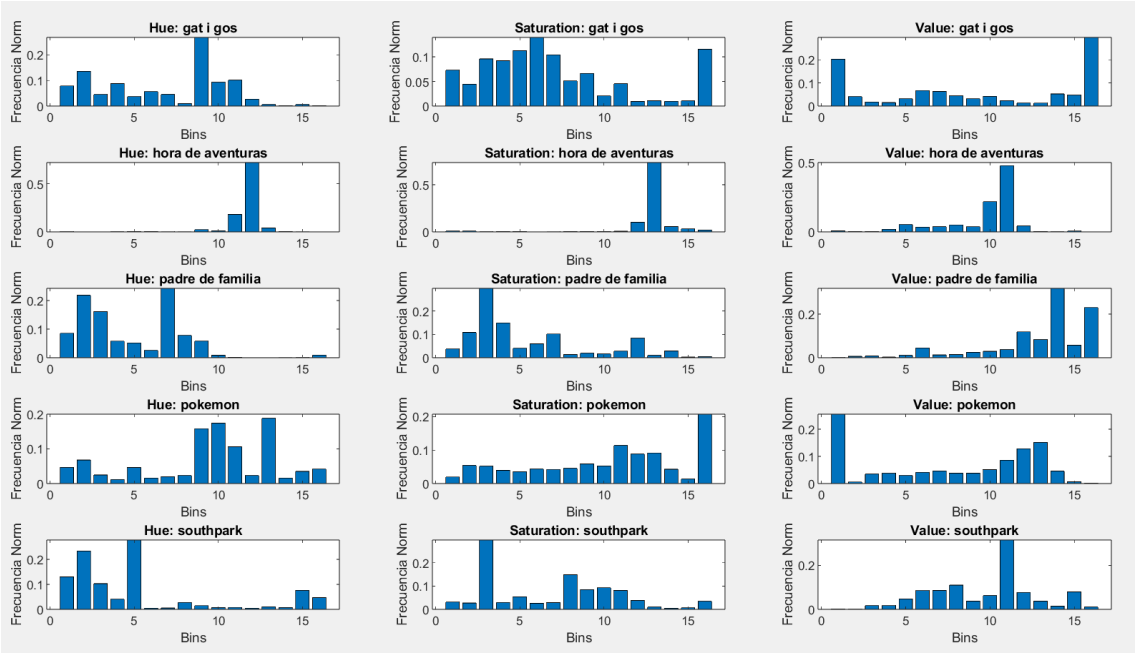

Figura 4: Histograma HSV

Com veiem, seguim apreciant grans diferencies entre sèries pel que fa als histogrames, el que ens inidca que probablement obtinguem resultats similars o millors.
Pel que fa als models, farem la prova amb un KNN explorant el nombre de veïns des de 1 fins a 10 veïns, pel que fa al randomForest ens quedarem amb 100 arbres ja que com vam veure el la versió inicial, era el valor a partir del cual ja no era molt rentable seguir afegint arbres al model i pel que fa al SVM provarem amb tres funcions de kernel diferents, lineal, gaussiana i polièmica.

Un cop aplicats els models a les dades, veiem les següents taules, on efectivament verifiquem que el color HSV ens ajuda més a predir millor les sèries, notem una diferència més alta en el KNN i en el SVM ja que el RF amb 100 arbres en el cas de RGB es capaç de millorar més fins quedar-se a un 2% de l'HSV. Veiem també que el SVM es comporta bastant pitjor que els altres dos i que el número de bins generalment és 16.

	KNN	RF	SVM
Best Accuracy	91.88%	94.02%	86.75%
Params	16 bins, 1 veí	16 bins, 100 arbres	16 bins, polinòmica

Taula 1: Comparació RGB

	KNN	RF	SVM
Best Accuracy	95.73%	95.94%	92.95%
Params	32 bins, 1 veí	16 bins, 100 arbres	16 bins, polinòmica

Taula 2: Comparació HSV

Podem veure també en les matrius de confusió com en els tres models, les clases que predim millor son la gat i el gos, pokemon i southpark. I en general en les clases que fallem no solem pecar de forma evident de recall o de acuraccy, és a dir no hi ha norma general al fallar.

Si agafem el model que confon menys clases entre elles, seria el KNN amb HSV i veiem que confonem les series Oliver i Benji amb Gumball, Tom & Jerry amb Bob esponja i Padre de familia amb Oliver i Benji i Hora d'aventures.

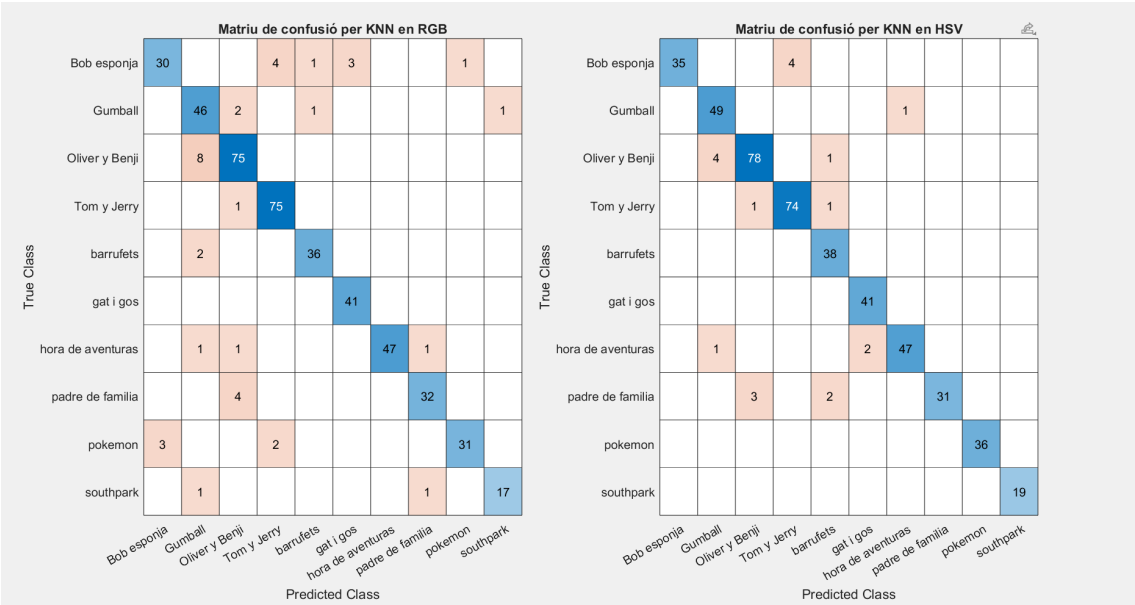

Figura 5: Comparació RGB vs HSV amb KNN

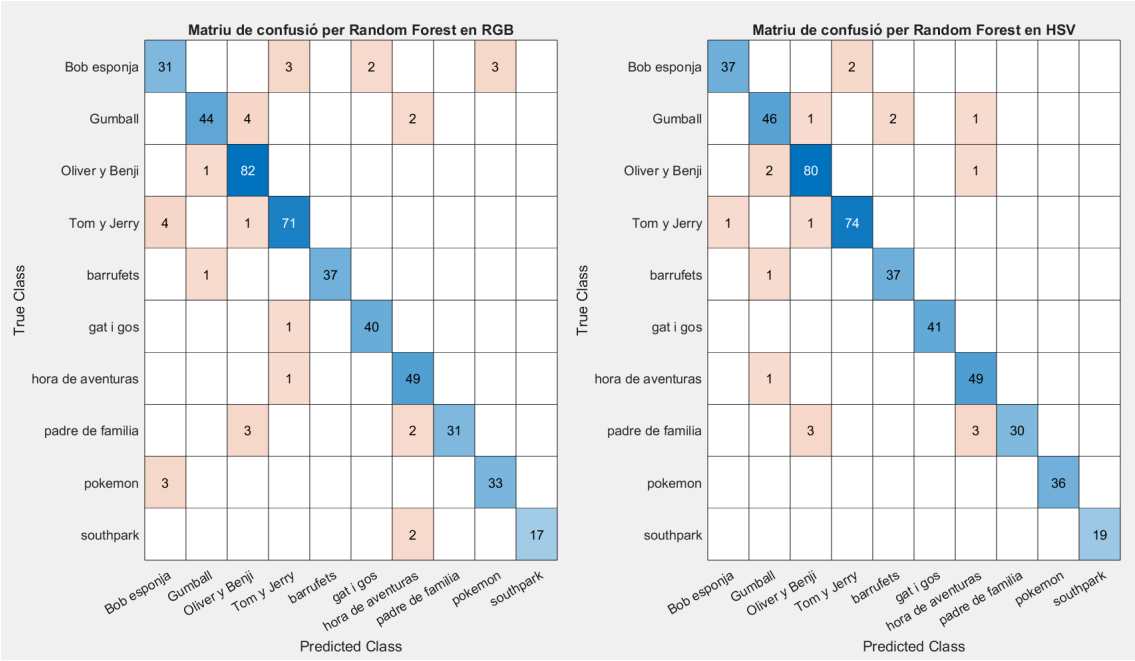

Figura 6: Comparació RGB vs HSV amb RF

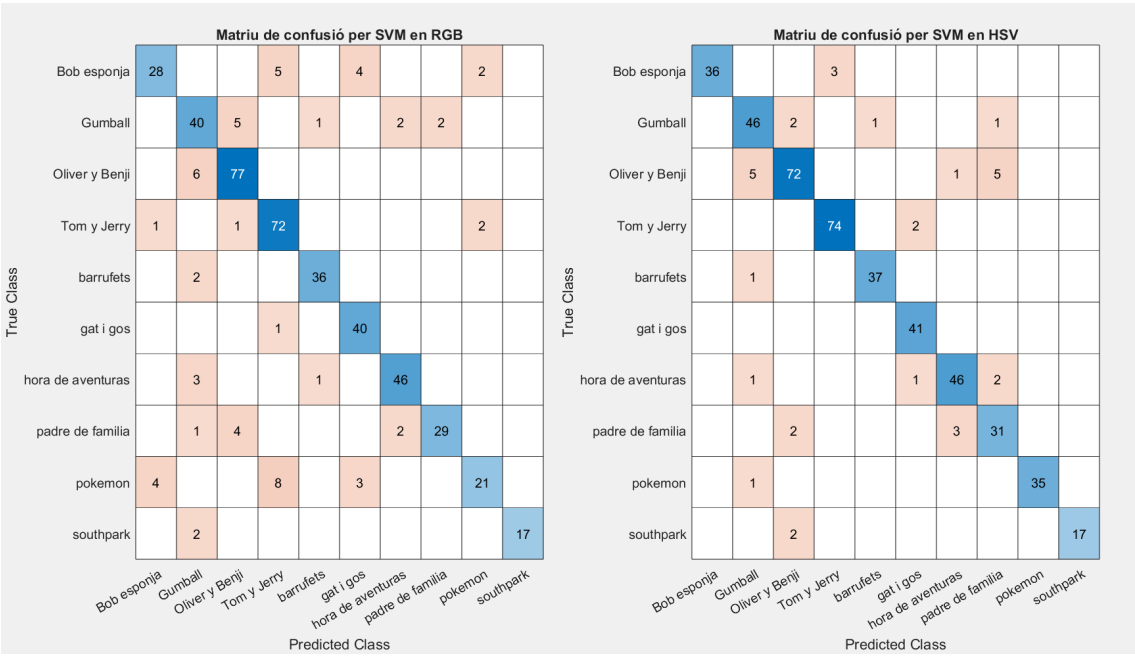

Figura 7: Comparació RGB vs HSV amb SVM

Podem veure a continuació la importancia que el Random Forest ens dona en el cas d'emprar 16 bins. Veiem com en general la importància es bastant similar en tots els bins de cada component, ara bé, en totes tres components veiem com en els valors extrems (inicials i finals) s'assigna més importància, en alguns fins i tot el doble d'importància que la mitjana.

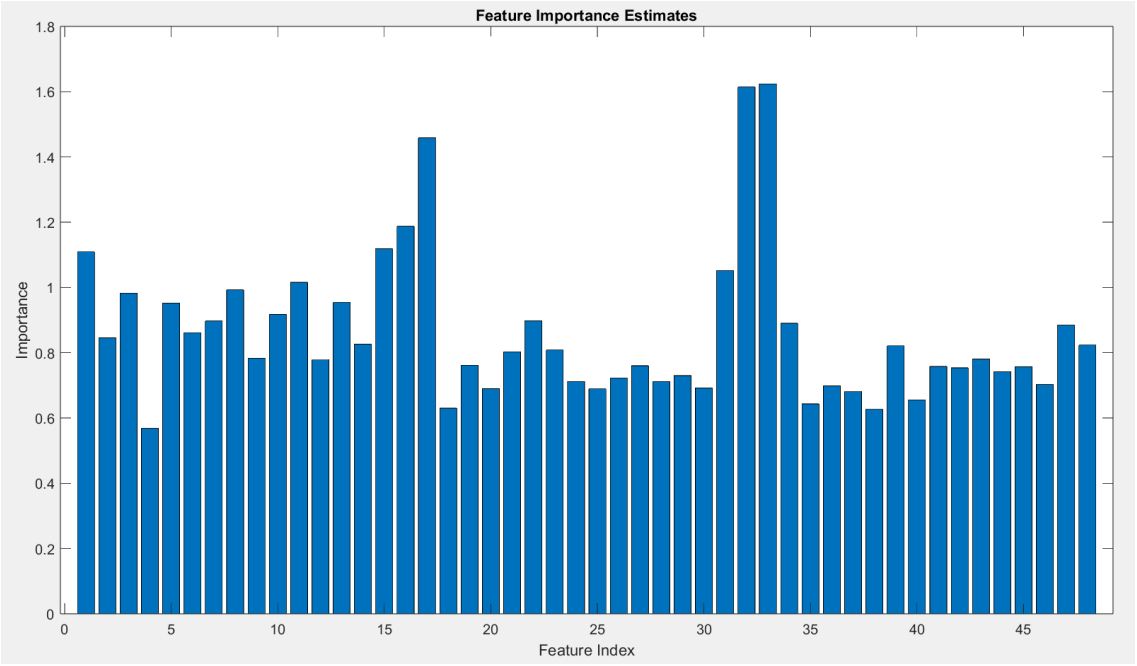

Figura 8: Feature imporance amb RF de 16 bins

Finalment concluïm amb que utilitzarem HSV en quant als histogrames de color ja que per tots tres models hem conseguit una major precisió i utilitzarem 16 bins com a parametre per els histogrames.

5.2 Prova 2: Incorporació de SIFT

La següent prova que vam fer, va estar la incorporació de SIFT en el model, que com bé sabem és un algoritme utilitzat per detectar i descriure característiques clau en imatges, conegut per la seva robustesa davant canvis d'escala, rotació, il·luminació i fins i tot petits canvis en perspectiva.

Com també sabem, sift utilitza una imatge de referencia de la cual n'extreiem els punts clau per desprer comparar-los amb els de la imatge rebuda i establir mitjançant matchFeatures si hi ha o no algun punt que pugui pertanyer a la imatge de referència.

Per tant, per fer-ho, vam decidir escollir les 10 imatges que per judici visual, millor representaven la sèrie la cual voliem predir (buscant aquells frames on surtien el/s personatje/s principals).

Primerament varem tenir el problema de que en les sèries Pokemon i Southpark les imatges que millor representaven la sèrie tenien o bé unes lletres del episodi al que pertanyia la imatge en el cas de Pokemon o bé el un logo que pot ser pertanyia a la plataforma on es va emetre la sèrie en el cas de southpark. Per tant en aquestes dues imatges, no detectavem correctament els keypoints ja que se li donava molta importància a aquelles lletres.

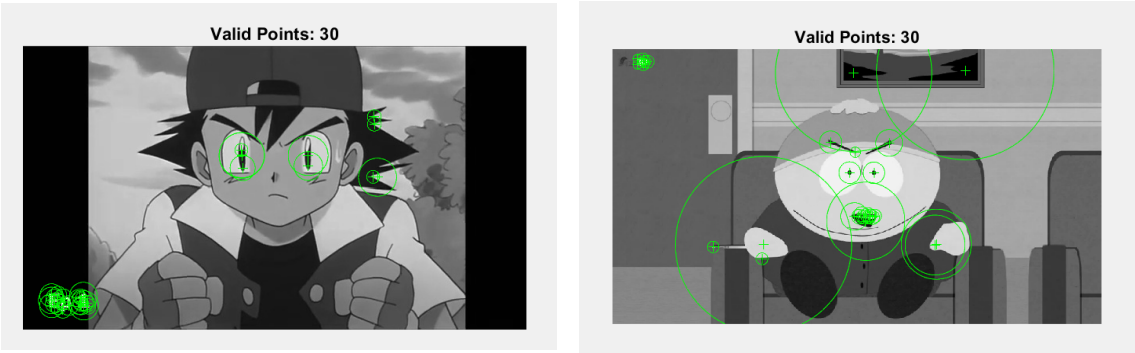

Figura 9: Punts incorrectes

Per tant, seguidament vam pensar en elaborar un script per eliminar aquelles parts on hi havia lletres ficant punts de color negre a damunt.

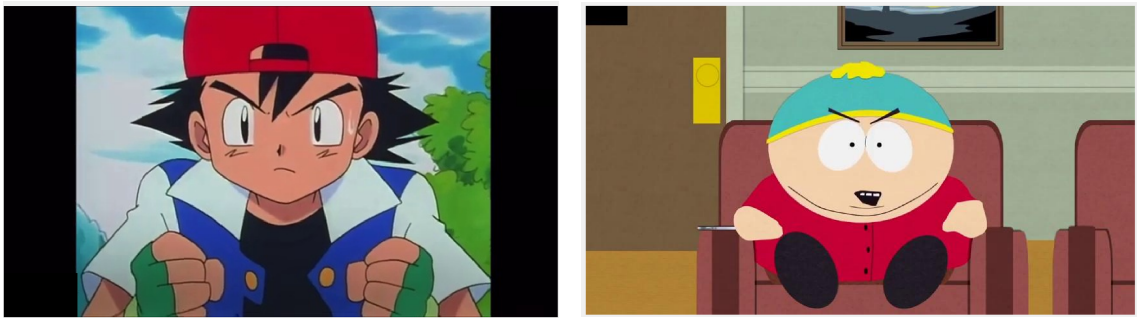

Figura 10: Substracció lletres i logo

Seguidament, vam veure que per el cas de southpark no era suficient ja que just era la sèrie on el personatje es més "plaï té menys característiques importants en la seva forma.

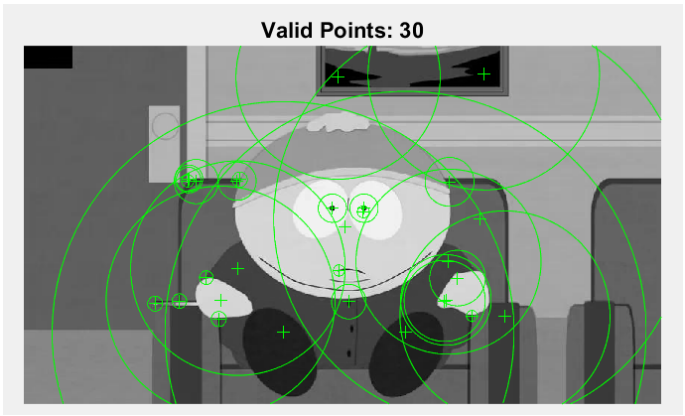

Figura 11: Punts clau incorrectes

Com hem vist ens detectava punts clau en la butaca i el cuadre, així que vam decidir retallar al màxim la imatge per mostrar només el personatje.

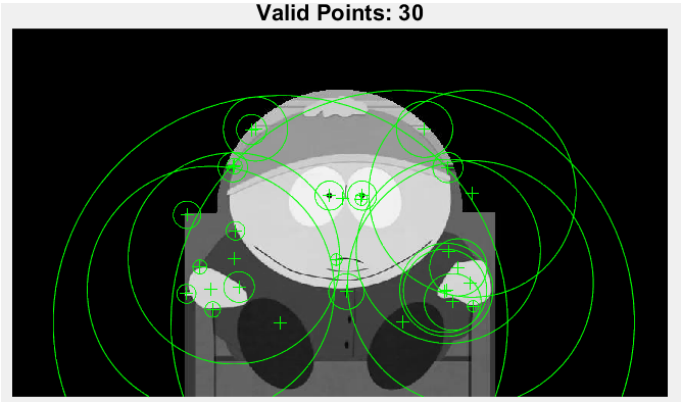

Figura 12: Punts clau correctes

Ara sí, un cop teniem totes les imatges amb els keypoints que voliem, vam procedir a quedarnos amb els 30 punts més importants. On vam tenir un altre problema ja que la funció extractFeatures per molt que li indiquis per parametre un número de punts, et tornarà aquell número de punts pero en alguns punts hi haurà més d'un descriptor, per tant, vam decidir quedarnos amb un descriptor per punt ja que necessitavem tenir un nombre de catergories constant per totes les instancies.

Els descriptors d'un punt tenen 128 valors per tant no vam considerar fer la clasificació utilitzant els 128 valors com a columnes del nostre conjunt de dades, vam plantejar diverses idees per reduir dimensionalitat per fer-ho, com per exemple aplicar PCA de forma individual a cada descriptor per quedarnos amb les 10-20 components principals de cada punt i tenir 30 punts importants el que farien unes 300-600 variables pero finalment no ho vam implementar ja que perdriem completament la interpretabilitat dels models i per altra banda, res ens garantitza que les components principals de descriptors que descriguin el mateix punt en imatges diferents, acabin tenint valors similars.

També vam barallar altres opcions com crea "bosses de paraules"(bag of words) a cada punt, però ens vam donar compte rapidament que era computacionalment molt car, per cada punt necessitavem uns 5 segons només per realitzar el clustering per les bosses de paraules.

Així que finalment, vam decidir optar per una solució en la que crearíem un vector de 10 posicions representant les 10 series i cada columna del conjunt de dades seria la probabilitat de ser d'aquella classe o no mitjançant matchFeatures, la cual ens retorna els punts que poden fer match entre dues imatges i amb la cuantitat de punts que fan match en cada clase establiriem una mena de contador que despres normalitzariem per cada imatge i per tant obtindriem una probabilitat. En els casos en els que ningun punt fa match amb alguna de les 10 imatges de referencia, simplement ficariem a 0 totes les probabilitats.

Per fer-ho vam crear una funció getUniqueFeaturesAndPoints que ens retornava ja les features úniques dels 30 punts més importants la qual utilitzavem en per

fer la comparació per cada imatge i despres inserir les probabilitats al vector que conformaria les nostres dades.

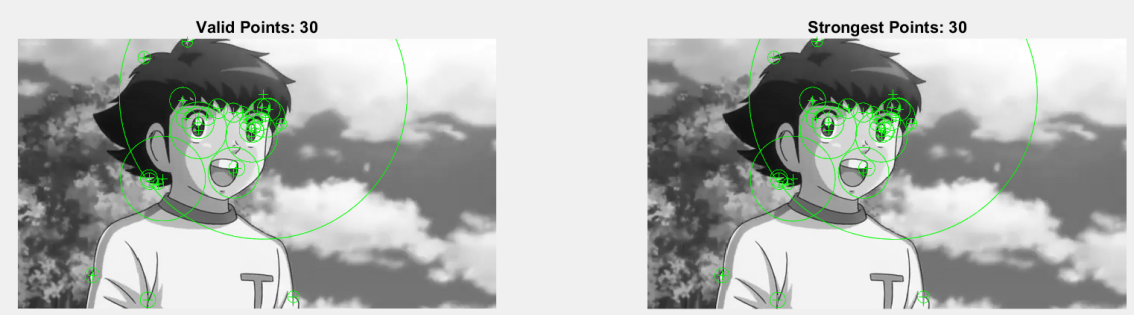

Figura 13: 30 punts vàlids contra els 30 puns més forts

A continuació mostrem els 30 punts més importants detectats per les imatges seleccionades per cada sèrie.

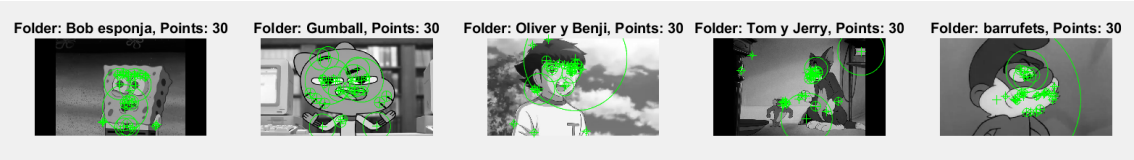

Figura 14: 30 punts seleccionats

Figura 15: 30 punts seleccionats

Un cop aplicats els resultats, veiem que hi ha una millora notoria en el Random Forest, podem apreciar com en classes com Gumball on la detecció de punts es realitza de forma correcta en la majoria de casos donat que te una forma bastant única que el permet discriminar correctament de les altres i veiem també com en Oliver i Benji on obteniem també molts punts clau discrimnatoris de qualitat, també tenim una millora, recuperem dos que prediem com Pare de familia, per altra banda empitjorem en la classe Pare de familia on la confonem en 2 imatges més amb Hora d'aventures, per això la millora no es tant notoria, guanyem algo menys d'un 1%.

Pel que fa al KNN veiem directament que empitjora, lleugerament, però empitjora. La informació que afegeix confon en alguns casos a l'algoritme i no ajuda en la predicció. Despres d'indagar en els motius hem vist que degut a que en les imatges hi ha molts frames on el personatge emprat per extreure els punts clau no surt o surt amb unes transformacions geomètriques que no son suportades per els keypoints i per tant, en el conjunt de entrenament tenim de 1099 instancies, 131 no detecten

cap match amb cap sèrie i de les restants, només 138 fan match únicament amb la sèrie que realment li tocaria.

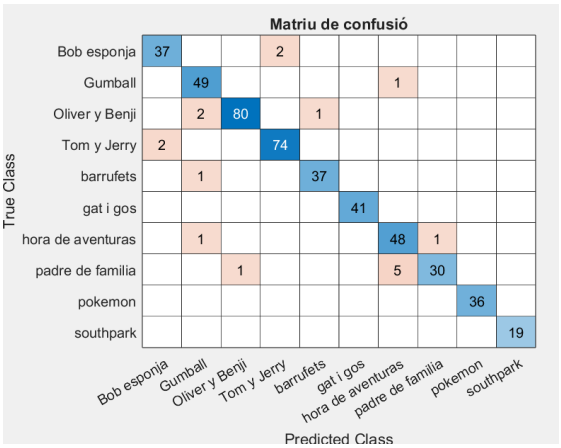

Figura 16: 96.37%

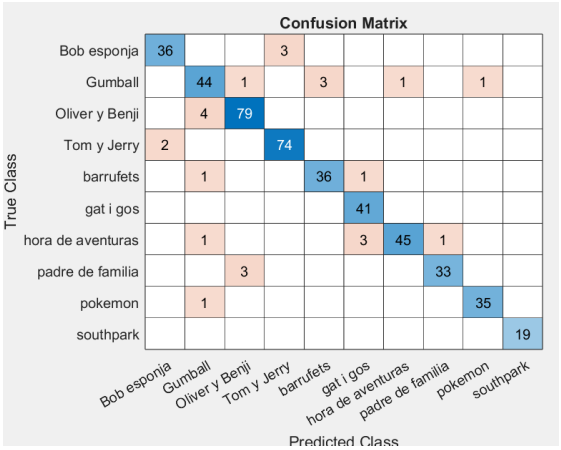

Figura 17: 94.44%

Figura 18: RF i KNN

Com hem vist, les estadístiques resultants ratifiquen que aquest mètode no aporta molta utilitat a l'objectiu del nostre problema, a part de que incorpora un temps de computació molt més elevat pel que fa a la formació dels conjunts d'entrenament i de prova ja que per cada imatge hem de fer un seguit d'operacions extractFeatures/matchFeatures que provoca l'afegit d'un temps més que considerable.

5.3 Prova 3: Incorporació de HOG

Després d'haver fet proves amb els histogrames de color amb HSV, que semblen un encert, i amb SIFT, que ha resultat ser poc útil per la finalitat de la pràctica, el que se'ns va acudir va ser provar amb altres descriptors vist a la teoria, concretament els Histogrames de Gradients o HOGs.

Per provar-ho, canviem els histogrames HSV pels descriptors de HOG i comprovem amb l'execució amb els models Random Forest i KNN:

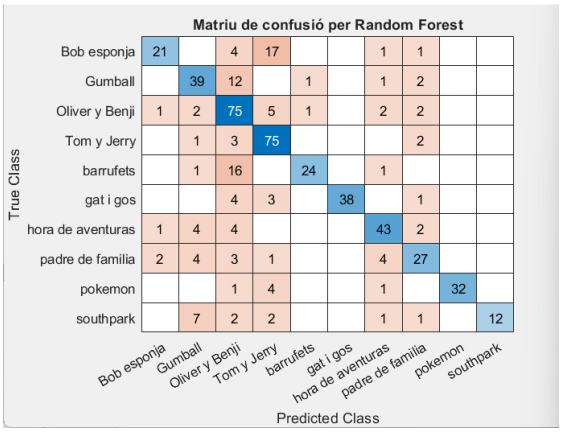

Figura 19: 75.10%

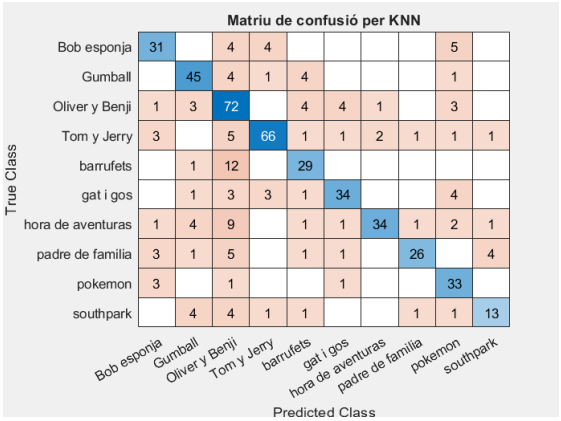

Figura 20: 74.51%

Figura 21: RF i KNN

Si entenem que HOG com a descriptor despunta en la classificació de figures/objectes, però fora utilitzar cap mena d'informació sobre color i/o textura, podem veure el perquè de la baixada de precisió en ambdos models. En sèries com Oliver i Benji o Tom i Jerry, on els fotogrames solen presentar una o vàries figures i uns fons poc carregats, HOG aconsegueix bons resultats, mentre que a sèries com Southpark o Bob Esponja, l'alta càrrega d'elements als fons dificulten enormament la seva capacitat de classificació. Aquest fet és deu a que en aquestes sèries els gradients no presenten un distribució dominant. Entenent això, vam decidir mantenir l'ús dels histogrames de color amb HSV, ja que presetaven millors resultats per la tasca a realitzar.

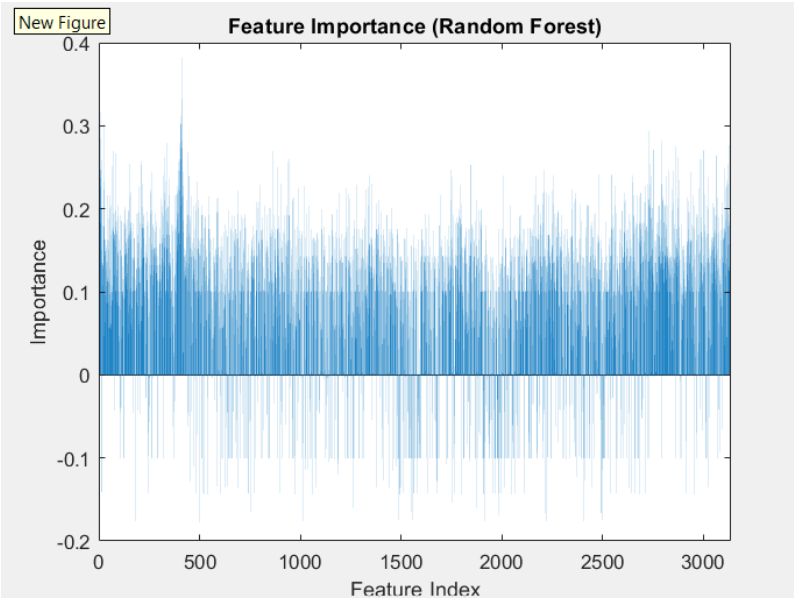

Figura 22: Feature imporance amb RF de 16 bins

A més, el gràfic de les característiques importants ens revela que a diferència dels

histogrames de color, HOG utilitza un ventall molt més ampli de carecterístiques, on la gran majoria s'utilitzen per fer la classificació general. Veiem com inclus, hi ha un nombre considerable de caracterísitques que resten valor a la predicció i llevat d'un pic important, la majoria del gràfic és relativament uniforme.

Concloiem en que el HOG per si sol no ens aporta molta utilitat, s'haurà de combinar amb altres tècniques per veure si ens és d'ajuda per millorar els resultats de l'histograma de color amb HSV

5.4 Prova 4: Incorporació de LBP

Vegent que HOG per si sol no augmentava l'eficacia del model, vam seguir investigant fins que ens vam topar amb el model Local Binary Patterns, que com esmentem a l'inici, presenta una millor efectivitat davant la detecció de diferents textures.

Després d'aplicar els canvis i fer les comprovaciones amb RF i KNN, vam obtenir els següents resultats:

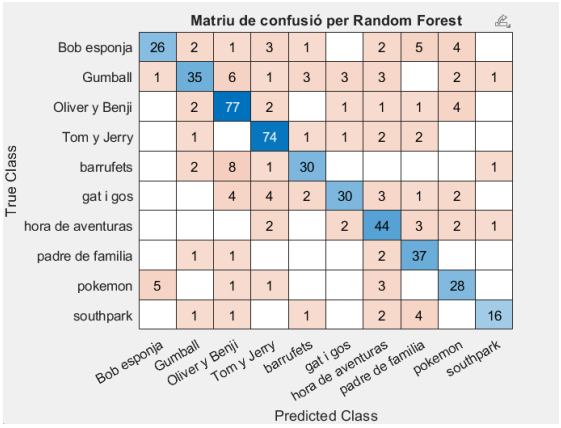

Figura 23: 77.24%

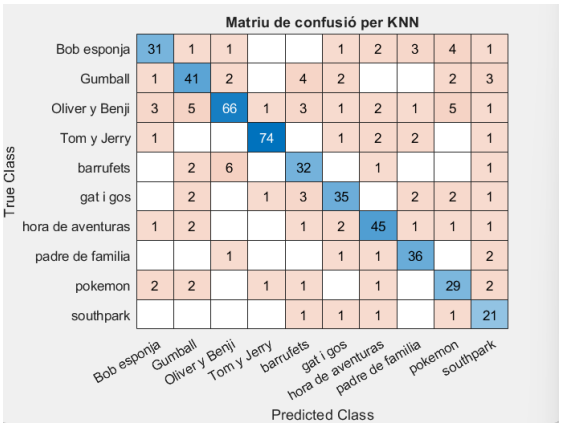

Figura 24: 79.77%

Figura 25: RF i KNN

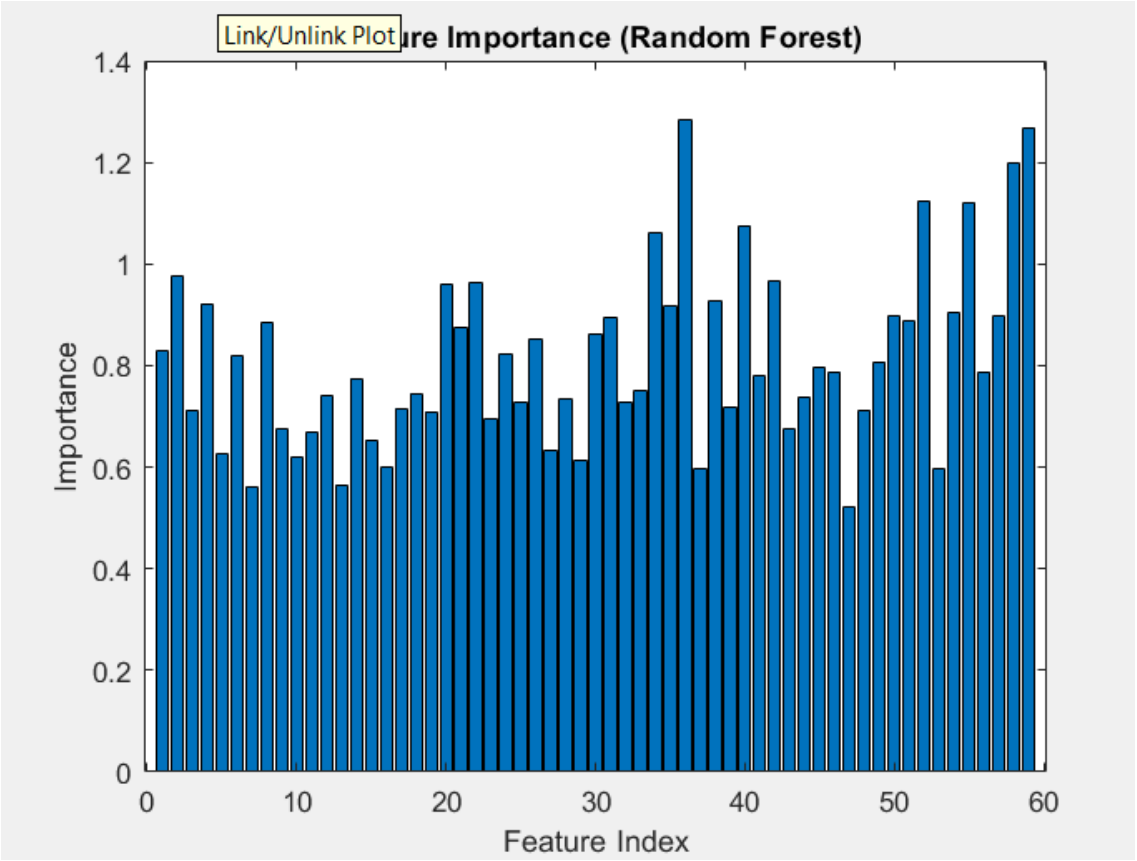

Figura 26: Feature imporance amb RF de 16 bins

Amb el descriptor LBP ha acabat passant una situació similar que amb HOG, on ens pensàvem podria ser bastant útil, però hem pogut comprovar que per si sol no ho és. El motiu, en aquest cas, és que LBP és centra en la identificació de textures mitjançant la comprovació dels píxels veïns donat un de central. Davant sèries de televisió, on la importància la sol donar el color y no tant diferències en textura, provoca que en sèries com Bob Esponja i Southpark amb estils visuals carregats i complexos (que no es limiten a personatges sobre fons senzills), LBP sigui incapaç de realitzar el treball complent.

Per tant quedem amb l'esperança de en una prova posterior sigui d'utilitat i ens aporti informació diferent a les altres tècniques i que al combinar-lo amb alguna altra tècnica obtinguem uns resultats més prometedors.

5.5 Prova 5: Fusió de models

Un cop realitzades totes les proves amb les diferents tècniques de forma individual, concloiem en que per obtenir el model definitiu, provarem combinacions amb les característiques extretes amb els histogrames de color amb HSV, els descriptors del HOG i el LBP.

Pel que fa als models utilizarem KNN i RF, ja que hem vist en les evaluacions

previes que el SVM no obtenia millors resultats que aquests dos en cap execució. Respecte al tamany de les imagtes seguirem utilizant 128*128. Utilitzarem pels dos algorismes els millors paràmetres obtinguts en les evaluacions previes:

- numBinsColor = 16
- numBinsHOG = 16
- cellSize = 32
- veinsLBP = 8
- radiLBP = 1
- veinsKNN = 1

5.5.1 Histogrames amb HSV + HOG

Per provar el model en el que fusionem els histogrames HSV amb els descriptors de HOG, tenim com a característiques les 48 corresponents al HSV i les 576 corresponents al HOG.

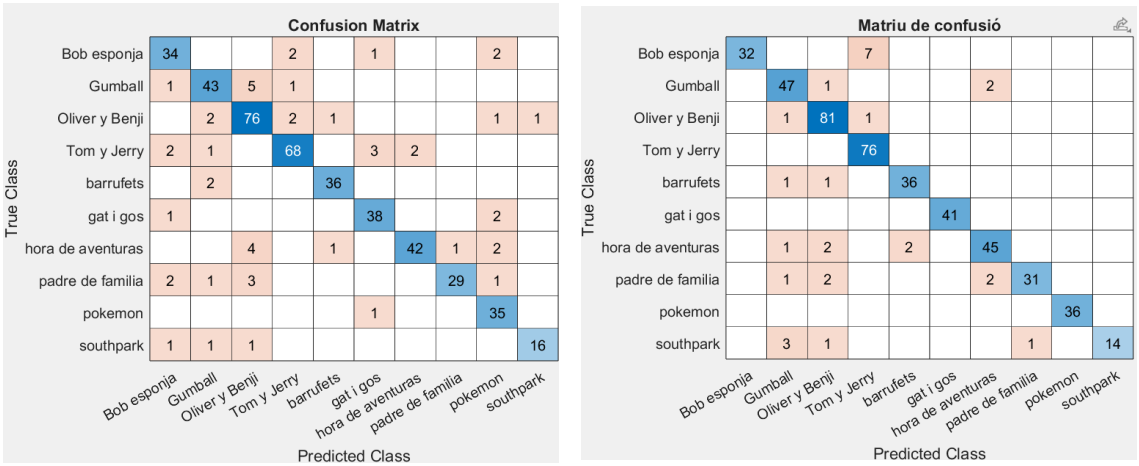

Figura 27: Matrius de confusió per HOG + HSV

Com veiem en les dues matrius de confusió, en els resultats del KNN no obtenim res esperançador de fet tenim un 89% de precissió en el conjunt de prova i por altra banda en el RF tampoc veiem millores respecte els models individuals o el model dels histogrames de color amb HSV tot sol, en aquest tenim un 93.8% de precissió, algo millor que el KNN però no suficient ni molt menys.

	KNN	RF
Accuracy	89%	93.8%

Taula 3: Comparació HOG + HSV

5.5.2 Histogrames amb HSV + LBP

Per provar el model en el que fusionem els histogrames HSV amb els descriptors de HOG, tenim com a característiques les 48 corresponents al HSV i les 59 corresponents a l'LBP.

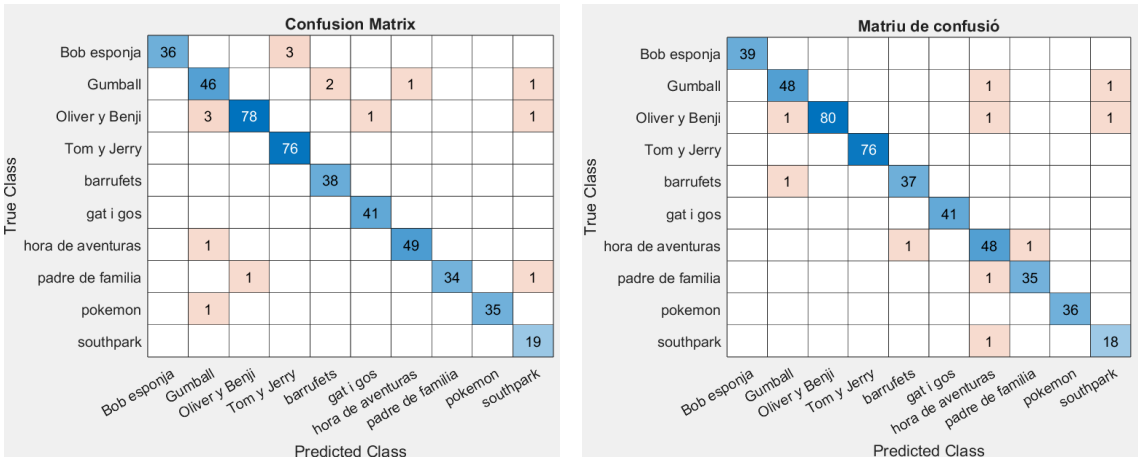

Figura 28: Matrius de confusió per HSV + LBP

Com podem observar, els resultats obtinguts son els millors vists fins al moment, tant en KNN on podem observar que millorem molt en totes les classes tenint entre 1 o 2 errades per classe excepte en la classe Gumball on tenim bastants més errors tant per recall com per precissió, com en RF on aquí si que tenim uns resultats molt bons, la única classe en la que es cometen més errors és la de Hora de aventures on classifiquem erroneament 6 de les seves imatges.

	KNN	RF
Accuracy	96.7%	97.9%

Taula 4: Comparació HSV + LBP

5.5.3 Histogrames amb HSV + HOG + LBP

Finalment provarem amb la combinació de les tres tècniques a veure si la barreja de la informació que ens aporten les caracterísitques de cada una, ajuda o no a millorar les prediccions. En aquest cas tindrem un total de 683 columnes que representen la suma de les 48 dels histogrames de color, les 576 del HOG i les 59 de l'LBP.

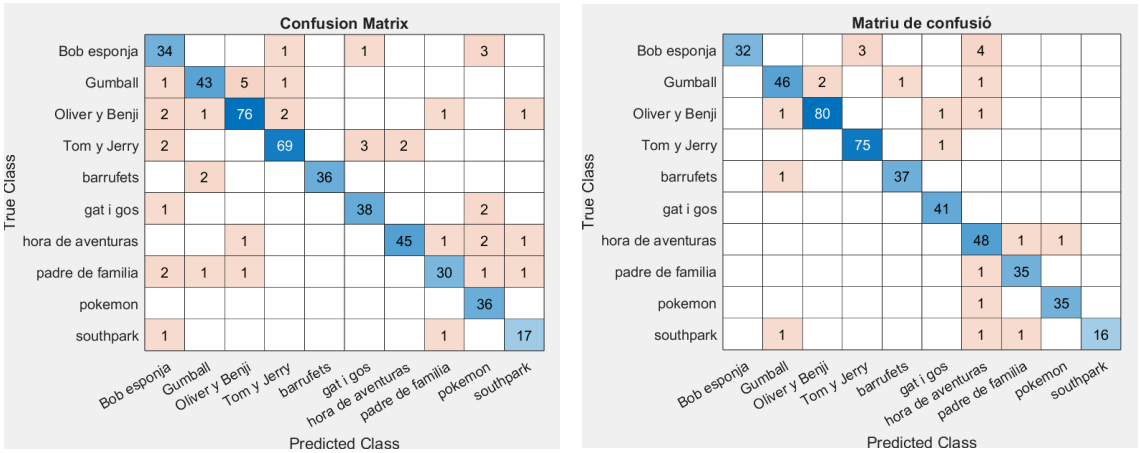

Figura 29: Matrius de confusió per HSV + HOG + LBP

Despres d'aplicar els models, com podem veure no obtenim millora en cap dels resultats, el KNN empitjora bastant en totes les clases i el RF tot i no empitjorar tant segueix sen el pitjor model obtes amb aquest algoritme usant histogrames amb HSV, tot i que és millor que els hisotgrames amb RGB.

	KNN	RF
Accuracy	90.6%	95.1%

Taula 5: Comparació HSV + LBP

5.6 Prova 6: Incorporació d'Oversampling

Finalment, com a darrera prova, vàrem voler incrementar el nombre d'imatges per sèrie realitzant un oversampling. El motiu és senzill, a major nombre d'elements amb que entrenar el model, millor la capacitat de classificació del mateix. A més, ens permetia experimentar la capacitat classificatòria del nostre model davant imatges que no estaven expresament seleccionades, situació similar que sabiem ens trobariem a l'avaluació d'aquesta pràctica.

Per tal de realitzar aquest oversampling, es va fer una recerca, obtenint 20 imatges extra per sèrie, del mateix tamany a les de TRAIN (1920x1080) i del mateix format JPG. A continuació, vam tornar a executar l'script per separar les imatges en *train* i *test*, per tal de tornar a comprovar la versió HSV+LBP, que tans bons resultats ens havia donat en les proves anteriors.

A continuació, els resultats obtinguts:

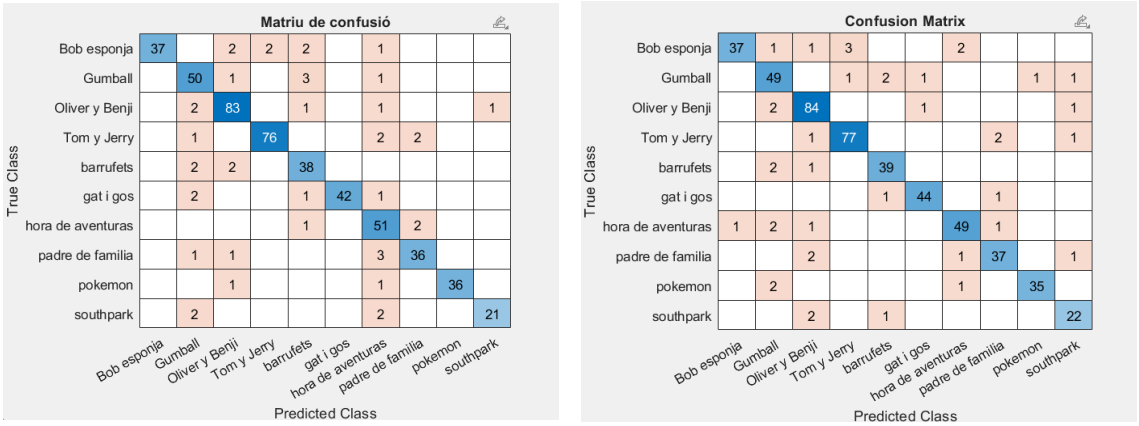

Figura 30: RF i KNN

	KNN	RF
Accuracy	91.44%	92.0%

Taula 6: Comparació HSV + LBP Oversampling

Sorprenentment, el resultat de l'oversampling ha resultat en un petit decrement en la capacitat de classificació del model que tan bons resultats ens havia donat en proves anteriors. Després d'analitzar-ho tot, vam concloure que el motiu que més pes semblava tenir és el fet de que es van incorporar un conjunt d'imatges que encara ser de les diferents sèries, presentaven, algunes diferències significatives respecte a les que teniem com a base d'entrenament. Això ha acbava provocant fallades puntuals generalitzades entre les diferents sèries, ja que els nous fotogrames podien presentar similituds amb altres sèries i no pas la seva. Per tant, podem concloure que, encara poder ser útil realitzar un oversampling per intentar millorar el model, per una banda era un oversampling força petit i, per l'altre banda, afegiem imatges significativament diferents en quant a il·luminació o distribució de figures/-personatges respecte a les d'entrenament que se'ns havien proporcionat ja que com hem comentat les imatges pertanyien a uns frames consecutius d'un mateix episodi i les afegides eren una mostra que representava de manera més genèrica la sèrie, provocant aquesta baixada de precisió.

6 Conclusions

Finalment per concloure amb la investigació de tècniques i combinacions d'aquestes per trobar el millor model per predir donada una imatge a quina sèrie pertany, hem vist que els resultats més prometedors han estat els obtinguts amb la combinació d'-histogrames de color amb HSV i la tècnica de Local Binary Patterns, dues tèniques que ens aporten informació respecte l'estil, el color, la iluminació, la saturació i les textures de la imatge, més que cap intent de segmentació o binartització com hem vist amb el fracàs de SIFT i HOG.

Si analitzem la importancia que el model ha assignat a les diferents característiques, veiem com realment les més importants, segueixen sent les corresponents a l'histograma de color en HSV, que corresponen a les 48 primeres. Les 59 següents que no és que siguin inútils, sino que li donen un valor afegit per discriminar en algunes sèries.

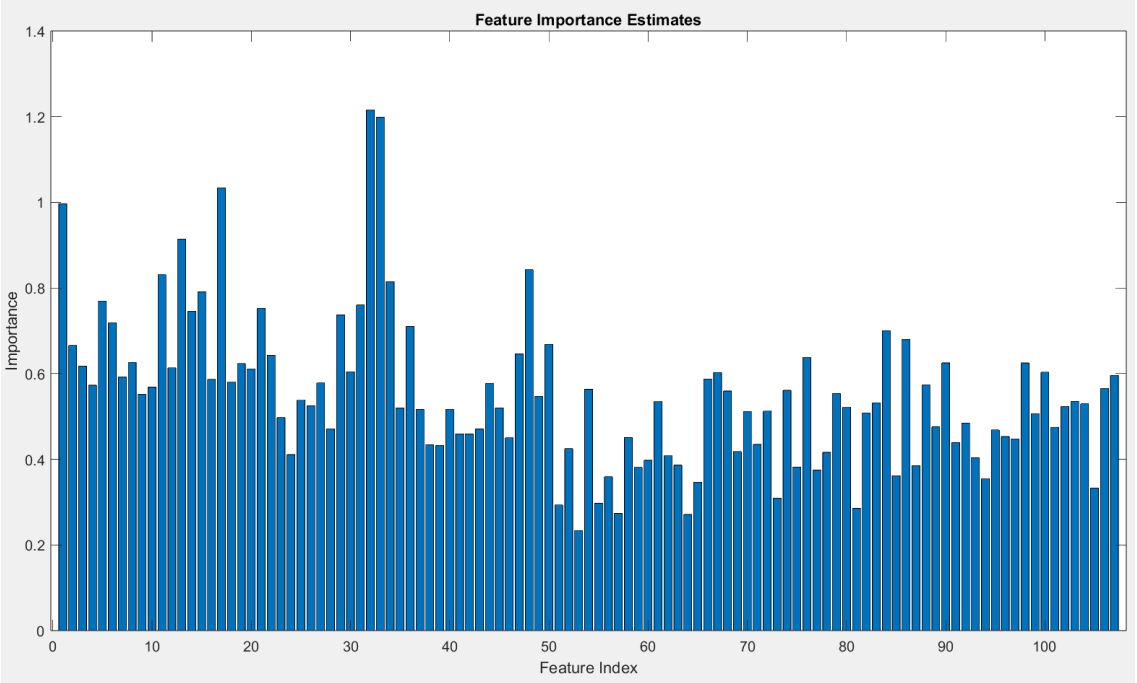

Figura 31: Feature importance en el millor model

6.1 Futures millores

Com en tot projecte, abans d'acabar, ens agradaria expressar quines millores i nous experiments es hagués agradat realitzar si haguessim disposat de més temps.

- **Oversampling**: Com hem comentat en la pràctica, ens hem centrat en maximitzar la predicció de les imatges de les cuals disposavem en el dataset inicial ja que despres de fer uns breus experiments amb un oversampling lleuger per cada sèrie, hem vist com ja ens suposavem que degut a que teniem imatges que corresponien a frames consecutius d'un mateix episodi, si vulguessim extrapolar aquesta detecció a la sèrie en sí, necessitariem un conjunt d'imatges molt més generos, la quantitat de dades de la que es disposa en l'entrenament sempre ha de ser una mostra el màxim de representativa del que és la realitat o del conjunt amb el que s'experimentarà amb el model entrenat.

- **Diferents escalats**: Com hem comentat en el nostre cas degut a que les imatges de cada sèrie tenien diferents escalats i en un cas real no sabem en quina mida ens arribarà la imatge i per tant el no tractar el redimensionament de la imatge provocaria errors en l'aplicació de moltes de les tècniques que hem emprat per resoldre el problema, hem decidit utilitzar un escalat a 128

* 128 píxels. Per tant, en futures millores ens agradaria realitzar proves amb imatges amb un escalat més gran per poder extreure més informació de cada imatge i millorar la qualitat de les dades.

- **Exploració més amplia d'hiperparametres**: Com hem explicat, els models emprats disposen de multiples parametres configurables, dels cuals no hi ha un manual de quins valors van millor per segons quin tipus de problemes, el que provoca que sempre s'hagi de fer una amplia exploració d'hiperparametres, no obstant, donat que no era l'objectiu de l'assignatura prioritzar l'aprenentatge d'aquestes tècniques, hem decidit dedicar el temps en explorar tècniques i algorismes propis de visió per computador.
- **Ensamble de diferents models**: En futures versions ens agradaria incorporar ensmables ja siguin d'Stacking o de Voting, en els que usariem un model especialitzat en cada tècnica treballada, un en els histogrames de color, l'altre en LBP, l'altre amb SIFT...

7 Funcions utilitzades

En aquesta secció presentarem de forma organitzada les diferents funcions que hem utilitzat desenvolupant el nostre codi. Encara que la major part del codi son bucles i funcions bàsiques, les relacionades amb descriptors i models cal especificar-les ja que les hem abstret i utilitzat del Matlab:

- Per calcular els hitogrames de colors, tant RGB com HSV, hem utilitzat la funció ***imhist*** [1].
- Per calcular els descriptors de HOG, hem utilitzat la funció ***extractHOGFeatures*** [2].
- Per calcular els descriptors de textura de LBP, hem utilitzat la funció ***extractLBPFeatures*** [3].
- Per entrenar els models Random Forest, hem utiltizat la funció ***TreeBagger*** [4].
- Per entrenar els models K-Nearest Neighbor, hem utilitzat la funció ***fitcknn*** [5].
- Per entrenar els models Support Vector Machines, hem utilitzat la funció ***fitcecoc*** [6].
- Per entrenar els models SIFT, hem utilitzat la funció ***detectSIFTFeatures*** [7].

Pel que fa al software, principalment hem utilitzat **LaTeX** [8] com a editor de textos, ***Matlab***[9] per a escriure el programa i ***Gogle Drive*** per mantenir constància de la resta de la documentació i les diferents versions de codi realitzades.

8 Estructura del projecte

Per tal de realitzar la pràctica hem utilitzat els següents arxius:

- **main.m**: arxiu final que agafa el millor model obtingut i espera una imatge d'entrada i retorna la sèrie a la que pertany.
- **substracció_soroll.m**: arxiu emprat per eliminar el soroll de les imatges de pokemon i southpark per utilitzar SIFT.
- **split_train_test.m**: arxiu emprat per dividir les dades en train i test.
- **sift.m**: arxiu emprat per dur a terme la tècnica de SIFT.
- **lbp.m**: arxiu emprat per dur a terme la tècnica d'LBP.
- **hog.m**: arxiu emprat per dur a terme la tècnica dels HOG.
- **histogrames_color_RGB.m**: arxiu emprat per realitzar els histogrames de color amb RGB.
- **histogrames_color_HSV.m**: arxiu emprat per realitzar els histogrames de color amb RGB.
- **optimitzacio_models.m**: arxiu de proves i per optimització de models i parametres.
- **fusio_models.m**: arxiu emprat per decidir el millor model i guardar-lo en un arxiu.
- **treeBaggerModel.mat**: model final.

Per tal de realitzar la pràctica hem utilitzat les següents carpetes (que no s'inclouran totes en la entrega per capacitat i redundància):

- **DATA**: carpeta que conté les imatges dividides en train i test.
- **TRAIN**: carpeta que conté les imatges originals.
- **OVERSAM**: carpeta que conté les imatges per l'oversampling.

9 Bibliografia

En aquest darrer apartat afegim referències a les diferents fonts de consulta o software que hem utiltizat per realitzar aquest treball. Principalment hem utiitzat la documentació de matlab ja que no es un llenguatge en el que cap dels dos havia tingut pràctica i ens ha estat de gran ajuda.

Referències

[1] MathWorks. (s.f.). *imhist*. Recuperado de https://es.mathworks.com/help/images/ref/imhist.htm

[2] MathWorks. (s.f.). *extractHOGFeatures*. Recuperado de https://es.mathworks.com/help/vision/ref/extracthogfeatures.html

[3] MathWorks. (s.f.). *extractLBPFeatures*. Recuperado de https://es.mathworks.com/help/vision/ref/extractlbpfeatures.html

[4] MathWorks. (s.f.). *TreeBagger*. Recuperado de https://es.mathworks.com/help/stats/treebagger.html

[5] MathWorks. (s.f.). *fitcknn*. Recuperado de https://es.mathworks.com/help/stats/fitcknn.html

[6] MathWorks. (s.f.). *fitcecoc*. Recuperado de https://es.mathworks.com/help/stats/fitcecoc.html

[7] MathWorks. (s.f.). *detectSIFTFeatures*. Recuperado de https://es.mathworks.com/help/vision/ref/detectsiftfeatures.html

[8] Overleaf. (s.f.). *Learn Overleaf*. Recuperado de https://www.overleaf.com/learn

[9] MathWorks. (s.f.). *MATLAB*. Recuperado de https://es.mathworks.com/products/matlab.html