

TFG – Arquitectura de computadors i sistemes operatius

***FITA#08: Optimització, conclusions i
consideracions finals.***

Gestió del projecte a GitHub:

Branca del repositori:

<https://github.com/UOC-Assignments/uoc.tfg.jbericat/tree/FITA%2308>

Dashboard de seguiment de les tasques associades a la fita:

<https://github.com/UOC-Assignments/uoc.tfg.jbericat/projects/11>

Estudiant: Jordi Bericat Ruz

Professor col·laborador: Daniel Rivas Barragan

Semestre: Tardor 2021/22 (Aula 1)

Versió: ESBORRANY_v5

Index

8. Prova de concepte: Conclusions, millores i consideracions finals	1
8.1 –Primera execució de la prova de concepte	2
8.1.1 – Conclusions obtingudes després de la primera execució de la prova.....	2
8.1.1.1 – Problemes de disseny	3
8.1.1.2 – Problemes amb el dataset i les característiques de les imatges	4
8.1.1.3 – Verificació del Model CNN entrenat	5
8.1.1.4 – Problemes amb la inferència del model	5
8.1.2 – recol·lecció de feedback i implementació de millores	6
8.1.2.1 – Dataset d’imatges.....	6
8.1.2.2 – Algorisme d’entrenament	7
8.1.2.3 – Algorisme d’inferència	7
8.2 – Segona execució de la prova de concepte	9
8.2.1 – Preparació de la prova	9
8.2.1.1 - Entrenament del model amb el nou dataset	9
8.2.2 - Inferència del model sobre el mateix conjunt d’imatges d’explotació utilitzat a la primera PdC	13
8.2.3 – Conclusions obtingudes després de la segona execució de la prova	14
8.2.3 – recol·lecció de feedback i implementació de millores	19
8.2.3.1 – Dataset d’imatges.....	19
8.2.3.2 – Algorisme d’entrenament	19
8.2.3.3 – Algorisme d’inferència	20
8.3 – Tercera execució de la prova de concepte.....	20
8.3.1 – Preparació de la prova	20
8.3.1.1 - Entrenament del model amb l’algorisme revisat a la secció 8.2.2.2	20
8.3.2 – Inferència de les dades	24
8.3.3 – Conclusions obtingudes després de la tercera execució de la prova.....	24

8.4 – Conclusions finals	28
8.5 – Consideracions addicionals al respecte de l'abast, objectius i compromisos establerts durant la planificació inicial d'aquest projecte	29

8. Prova de concepte: Conclusions, millores i consideracions finals

8.1 –Primera execució de la prova de concepte

VEURE SECCIÓ 7 (dues línies)

8.1.1 – Conclusions obtingudes després de la primera execució de la prova

Un cop obtinguts els resultats de la classificació¹ de les imatges obtingudes durant els vols de reconeixement del dron, s'observa que, a diferència del rati del 86% d'encert que s'havia aconseguit un cop entrenat el model (veure [secció 6.3](#)), aquest cop les prediccions són incorrectes en la gran majoria de casos, i sense presentar un criteri clar a l'hora de determinar-ne la seva classe. A tall d'exemple, a la figura 8.01 s'observa com la classificació es realitza gairebé d'una manera aleatòria. Recordem que cada imatge es marca amb vores de color verd, groc i vermell per a incendis d'alta, baixa intensitat, o no-incendis, respectivament (veure [secció 7.x.x.x](#)):

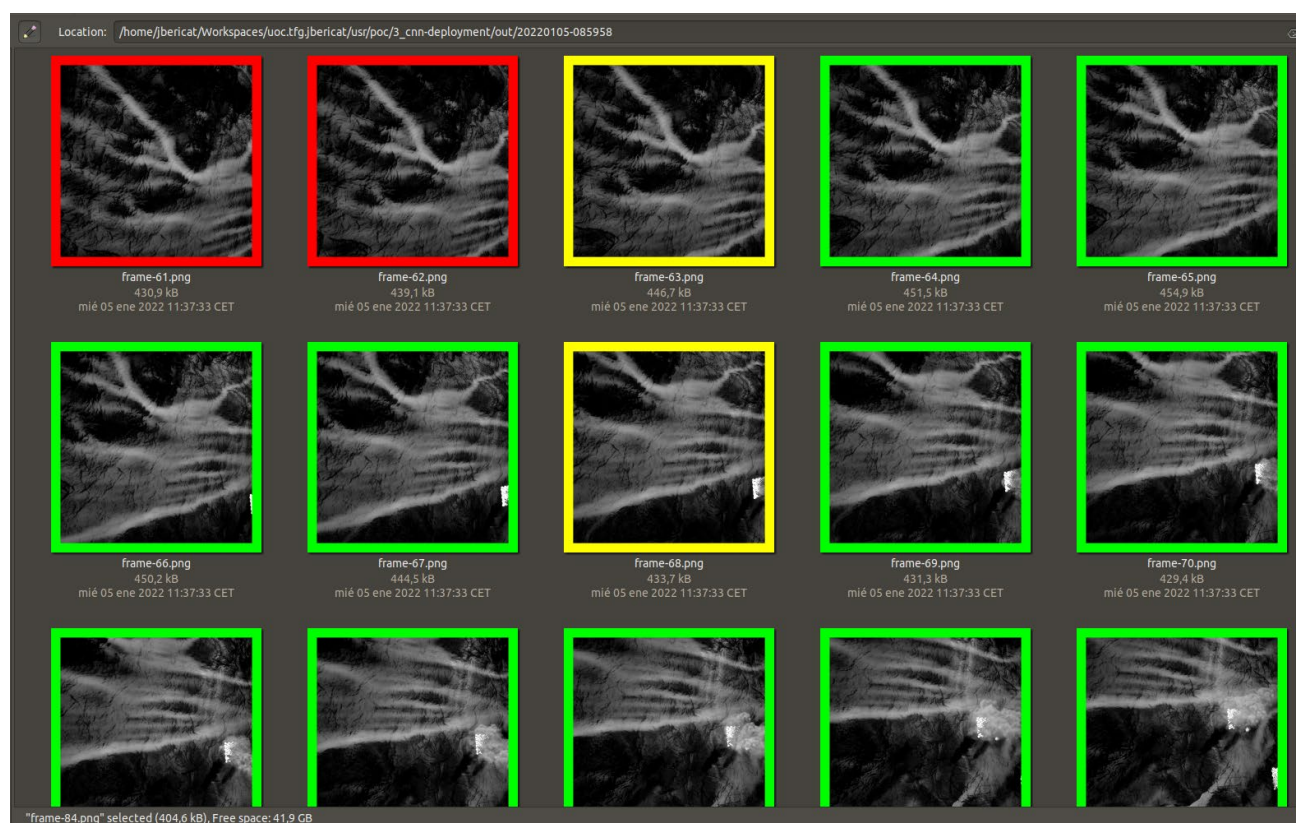


Figura 8.01 – Mostra representativa dels resultats de classificació obtinguts durant la execució de la primera PdC. La col·lecció completa amb tots els resultats es pot trobar al repositori git del projecte:
`/usr/poc/3_cnn-deployment/out/20220105-085958/`

¹ A la versió [v07.xx \(LINK BRANCH REPO\)](#) del branch "FITA#07" de l'algorisme d'inferència `pytorch_deployment.py`, la classificació de les imatges es realitza per lots de 32 o 64 imatges i utilitzant la GPU mitjançant directrius CUDA de pytorch.

Concretament, en aquesta petita mostra observem:

- Els frames 61 a 65 són imatges sense incendis (`no-wildfires`), però només trobem dues prediccions correctes (`frame-64.png` i `frame-65.png`)
- La resta de frames (66 a 75) contenen incendis d'alta intensitat però no n'hi ha cap que sigui reconegut com a tal pel model.

Per tant, podem concloure que durant el *pipeline* d'execució de la primera PdC han esdevingut un o més problemes que han influït negativament en els resultats de la classificació. Si estudiem per separat tots els punts crítics de la cadena d'execució de la PdC, podem determinar que el problema ha d'estar en algun d'aquest 4 punts (o bé, en més d'un):

1. Es un problema de disseny: les característiques de les imatges FLIR simulades no són adequades per a l'entrenament del model
2. El problema rau en les característiques del dataset d'imatges i en les característiques d'aquestes
3. El model no ha restat adequadament entrenat degut a una errada de disseny de la seva arquitectura, o bé per una errada d'implementació de l'algorisme d'entrenament implementat durant la **secció 5.3**
4. L'algorisme d'inferència implementat a la **secció 7.3.3** conté errades de disseny o implementació

8.1.1.1 – Problemes de disseny

Pel que fa la part de visió per computador, de partida es descarta la existència de cap problema de disseny, ja que haver de redissenyar la part de simulació de les imatges tindria un efecte “cascada” sobre la resta del projecte que suposaria una alta càrrega de feina. Per aquest motiu, es considera oportú explorar altres vies primer.

D'altra banda però, pel que fa al disseny del pipeline d'execució de la primera PdC, s'ha trobat que la generació de fitxers a la estructura de directoris dissenyada per a la obtenció de resultats

(carpeta /usr) es realitza d'una manera poc predictiva i usable que podria dificultar la feina a posteriors investigadors que basin el seu treball en aquest projecte. Per tant, es determina que caldrà redissenyar la estructura de directoris i els codi font relacionat, de manera que es podran efectuar les següents PdC d'una manera més intuïtiva i automatitzada.

8.1.1.2 – Problemes amb el dataset i les característiques de les imatges

En aquest punt, durant una revisió del dataset v9.0² utilitzat per a realitzar la primera PdC, es consideren les següents observacions:

- **Cal ampliar i enriquir el subgrup d'imatges sense incendis** (de classe “no wildfires”), ja que el model pot estar confonent les imatges que contenen molt de fum, amb les que contenen el reflex de la neu que hi ha a les muntanyes de l'entorn³.
- La quantitat d'imatges de cada subgrup **pot no estar apropiadament balancejada** degut als canvis a darrera hora introduïts a la **secció 6.2.2.7** (unió de les classes d'incendis `medium-intensity` i els `low-intensity` en una mateixa classe)
- La **riquesa i varietat d'imatges** en una mateixa classe podria no estar representant adequadament el tipus d'imatges que recull el dron durant el vol de reconeixement. En aquest sentit, com les imatges es prenen en el mateix angle (perpendicular a terra), aleshores podem concloure que ampliant la quantitat de distàncies a les que es prenen les imatges per a l'entrenament es podria millorar el rendiment del model durant la seva explotació
- En el mateix sentit que al punt anterior, tenim que la quantitat d'imatges requerida per a entrenar un model que pugui arribar a ser en certa mesura productiu es necessita una **quantitat d'imatges** major que les utilitzades per a entrenar el model de la primera PdC.

² Es pot consultar el dataset d'imatges v9.0 a la següent **ruta del repositori git**:

³ Per a realitzar aquesta PdC s'ha utilitzat íntegrament programari llibreria en codi obert (excepte per a tasques d'edició de textos i vídeo no relacionades amb el desenvolupament de la mateixa). Per aquest motiu s'ha hagut d'utilitzar un entorn virtual que no era perfectament adequat per a simular a les condicions de sequera, temperatura, etc comuns en incendis forestals. En aquest sentit, tot i que s'ha fet tot el possible per adequar l'entorn durant la secció 3 d'aquest projecte i fer-lo el més apte possible, pel que fa als objectes que simulen neu no s'ha pogut fer res (fet que implica que algunes imatges amb molta neu siguin potencialment confoses per imatges amb molt de fum, per part del model).

Per tant, també seria adient ampliar el catàleg de tècniques d'ampliació proposades a la **secció 5.2.1**, per exemple, afegint la tècnica del *shear* o deformació d'imatges.

- El fet de no haver realitzat un “**curat**” **manual** de les imatges un cop feta la recollida automàtica d'imatges del dataset a la **secció 4.7** pot ser un dels motius principals pels quals el rendiment del model cau tant durant la explotació, ja que del conjunt de test tampoc no se'n havia fet, de curat manual. Per tant, caldria per una repassada dels “thumbnails” de cada classe i eliminar aquelles imatges no representatives (sobretot de la classe `no-wildfires`, ja que la resta disposa de posicionament automàtic el qual no tendeix a fallar). Com a conseqüència, podria ser que s'estigui sobre-entrenant el model per a encaixar amb les característiques del conjunt de test però no per a satisfer les del conjunt d'explotació.

8.1.1.3 – Verificació del Model CNN entrenat

Durant una primera revisió de l'algorisme d'entrenament⁴ a priori es verifica que el procés d'entrenament es realitza correctament i de partida no es valora efectuar cap canvi en el mateix. Tanmateix, com s'ha comentat al punt anterior, caldrà ampliar les tècniques de augmentació, les quals es realitzen en temps d'execució al realitzar l'entrenament del model.

D'altra banda, pel que fa al conjunt de test es considera desactivar les augmentacions per tal de no perdre característiques i no perdre precisió durant les avaluacions, tal i com s'ha mencionat anteriorment.

8.1.1.4 – Problemes amb la inferència del model

S'observen diferents problemàtiques a adreçar en l'algorisme d'inferència⁵ desenvolupat a la **secció 7.3.3**:

⁴ Es pot consultar l'estat del codi d'aquesta versió al branch del repositori corresponent, concretament la revisió v07.27 al següent enllaç → https://github.com/UOC-Assignments/uoc.tfg.jbericat/blob/FITA%2307/src/poc/cnn-training/pytorch_training.py

⁵ Es pot consultar l'estat del codi d'aquesta versió al branch del repositori corresponent, concretament la revisió v07.29 al següent enllaç → https://github.com/UOC-Assignments/uoc.tfg.jbericat/blob/FITA%2307/src/poc/cnn-deployment/pytorch_deployment.py

- **Algorisme d'inferència:** S'observen algunes anomalies al codi general de l'algorisme d'inferència que podrien estar afectant els resultats. Per tant, es decideix re-dissenyar el mateix per complet, tot aprofitant aquelles parts vàlides de l'anterior.
- **Informació de sortida:** No s'està adjuntant la puntuació de probabilitat de classificació en cada predicció, fet que impedeix fer un anàlisi més profund de les causes de les males prediccions. També seria adjuntar una taula de confusió als resultats.
- **Procés per lots de les imatges:** Després de revisar més a fons els resultats de la classificació, s'observa que la sortida no sempre respecta l'ordre seqüencial de les imatges. D'altra banda, per tal de fer una simulació més fidel de la recollida i anàlisi d'imatges en temps real per part dels drons, es considera alimentar el model de manera serialitzada amb lots d'una sola imatge.
- **Utilització de la GPU per a la inferència:** Tot i estar alimentant el model amb lots d'entre 32 i 64 imatges per lot, el fet d'estar realitzant la inferència mitjançant un dispositiu GPU en comptes de una CPU tradicional, implica que s'hagin de realitzar tantes transferències de memòria com lots resultants. Aquest fet fa que l'aprofitament d'un dispositiu GPU per a fer la inferència no sigui molt millor que el que realitzaria un dispositiu CPU durant l'execució de la PdC. A més, aquest *drawback* en el temps de transferència s'accentua en el cas de serialitzar l'anàlisi d'imatges per part del model. Així doncs, tindrem en compte tots aquests detalls i realitzarem una versió de l'algorisme que no requereixi GPU per a fer la inferència (d'aquesta manera es pot reduir el cost dels drons de reconeixement en cas que aquest no requereixin característiques de procés en real-time imperatiu (com vindria a ser en el sector automobilístic). En cas però que s'haguessin d'equipar els drons amb dispositius GPU, existeixen solucions Edge Computing de tipus encastada (embedded) com per exemple les SoC Jetson nano de NVIDIA o els acceleradors DLHA de hardware específics per a aplicacions basades en Deep Learning, com les TDA4VM de Texas Instruments.

8.1.2 – **recol·lecció de feedback i implementació de millores**

Donat l'alt grau d'escalabilitat del *pipeline* dissenyat per a la realització de la primera PdC, podem efectuar una segona PdC amb relativa rapidesa un cop aplicades les consideracions anteriors. Per tant, es procedeix a realitzar els canvis següents per tal d'efectuar una segona prova de concepte:

8.1.2.1 – Dataset d'imatges

En primer lloc es procedeix a generar tot un seguit de datasets d'imatges fins assolir-ne un que

inclou totes les millors observades al punt anterior. El nou dataset (**v12.0**) es pot trobar a la ruta del repositori indicada a peu de la figura 08.xx:

Name	Size	Type	Date Modified
bin	1 item	folder	dom 16 ene 2022 16:47:18 CET
doc	7 items	folder	sáb 15 ene 2022 17:29:18 CET
src	5 items	folder	sáb 15 ene 2022 23:22:37 CET
usr	3 items	folder	dom 16 ene 2022 16:47:44 CET
archive	1 item	folder	vie 29 oct 2021 21:07:35 CEST
packages	10 items	folder	vie 19 nov 2021 07:52:54 CET
poc	3 items	folder	dom 16 ene 2022 16:21:44 CET
1_datasets	2 items	folder	sáb 15 ene 2022 23:19:38 CET
curated-data	7 items	folder	sáb 15 ene 2022 22:59:22 CET
v4.0	2 items	folder	mié 05 ene 2022 11:37:29 CET
v5.0	2 items	folder	mié 05 ene 2022 11:37:29 CET
v6.0	2 items	folder	mié 05 ene 2022 11:37:29 CET
v7.0	2 items	folder	mié 05 ene 2022 11:37:30 CET
v8.0	2 items	folder	mié 05 ene 2022 11:37:30 CET
v9.0	2 items	folder	mié 05 ene 2022 11:37:31 CET
v12.0	2 items	folder	sáb 15 ene 2022 17:29:36 CET
test	3 items	folder	sáb 15 ene 2022 17:29:36 CET
high-intensity-wildfires	76 items	folder	sáb 15 ene 2022 17:29:35 CET
low-intensity-wildfires	382 items	folder	sáb 15 ene 2022 17:29:36 CET
no-wildfires	171 items	folder	sáb 15 ene 2022 17:29:36 CET
training+validation	3 items	folder	sáb 15 ene 2022 17:29:36 CET
high-intensity-wildfires	815 items	folder	sáb 15 ene 2022 17:29:36 CET
low-intensity-wildfires	1.046 items	folder	sáb 15 ene 2022 17:29:36 CET
no-wildfires	1.249 items	folder	sáb 15 ene 2022 17:29:37 CET

Figura 08.xx – Generació del dataset v12.0 que s'utilitzarà per a la execució de la segona PdC.
 Ruta al repositori git: `/usr/poc/1_datasets/curated-data/v12.0`

8.1.2.2 – Algorisme d'entrenament

Un cop desactivades les funcionalitats d'augmentació del catàleg d'imatges per al subgrup grup de test, la nova versió de l'algorisme d'entrenament realitzada es pot trobar a la **revisió v08.5** del repositori:

<https://github.com/UOC-Assignments/uoc.tfg.jbericat/tree/47de41d3b1963f47ea1805faa50503f2720f64af/src/poc>

8.1.2.3 – Algorisme d'inferència

Tal i com s'ha indicat a la **secció 8.1**, un dels cursos d'acció necessaris per tractar de millorar els resultats de la classificació consisteix en el fet de re-escriure el codi de l'algorisme d'inferència per tal que aquesta es realitzi de manera serialitzada. D'altra banda, un altra modificació que s'ha al

codi del mateix algorisme ha sigut la de realitzar l'etiquetat de cadascuna de les imatges un cop classificades (mitjançant la llibreria matplotlib per a python), de la manera com segueix:



Figura 8.xx – Format de l'etiquetat afegit a les imatges un cop realitzada la inferència – `pytorch_deployment.py`

La nova versió de l'algorisme d'inferència realitzat es pot trobar a la **revisió v08.7** del branca **FITA#08** del repositori git:

https://github.com/UOC-Assignments/uoc.tfg.ibericat/blob/FITA%2308/src/poc/3_cnn-deployment/pytorch_deployment.py

8.2 – Segona execució de la prova de concepte

8.2.1 – Preparació de la prova

8.2.1.1 - Entrenament del model amb el nou dataset

En aquest punt podem procedir a tornar a entrenar el mateix model de xarxa neuronal que es va utilitzar a la primera prova, però en aquest cas utilitzant el dataset d'imatges millorat. Ho farem mitjançant la execució de l'script `pytorch_training.sh`:

```
(base) jbericat@TFG-UOC:~/Workspaces/uoc.tfg.jbericat/src/poc/2_cnn-training$ ./pytorch_training.sh

*****
2. CNN Training Algorhythm
*****

Set the model version you want to train:
1 = v1.0 -> 3 layers CNN (simplified LeNet)
2 = v2.0 -> 5 layers CNN (LeNet)
3 = v3.0 -> 14 layers CNN (custom)

Please choose an option (1-3 - Default = 3): 3
Set the dataset version you want to use to train the model:
4 = v4.0 -> ARCHIVED (EXPERIMENTAL)
5 = v5.0 -> ARCHIVED (EXPERIMENTAL)
6 = v6.0 -> ARCHIVED (EXPERIMENTAL)
7 = v7.0 -> 600 samples: 3 classes, close distance images (SMALLEST DATASET, appropriate for adjusting parameters)
8 = v8.0 -> 1.5K samples: 4 classes, close distance images
9 = v9.0 -> 1.5K samples: 3 classes, close distance images
10 = v10.0 -> ARCHIVED (EXPERIMENTAL)
11 = v11.0 -> ARCHIVED (EXPERIMENTAL)
12 = v12.0 -> 3.7K samples: 3 classes, close, mid & long distance images (Improved no-wildfire class images).

Please choose an option (4-12 - Default = 7): 12
[INFO] generating the train/validation split...
[INFO] Training the convolution neural network model, hold-on tight...! call last!
File ~/home/jbericat/workspaces/uoc.tfg.jbericat/src/poc/2_cnn-training/pytorch_training.py, line 87, in <module>
  MODEL_VERSION = input("Please choose an option (1-3 - Default = 3): ") or "3"
KeyboardInterrupt

jbericat@TFG-UOC:~$ jbericat@TFG-UOC:~/Workspaces/uoc.tfg.jbericat/src/poc/2_cnn-training$ jbericat@TFG-UOC:~/Workspaces/uoc.tfg.jbericat/src/poc/3_cnn-deployment$
```

Figura 8.xx – Execució de l'script d'entrenament del model `pytorch_training.sh`:

Un cop entrenat el model⁶ tot utilitzant el dataset v12.0 (que inclou les modificacions realitzades a la [secció 8.2](#)), els resultat obtinguts milloren els del model utilitzat per a la primera PdC.

Concretament tenim que ara s'assoleix una **precisió (accuracy) de fins 92%** en les prediccions realitzades contra el dataset de test, xifra que millora el 86% obtingut pel model utilitzat durant la primera PdC. Les figures [8.xx](#) a, b i c mostren el resum d'entrenament de la xarxa neuronal donades aquestes noves condicions:

⁶ A la [secció 7.4.5](#) s'han presentat tot un seguit d'scripts bash que permeten la execució àgil de l'entrenament mitjançant la càrrega del codi python a l'entorn de llibreries adequat de manera totalment transparent.

```

*****
***                                     ***
***                               PoC's CNN Model Training Summary                               ***
***                                     ***
*****
***                                     ***
***                               Model version: v3.0                                       ***
***                               Dataset version: v12.0                                   ***
***                                     ***
*****

TRAINING PARAMETERS:

- Image size = (229 x 229 x 1)
- Number of classes / labels = 3
- Batch size = 128
- Learning rate = 0.0001

*****

DATASET TOTALS:

- The number of images in a training set is: 11776
- The number of images in a validation set is: 3968
- The number of images in a test set is: 640
- The number of batches per epoch is: 92

*****

CNN BLUEPRINT:

Network_v3(
  (conv1): Conv2d(1, 16, kernel_size=(9, 9), stride=(2, 2), padding=(2, 2))
  (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv2): Conv2d(16, 16, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
  (bn2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (pool): MaxPool2d(kernel_size=2, stride=1, padding=0, dilation=1, ceil_mode=False)
  (conv3): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(2, 2))
  (bn3): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv4): Conv2d(32, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (bn4): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (fc1): Linear(in_features=107648, out_features=3, bias=True)
)

*****

MODEL TRAINING STATS:

- Model trained on cuda:0 device
- Training time: 10767.402 seconds

*****

MODEL TESTING STATS:

- For epoch 1 the TEST accuracy over the whole TEST dataset is 80 %
- For epoch 2 the TEST accuracy over the whole TEST dataset is 73 %
- For epoch 3 the TEST accuracy over the whole TEST dataset is 84 %
- For epoch 4 the TEST accuracy over the whole TEST dataset is 82 %
- For epoch 5 the TEST accuracy over the whole TEST dataset is 87 %
- For epoch 6 the TEST accuracy over the whole TEST dataset is 76 %
- For epoch 7 the TEST accuracy over the whole TEST dataset is 76 %
- For epoch 8 the TEST accuracy over the whole TEST dataset is 71 %
- For epoch 9 the TEST accuracy over the whole TEST dataset is 86 %
- For epoch 10 the TEST accuracy over the whole TEST dataset is 92 %
- For epoch 11 the TEST accuracy over the whole TEST dataset is 82 %
- For epoch 12 the TEST accuracy over the whole TEST dataset is 81 %
- For epoch 13 the TEST accuracy over the whole TEST dataset is 81 %
- For epoch 14 the TEST accuracy over the whole TEST dataset is 72 %
- For epoch 15 the TEST accuracy over the whole TEST dataset is 88 %

```

Figura 08.xx.a - Resum d'entrenament del model CNN utilitzant el dataset d'imatges v12.0

- For epoch 16 the TEST accuracy over the whole TEST dataset is 87 %
- For epoch 17 the TEST accuracy over the whole TEST dataset is 72 %
- For epoch 18 the TEST accuracy over the whole TEST dataset is 87 %
- For epoch 19 the TEST accuracy over the whole TEST dataset is 86 %
- For epoch 20 the TEST accuracy over the whole TEST dataset is 91 %

FINAL PREDICTION TEST:

- Model tested on cuda:0 device
- Final test accuracy: 92 %

Real Label	Predicted Label	Output Tensor
no-wildfires	no-wildfires	[-2.3327, -2.5394, 5.2414]
no-wildfires	no-wildfires	[-2.4208, -2.0767, 5.3927]
high-intensity-wildfires	no-wildfires	[0.2980, -2.6518, 1.3574]
low-intensity-wildfires	low-intensity-wildfires	[-9.1065, 9.9720, -4.7903]
low-intensity-wildfires	low-intensity-wildfires	[-3.5871, 8.7342, -7.2828]
no-wildfires	no-wildfires	[-0.9034, -6.6263, 5.0902]
low-intensity-wildfires	low-intensity-wildfires	[-18.0717, 10.7260, 3.5805]
low-intensity-wildfires	low-intensity-wildfires	[-1.4727, 6.1938, -6.5996]
no-wildfires	no-wildfires	[-4.7247, 0.6882, 3.6004]
low-intensity-wildfires	low-intensity-wildfires	[-2.7809, 5.5997, -4.4440]
low-intensity-wildfires	low-intensity-wildfires	[-7.8821, 8.9408, -3.7572]
no-wildfires	no-wildfires	[0.1228, -2.3730, 3.6613]
low-intensity-wildfires	low-intensity-wildfires	[-1.8697, 2.1304, -0.3099]
low-intensity-wildfires	high-intensity-wildfires	[21.0831, -4.5426, -38.1306]
low-intensity-wildfires	low-intensity-wildfires	[-3.5861, 6.5604, -4.2513]
high-intensity-wildfires	low-intensity-wildfires	[-0.1135, 0.6394, -2.1460]

Figura 08.xx.b - Resum d'entrenament del model CNN utilitzant el dataset d'imatges v12.0

```

+-----+-----+-----+
| low-intensity-wildfires | low-intensity-wildfires | [-6.5662, 7.4135, -3.7574] |
+-----+-----+-----+
| low-intensity-wildfires | low-intensity-wildfires | [-2.0723, 1.6515, -0.0409] |
+-----+-----+-----+
| no-wildfires            | no-wildfires            | [-4.6834, -0.0276, 4.5610] |
+-----+-----+-----+
| no-wildfires            | no-wildfires            | [-1.1020, -2.8442, 5.5550] |
+-----+-----+-----+
| no-wildfires            | no-wildfires            | [-2.0979, -4.5407, 5.9977] |
+-----+-----+-----+
| no-wildfires            | no-wildfires            | [-1.8086, -2.1718, 4.7881] |
+-----+-----+-----+
| low-intensity-wildfires | low-intensity-wildfires | [-1.0145, 5.7108, -7.7168] |
+-----+-----+-----+
| low-intensity-wildfires | low-intensity-wildfires | [-2.5241, 2.8725, -0.4059] |
+-----+-----+-----+

```

Figura 08.xx.c - Resum d'entrenament del model CNN utilitzant el dataset d'imatges v12.0

Tot seguit, a les figures 8.xx i 8.xx es mostren les corbes de pèrdua i d'aprenentatge, respectivament. Pel que fa la primera, s'observa que el fet d'haver realitzat 20 iteracions d'entrenament no era necessari, ja que a partir de la epoch nº 10 es deixen de produir millores significatives (la corba deixa de decreïxer). Això es pot verificar mirant la corba d'aprenentatge, la qual també ens mostra que el millor rendiment obtingut s'assoleix a la epoch nº 9:

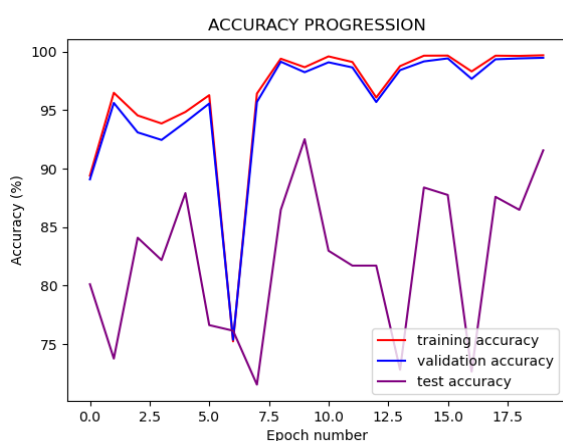


Figura 08.xx - Corba d'aprenentatge del model. S'observa un extrem relatiu a la iteració 9^a d'entrenament, moment en el que s'assoleix la precisió màxima

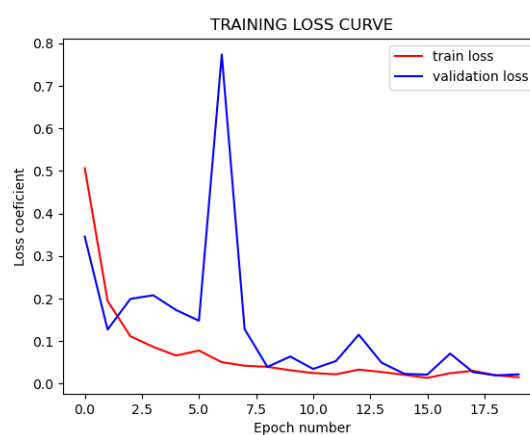


Figura 08.xx - Corba de la funció de pèrdua utilitzada per la funció d'optimització del model durant l'entrenament per al reajust dinàmic de paràmetres. El pic que s'observa a la epoch nº6 podria correspondre a un reajustament (back-propagation) de paràmetres contraproduent amb valors que han sigut descartats a iteracions posteriors

Es pot trobar el fitxer `.pth`⁷ del nou model entrenat, junt als resums d'entrenament i les gràfiques anteriors a la següent carpeta del repositori git del projecte:

`/usr/poc/2_cnn-training/cnn-training_20220114-225947.tar.gz`

8.2.2 - Inferència del model sobre el mateix conjunt d'imatges d'explotació utilitzat a la primera PdC

Finalment podem procedir a realitzar de nou la inferència de les dades per a comprovar si els canvis realitzats a la [secció 8.2](#) han implicat millores en comparació amb les prediccions realitzades a la primera PdC. És important remarcar que les inferències es realitzaran exactament contra el mateix conjunt de dades d'explotació utilitzat durant la primera prova, donat que d'aquesta manera podrem estudiar els casos en què no s'hagi produït cap millora. Això darrer, junt a la inclusió de la puntuació de probabilitats (*energy score*) en els resultats de classificació ens proporcionarà la **traçabilitat** necessària per tal de poder seguir aplicant les millores que es considerin oportunes en futures iteracions d'optimització dels resultats (execucions de PdC i posterior recollida de feedback).

Així doncs, procedim a executar l'algorisme d'explotació del model que s'encarrega de realitzar la inferència mitjançant l'script `pytorch_deployment.sh`⁸:

```
(base) jbericat@TFG-UOC:~/Workspaces/uoc.tfg.jbericat/src/poc/3_cnn-deployment$ ./pytorch_deployment.sh
*****
21) 3. CNN Deployment Algorithm
*****
[INFO] - The number of images in a deploy set is: 601
[INFO] - The batch-size is: 1
[INFO] - Using the model on CPU device to make inferences over the serialized data. This might take a minute or two...
[INFO] - The CNN model deployment has finished successfully. See the output .png files to visually check how accurate the predictions were.
[INFO] - OUTPUT FILES:
1 = 0.0 - 3 layers CNN (simplified LeNet)
2 = 0.0 - Classification results: /home/jbericat/Workspaces/uoc.tfg.jbericat/usr/poc/3_cnn-deployment/out/20220117-010803
3 = 0.0 - 14 layers CNN (custom)
(base) jbericat@TFG-UOC:~/Workspaces/uoc.tfg.jbericat/src/poc/3_cnn-deployment$
```

Figura 8.xx – Execució de l'script d'inferència del model `pytorch_deployment.sh`:

⁷ Extensió utilitzada per models CNN implementats i generats amb pytorch

⁸ Veure [secció 7.4](#) per a tenir una idea general del flux d'execució o pipeline de la prova de concepte

8.2.3 – Conclusions obtingudes després de la segona execució de la prova

Ara, si estudiem una mostra dels resultats de classificació obtinguts en aquesta segona prova podem determinar el grau de millora aproximat aconseguit respecte de la primera prova. Per a fer aquest estudi comparatiu es realitzarà una inspecció visual de la classificació i en aquells casos en que es detectin errades en les prediccions, s'estudiaran les metadades adjuntes a cada predicció (puntuació o *score*) per tal de **determinar la existència de patrons**. A tall d'exemple; si en un conjunt d'imatges que contenen una característica comuna (com pot ser el posicionament d'un incendi massa a la bora de la imatge) mostren a les metadades una diferència molt reduïda entre les puntuacions de probabilitat de dues classes diferents, aleshores podríem mirar de re-entrenar el model tot afegint més imatges que continguin característiques similars (sempre mirant de generalitzar per a que no es produeixi sobre-ajustament o *overfitting* i s'acabin memoritzant les característiques).

En primer terme, si ens fixem en la primera seqüència de frames (figura 08.xx) ens trobem amb què, al contrari del que passava amb la primera prova (en la que obteníem resultats aleatoris), els resultats de classificació semblen seguir un criteri, sigui encertat o no. Concretament, observem que en aquest cas les imatges sense incendi dels frames frame-0-1.png a frame-0-40.png han sigut classificades com a incendis de baixa intensitat, mentre que realment haurien d'haver sigut classificades com a imatges sense incendis.

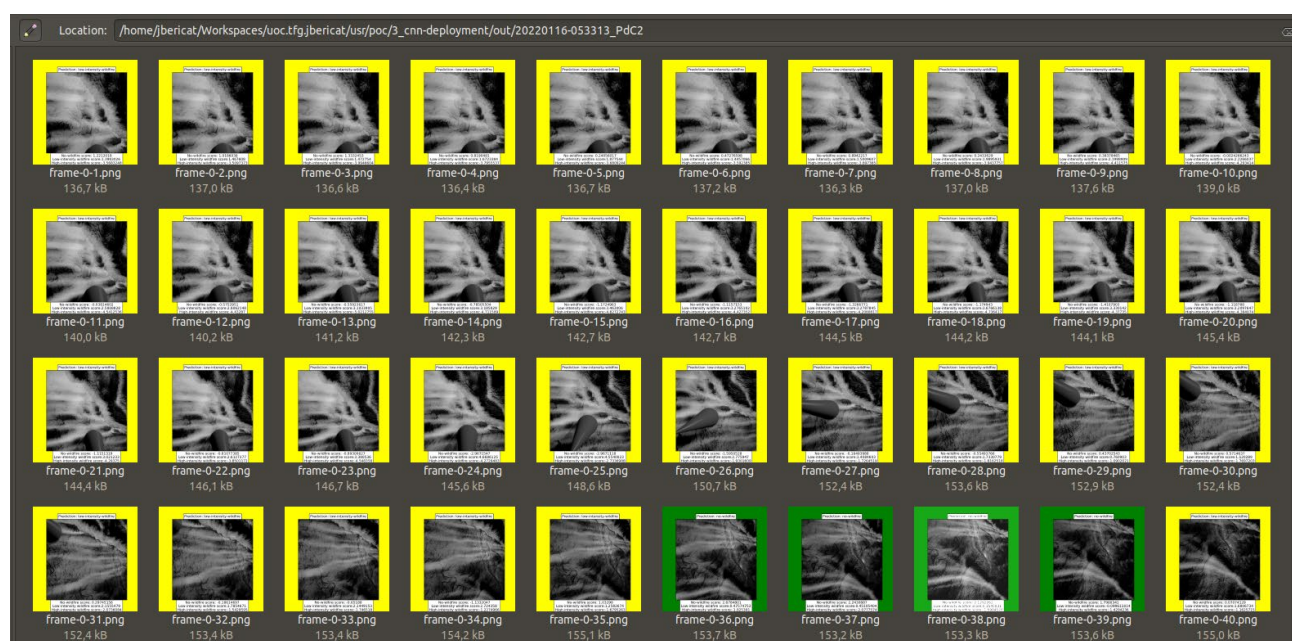


Figura 08.xx.a - Mostra d'imatges de classe *no-wildfire* representativa dels resultats de classificació després de la segona execució de la PdC

D'altra banda, si ens fixem en la seqüència de la figura 08.xx podem observar com, generalment, en aquest cas les prediccions realitzades són en la majoria de casos encertades, excepte:

- Els frames 5 a 9 han sigut classificats com a incendis de baixa intensitat mentre que realment contenen imatges d'incendis d'alta intensitat.
- Els frames 16 i 17 han sigut classificats com a imatges sense incendis, mentre que realment contenen imatges d'incendis d'alta intensitat.
- El frame 37 conté imatges d'incendis d'alta intensitat i ha estat classificat com a imatge sense incendis.



Figura 8.xx.b – Mostra representativa dels resultats de classificació després de la segona execució de la PdC

Per contrapartida, si ens fixem en la mostra de la figura següent, les quals o bé contenen imatges d'incendis d'alta intensitat, o bé no mostren cap incendi:

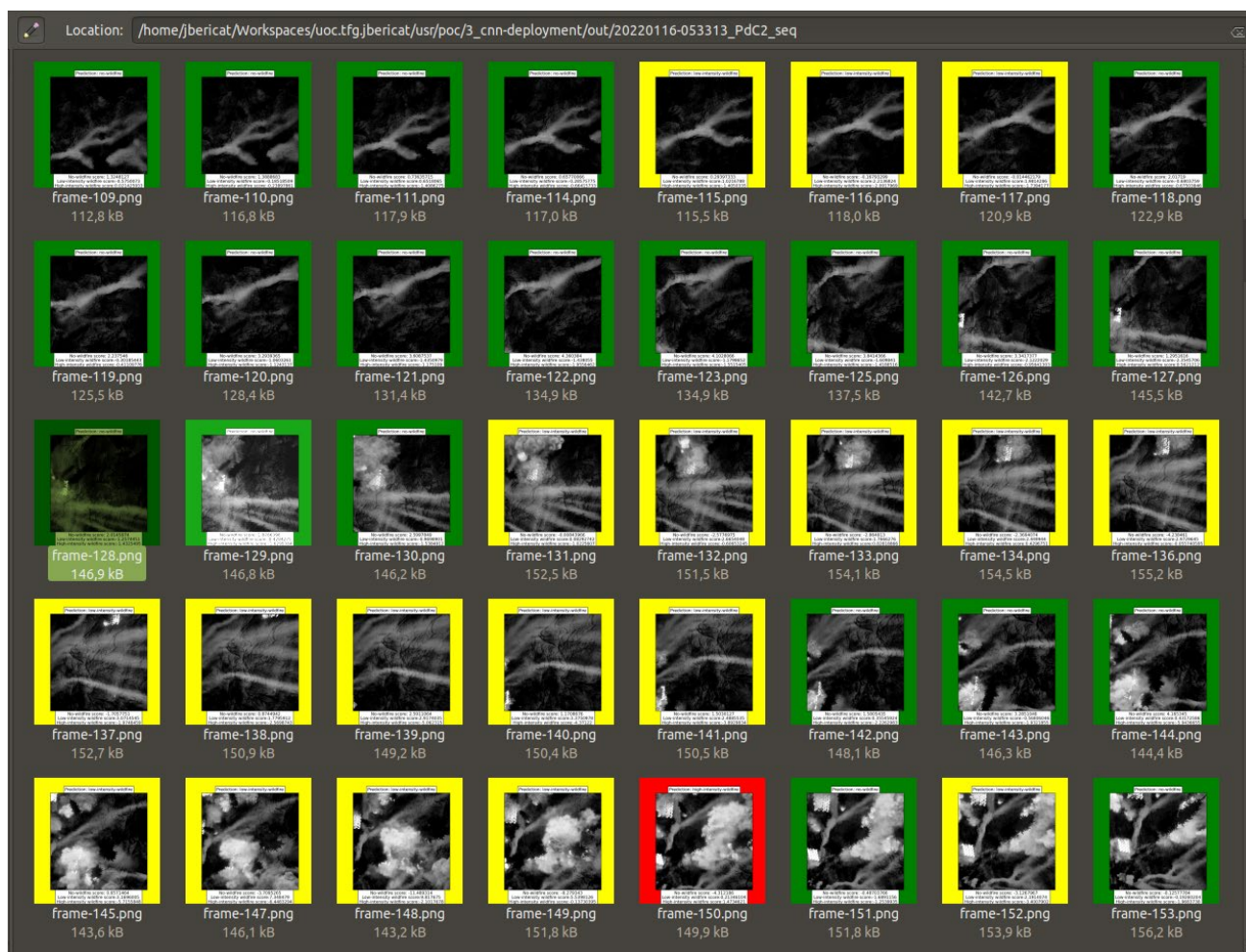


Figura 8.xx.c -

Observem que en aquest cas, tot i encertar la majoria de casos en els que no hi ha cap instància d'incendi, en la resta de casos en els que es troba un incendi d'alta intensitat tenim que només es produeix un encert de cada 26 mostres. Aleshores, com que els resultats observats a la figura 8.xx.c contradiuen els obtinguts a les figures 8.xx.a i 8.xx.b, tenim que s'haurà d'estudiar perquè ara les imatges de classe **no-wildfires** sí que són classificades correctament (1), però aquelles que aquelles que contenen incendis de classe **high-intensity-wildfires** deixen de ser-ho (2). Es motius podrien ser els següents:

- Les imatges de classe **no-wildfire** analitzades al lot de la figura 8.x.c són més fosques que les del lot 8.x.b (a causa del problema de la neu indicat a la secció 8.1.1)
- Les imatges de classe **high-intensity-wildfire** són massa llunyanes (el model s'ha entrenat a

Finalment, si prenem una tercera i quarta mostra que inclogui imatges de la classe `low-intensity-wildfires`, (**figures 8.xx.a i 8.xx.b**), observarem com en la primera la quantitat d'encerts en les prediccions torna a ser relativament alt, entre el 75-80%, mentre que si ens fixem en la segona mostra veurem que aquest cop el percentatge d'encert baixa fins tansols el 25-30%

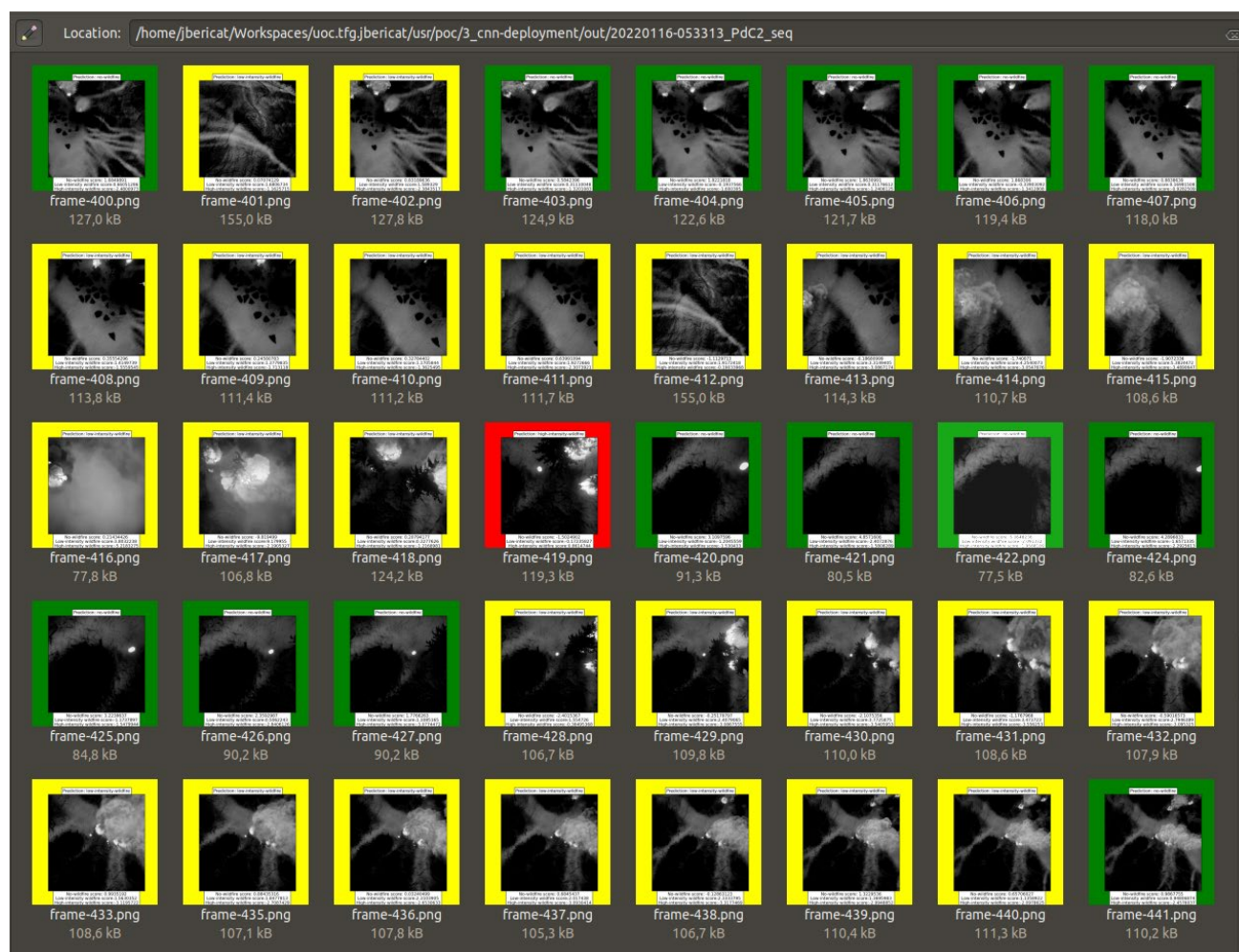
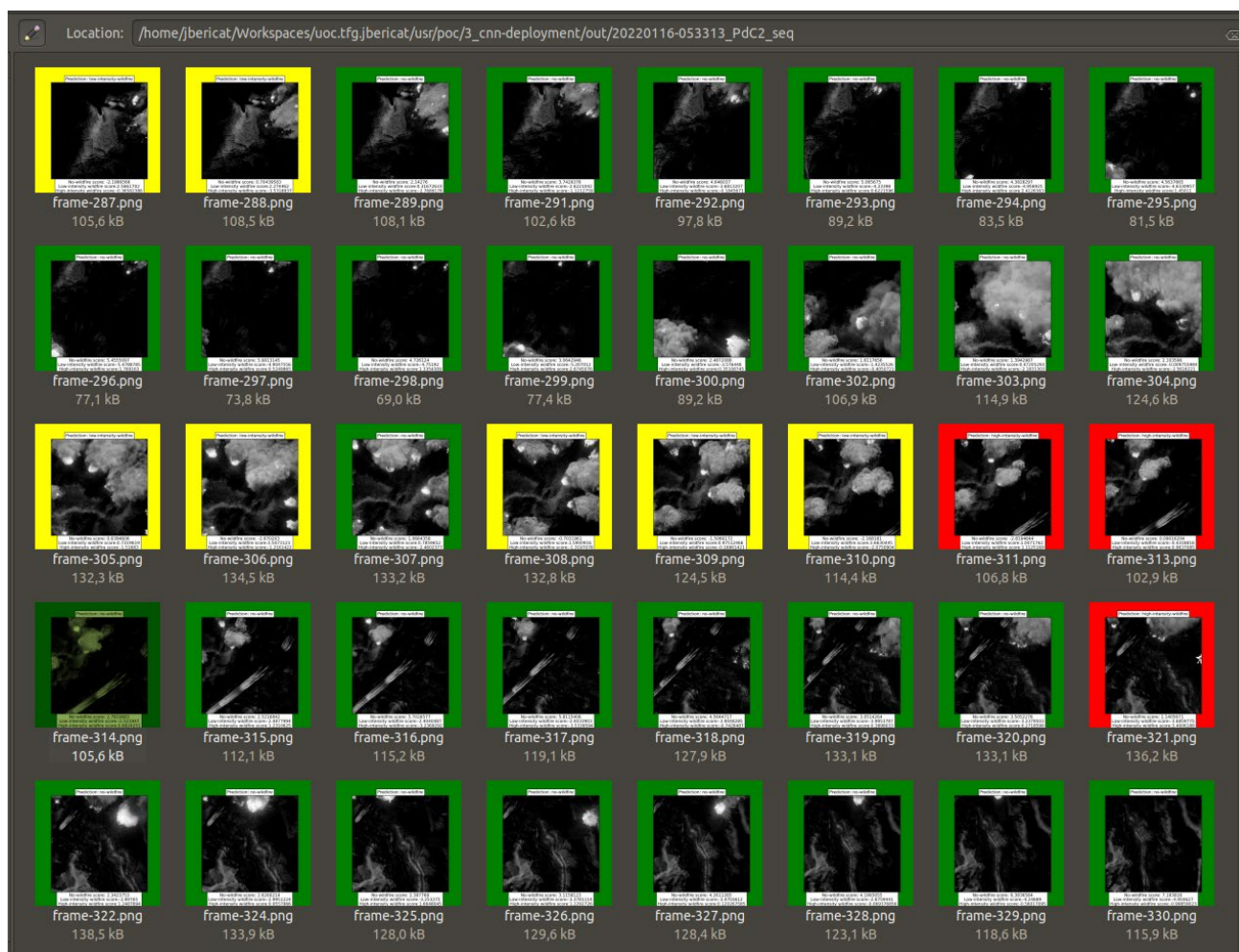


Figura 8.xx.a -



De les deliberacions anteriors podem concloure que encara pot estar-se produint un factor d'aleatorietat que alteri els resultats de les inferències, tot i que en menor grau que durant la primera execució de la PdC realitzada a la [secció 8.1](#). Per tant, estudiar quines mesures prendre per aconseguir que el model es comporti de manera esperada, o que al menys deixi de mostrar comportaments tant aleatoris.

Es pot consultar el conjunt total de mostres classificades a la següent ruta del projecte

RUTA!!!!*

8.2.3 – recol·lecció de feedback i implementació de millores

Donada la existència d'un factor d'aleatorietat (o si més no un factor desconegut) que dificulta centrar-se en l'estudi d'un problema concret, aleshores en comptes de fer un estudi estadístic de la informació continguda en les metadades de cada resultat (per tal de trobar patrons que es repeteixin en situacions concretes, o bé casos particulars amb puntuacions molt altes, etc), el que farem serà procedir de manera selectiva i revisar per eliminació aquells punts del *pipeline* d'execució de la PdC que no s'hagin revisat en la primera prova.

8.2.3.1 – Dataset d'imatges

En aquesta segona execució de la PdC no farem cap canvi al dataset d'imatges, ja que encara podem emprendre altres accions abans de tornar a revisar-ho de nou.

8.2.3.2 – Algorisme d'entrenament

Pel que respecta l'algorisme d'entrenament, després d'una revisió exhaustiva del codi s'ha trobat que la funció `test_batch()` del fitxer `pytorch-training.py` no estava fent la crida la directiva `model.eval()`. Per a ser més precisos, aquesta directiva s'encarrega de desactivar el mode d'entrenament del model, de manera que si no s'utilitza en les funcions que tinguin com a finalitat la de mesurar el rendiment del model (ja sigui durant l'entrenament o durant la inferència), aleshores el que passarà és que estarem re-entrenant el model a mida que realitza la seva tasca de classificació. Per a comprovar si aquesta darrera hipòtesi és certa, o si més no aproximada, el que farem serà afegir la directiva `model.eval()` a la funció corresponent d'inferència per lots esmentada, tal i com s'indica a la **figura 8.xx**:

```
def testBatch():  
  
    # This directive MUST be enabled for testing  
    # (I'd say it wasn't present on the original code)  
    model.eval()
```

Figura 8.xx – activació del mode avaluació del model de xarxes neuronals durant els assajos

El codi revisat de `__` es troba a la següent ruta de la branca FITA#08 (v08.8) del repositori

8.2.3.3 – Algorisme d'inferència

Com en el cas de les dades, l'algorisme d'explotació `pytorch_deployment.py` ja s'ha revisat a la [secció 8.2.1.2](#) i de moment no farem cap mes canvi, a part de baixar la quantitat d'epochs de 40 a 20

8.3 – Tercera execució de la prova de concepte

8.3.1 – Preparació de la prova

8.3.1.1 - Entrenament del model amb l'algorisme revisat a la secció 8.2.2.2

Per a tornar a entrenar el model només cal que executem de nou l'script `pytorch_training.sh`, tal i com s'indica a la següent figura:

```
(base) jbericat@TF6-UOC:~/Workspaces/uoc.tfg.jbericat/src/poc/2_cnn-training$ ./pytorch_training.sh

*****
2. CNN Training Algorhythm
*****

Set the model version you want to train:
1 = v1.0 -> 3 layers CNN (simplified LeNet)
2 = v2.0 -> 5 layers CNN (LeNet)
3 = v3.0 -> 14 layers CNN (custom)

Please choose an option (1-3 - Default = 3): 3
Set the dataset version you want to use to train the model:
4 = v4.0 -> ARCHIVED (EXPERIMENTAL)
5 = v5.0 -> ARCHIVED (EXPERIMENTAL)
6 = v6.0 -> ARCHIVED (EXPERIMENTAL)
7 = v7.0 -> 600 samples: 3 classes, close distance images (SMALLEST DATASET, appropriate for adjusting parameters)
8 = v8.0 -> 1.5K samples: 4 classes, close distance images
9 = v9.0 -> 1.5K samples: 3 classes, close distance images
10 = v10.0 -> ARCHIVED (EXPERIMENTAL)
11 = v11.0 -> ARCHIVED (EXPERIMENTAL)
12 = v12.0 -> 3.7K samples: 3 classes, close, mid & long distance images (Improved no-wildfire class images).

Please choose an option (4-12 - Default = 7): 12

[INFO] generating the train/validation split...
[INFO] Training the convolution neural network model, hold-on tight...

[INFO] EPOCH: 1/20
Train loss: 0.486503, Train accuracy: 84.3522
Val loss: 0.370824, Val accuracy: 84.0751

[INFO] EPOCH: 2/20
Train loss: 0.163241, Train accuracy: 96.9047
Val loss: 0.143263, Val accuracy: 95.8323

[INFO] EPOCH: 3/20
Train loss: 0.096850, Train accuracy: 97.8993
Val loss: 0.087322, Val accuracy: 97.7360
```

Figura 08.x.a – Entrenament del model de xarxes neuronals corresponent a la tercera execució de la prova de concepte


```
[INFO] EPOCH: 14/20
Train loss: 0.044353, Train accuracy: 99.2369
Val loss: 0.032972, Val accuracy: 98.8937

[INFO] EPOCH: 15/20
Train loss: 0.027487, Train accuracy: 99.6742
Val loss: 0.030333, Val accuracy: 99.2539

[INFO] EPOCH: 16/20
Train loss: 0.025455, Train accuracy: 98.5424
Val loss: 0.067219, Val accuracy: 98.0190

[INFO] EPOCH: 17/20
Train loss: 0.013326, Train accuracy: 98.7310
Val loss: 0.075983, Val accuracy: 98.7651

[INFO] EPOCH: 18/20
Train loss: 0.024936, Train accuracy: 99.6142
Val loss: 0.026076, Val accuracy: 99.4083

[INFO] EPOCH: 19/20
Train loss: 0.027208, Train accuracy: 99.6313
Val loss: 0.021102, Val accuracy: 99.4340

[INFO] EPOCH: 20/20
Train loss: 0.015770, Train accuracy: 99.8628
Val loss: 0.014469, Val accuracy: 99.5626

[INFO] Finished Training. Generating summary tarball...

[INFO] Training model and summary file saved at: /home/jbericat/Workspaces/uoc.tfg.jbericat/usr/poc/2_cnn-training/cnn-training_20220117-130856.tar.gz
```

Figura 098.x.b – Entrenament del model de xarxes neuronals corresponent a la tercera execució de la prova de concepte

A la carpeta de sortida indicada a la figura anterior trobem el resum d'entrenament següent:

```
*****
***                                     ***
***                               PoC's CNN Model Training Summary             ***
***                               ***
*****
***                               ***
***                               Model version: v3.0                        ***
***                               Dataset version: v12.0                     ***
***                               ***
*****

TRAINING PARAMETERS:

- Image size = (229 x 229 x 1)
- Number of classes / labels = 3
- Batch size = 128
- Learning rate = 0.0001

*****

DATASET TOTALS:

- The number of images in a training set is: 11776
- The number of images in a validation set is: 3968
- The number of images in a test set is: 640
- The number of batches per epoch is: 92

*****
```

Figura 8.xx.a - Resum d'entrenament del model CNN utilitzant l'algorisme `pytorch_training.py` revisat


```

*****

CNN BLUEPRINT:

Network_v3(
  (conv1): Conv2d(1, 16, kernel_size=(9, 9), stride=(2, 2), padding=(2, 2))
  (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv2): Conv2d(16, 16, kernel_size=(5, 5), stride=(2, 2), padding=(2, 2))
  (bn2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (pool1): MaxPool2d(kernel_size=2, stride=1, padding=0, dilation=1, ceil_mode=False)
  (conv3): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(2, 2))
  (bn3): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv4): Conv2d(32, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (bn4): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (fc1): Linear(in_features=107648, out_features=3, bias=True)
)

*****

MODEL TRAINING STATS:

- Model trained on cuda:0 device
- Training time: 10677.349 seconds

*****

MODEL TESTING STATS:

- For epoch 1 the TEST accuracy over the whole TEST dataset is 79 %
- For epoch 2 the TEST accuracy over the whole TEST dataset is 84 %
- For epoch 3 the TEST accuracy over the whole TEST dataset is 74 %
- For epoch 4 the TEST accuracy over the whole TEST dataset is 88 %
- For epoch 5 the TEST accuracy over the whole TEST dataset is 80 %
- For epoch 6 the TEST accuracy over the whole TEST dataset is 84 %
- For epoch 7 the TEST accuracy over the whole TEST dataset is 80 %
- For epoch 8 the TEST accuracy over the whole TEST dataset is 83 %
- For epoch 9 the TEST accuracy over the whole TEST dataset is 78 %
- For epoch 10 the TEST accuracy over the whole TEST dataset is 75 %
- For epoch 11 the TEST accuracy over the whole TEST dataset is 74 %
- For epoch 12 the TEST accuracy over the whole TEST dataset is 81 %
- For epoch 13 the TEST accuracy over the whole TEST dataset is 78 %
- For epoch 14 the TEST accuracy over the whole TEST dataset is 77 %
- For epoch 15 the TEST accuracy over the whole TEST dataset is 84 %
- For epoch 16 the TEST accuracy over the whole TEST dataset is 80 %
- For epoch 17 the TEST accuracy over the whole TEST dataset is 91 %
- For epoch 18 the TEST accuracy over the whole TEST dataset is 83 %
- For epoch 19 the TEST accuracy over the whole TEST dataset is 80 %
- For epoch 20 the TEST accuracy over the whole TEST dataset is 88 %

*****

```

Figura 8.xx.b - Resum d'entrenament del model CNN utilitzant l'algorisme `pytorch_training.py` revisat

FINAL PREDICTION TEST:

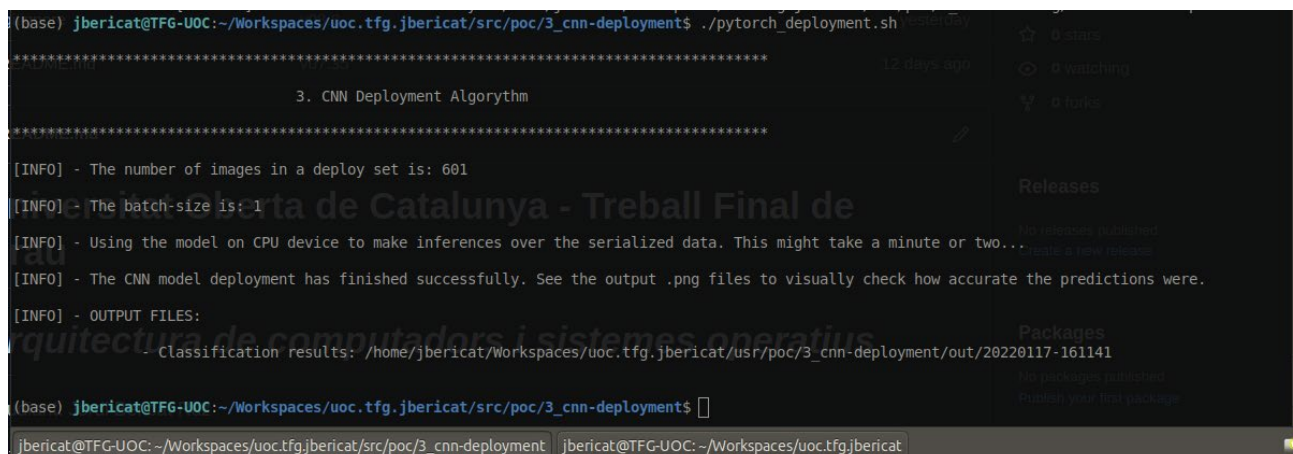
- Model tested on cuda:0 device
- Final test accuracy: 91 %

Real Label	Predicted Label	Output score [High / Low / No]
low-intensity-wildfires	high-intensity-wildfires	[1.232942 0.09906744 -5.1467724]
low-intensity-wildfires	low-intensity-wildfires	[-8.845539 2.417996 1.578095]
no-wildfires	no-wildfires	[-3.2360299 -3.9604971 5.659079]
no-wildfires	no-wildfires	[-4.5699334 -3.7302165 5.988307]
low-intensity-wildfires	low-intensity-wildfires	[-4.5925546 7.610162 -10.001231]
low-intensity-wildfires	low-intensity-wildfires	[0.9356603 1.2943679 -6.957526]
high-intensity-wildfires	high-intensity-wildfires	[20.771608 -9.698544 -20.058754]
no-wildfires	no-wildfires	[-4.2560596 -3.8140867 6.3152194]
low-intensity-wildfires	low-intensity-wildfires	[-4.277995 2.8907914 -2.0460386]
low-intensity-wildfires	no-wildfires	[-5.1211605 0.12469656 1.8801923]
low-intensity-wildfires	low-intensity-wildfires	[-7.893419 4.4963374 -3.2514513]
no-wildfires	no-wildfires	[-3.1900156 -5.15512 7.537456]
high-intensity-wildfires	high-intensity-wildfires	[7.5525484 -5.4916716 -6.1236277]
no-wildfires	no-wildfires	[-3.46193 -2.5222769 3.442431]
high-intensity-wildfires	high-intensity-wildfires	[32.59405 -13.47883 -36.692665]
low-intensity-wildfires	no-wildfires	[-3.8990552 -0.1359438 -0.10370919]
low-intensity-wildfires	low-intensity-wildfires	[-5.439094 5.61463 -4.5535164]
low-intensity-wildfires	low-intensity-wildfires	[-1.8680964 3.887939 -8.090165]
no-wildfires	no-wildfires	[-4.543063 -4.217307 6.106032]
low-intensity-wildfires	low-intensity-wildfires	[-4.3644395 4.2229605 -5.569517]
no-wildfires	no-wildfires	[-5.5659146 -4.7293787 7.5036035]
no-wildfires	no-wildfires	[-0.27666548 -4.689765 4.2440248]
high-intensity-wildfires	high-intensity-wildfires	[24.646006 -5.791209 -29.864662]
no-wildfires	no-wildfires	[-5.4311285 -1.7750133 4.52081]

Figura 8.xx.c - Resum d'entrenament del model CNN utilitzant l'algorisme `pytorch_training.py` revisat

8.3.2 – Inferència de les dades

En aquest cas, com que no hem realitzat cap canvi en l'algorisme d'inferència, procedim directament a executar l'script corresponent d'explotació del model:



```

(base) jbericat@TFG-UOC:~/Workspaces/uoc.tfg.jbericat/src/poc/3_cnn-deployment$ ./pytorch_deployment.sh
*****
3. CNN Deployment Algorhythm
*****

[INFO] - The number of images in a deploy set is: 601
[INFO] - The batch-size is: 1
[INFO] - Using the model on CPU device to make inferences over the serialized data. This might take a minute or two...
[INFO] - The CNN model deployment has finished successfully. See the output .png files to visually check how accurate the predictions were.
[INFO] - OUTPUT FILES:
- Classification results: /home/jbericat/Workspaces/uoc.tfg.jbericat/usr/poc/3_cnn-deployment/out/20220117-161141

(base) jbericat@TFG-UOC:~/Workspaces/uoc.tfg.jbericat/src/poc/3_cnn-deployment$
  
```

A la captura de la figura anterior es pot consultar la ruta del repositori on trobar els resultats de classificació.

8.3.3 – Conclusions obtingudes després de la tercera execució de la prova

En primer lloc tal i com podem veure a la **figura 8.xx**, tenim que després d'entrenar de nou el model de xarxes neuronals en les mateixes condicions que durant la segona execució, però corregint la errada esmentada a la [secció 8.2.3.2](#), la precisió que ofereix aquest avaluat contra el subconjunt de test (assaigs) segueix estant per sobre de l'esperat, en aquest cas als volts del **91%**.

D'altra banda, i més important, aquest cop trobem que el model ja no es comporta de manera erràtica, tot i que la precisió que ofereix aquest contra el subconjunt de dades d'explotació està per sota del que ofereix amb el d'assaigs (**entre el 70-80%**)

Si estudiem cas per cas les millores, tenim:

a) Classe no-wildfires

Com es pot observar a la figura següent, tenim que ara el model ja no confon la neu amb el fum, de manera que classifica correctament la majoria d'imatges sense foc (continguin o no elements amb neu o gel):



Figura 8.xx – Resultats de la classificació després de la tercera execució de la PdC. Fins a la frame-57.png totes són imatges sense incendis, i per tant, prediccions correctes. A partir de la 58 són imatges amb incendis d'alta intensitat, que comentarem al punt següent

b) Classe low-intensity wildfires

Pel que fa les imatges d'aquesta classe, tenim que el model tendeix a confondre-les amb les de la classe no-wildfire, sobretot en els casos en què les imatges per a la inferència es prenen des d'una distància massa llunyana, tal i com es pot comprovar a la figura següent:



Figura 8.xx – Imatges que contenen incendis de baixa intensitat, capturades des de diferents distàncies

A més, això darrer ho podem verificar mitjançant la puntuació associada a la inferència. Per posar un exemple, a la figura següent tenim que les puntuacions de les classes **no-wildfire** i **low-intensity-wildfire** són molt properes, fet que indica que el model no estava del tot segur de si estava classificant correctament la imatge (tot i que la probabilitat més alta segons els seus càlculs fou la de que fos de classe **no-wildfire**):

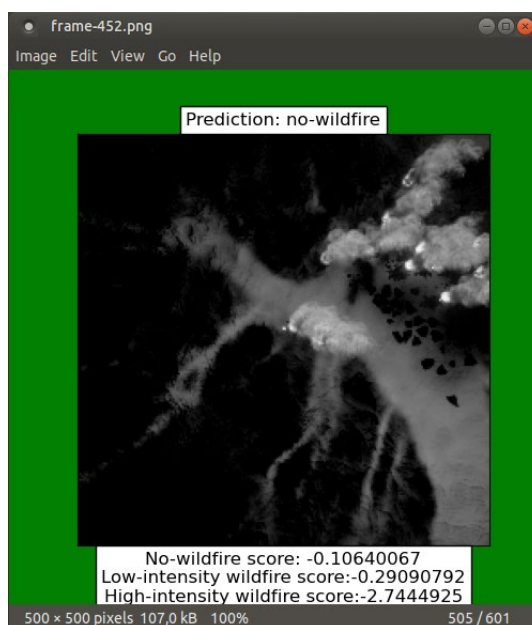


Figura 8.xx – Classificació incorrecta com a imatge sense incendis d'una imatge amb incendis de baixa intensitat. fitxer `/usr/poc/3_cnn-deployment/out/20220117-161141_PdC3/frame-452.png`

c) Classe **high-intensity wildfires**

Pel que fa aquesta classe, tot i que el rendiment obtingut no és tant alt que l'obtingut durant l'entrenament i testeig (91%), trobem que la quantitat d'encerts es bastant elevada, vora del 80%. En aquest cas observem que aquesta xifra és significativament superior a la obtinguda per a la de classe **low-intensity** (veure **figura 8.xx**), tot i que no tant bona com en el cas de la classe **no-wildfire** vista al punt anterior. A la **figura 8.xx** es pot observar una mostra representativa de la precisió aconseguida:



Figura 8.xx – Imatges que contenen incendis de classe **high-intensity-wildfire**.

Tanmateix, s'observa que en els casos en què les imatges es situen massa a la bora de la imatge, el classificador tendeix a confondre les imatges amb les d'una altra classe (majoritàriament, amb les de classe **no-wildfire**, però no sempre). Això pot ser degut a un excés en l'ús de les tècniques d'augmentació. Per a arreglar-ho, podríem tornar a entrenar el model substituint aquesta tècnica per una altra que no suposi **pèrdua d'informació**⁹ durant la transformació (per exemple, *mirroring* horitzontal o vertical).

⁹ Per exemple, fent la rotació d'una imatge que només presenta característiques de foc intens molt a la bora de la captura, després d'una rotació de 45° aquesta característica pot desaparèixer de la captura i convertir-se en una imatge de classe **no-wildfire** sense estar etiquetada com a tal.

8.4 – Conclusions finals

- **Calen moltes més imatges** del que es pensava per a assolir nivells òptims de precisió durant els assajos, tot i que ha quedat demostrat que la generació de dades sintètiques es pot efectuar de manera massiva amb un esforç computacional relativament baix i hardware “off-the-shelves”.
- S’ha d’anar en compte amb les **tècniques d’ampliació**, el balanceig i la riquesa / representació de cada classe al generar els dataset sintètics¹⁰
- **L’estudi de recerca** previ en el tipus d’imatges que es volen recrear de manera sintètica, així com de l’àmbit o entorn és una part extremadament important en la qual no s’ha d’estalviar temps i recursos si després utilitzar aquest coneixement per a entrenar un model de IA productiu. Cal considerar també que existeixen tècniques de **transfer-learning**, de manera que la inversió feta en una àrea de recerca pot ser recuperada en el futur. De totes maneres, cal remarcar que aconseguir imatges sintètiques útils tant per a la recerca acadèmica com per al seu ús en entorns de producció no és una tasca trivial i que requereix l’anàlisi d’una gran quantitat de variables.
- La utilització d’una **arquitectura convolucional de 14 capes** (poc complexa) per implementació de la xarxa convolucional ha sigut suficient per a desenvolupar un classificador d’imatges en blanc i negre (1 canal) i on les característiques dels incendis s’han simulat de manera molt semblant en totes les instàncies cada classe (mitjançant l’entorn Unreal Engine). Tanmateix, en un entorn real amb càmeres FLIR que permetin detectar diferents graus de temperatura, potser caldria una arquitectura més complexa que oferís un bon rendiment per a imatges en color (3 canals), i amb capacitat de detectar moltes més característiques.
- Dels resultats de classificació assolits després de la tercera iteració es pot concloure que **la execució de la prova de concepte ha sigut un èxit**, sobretot donada la naturalesa del projecte i el manteniment dels compromisos establerts durant la fase de disseny inicial, els quals s’han prioritzat sempre (per exemple, la implementació de la visió tèrmica nocturna ha sigut un dels hàndicaps més significants). Tanmateix, cal deixar palès que aquests resultats no són els òptims per a un entorn de producció, tot i que les conclusions obtingudes poden esdevenir el punt de partida d’altres treballs en l’àrea de la visió per computador i la utilització de dades sintètiques per al seu desenvolupament.
- El fet d’haver escollit **python + pytorch** per a realitzar aquest treball ha facilitat molt la feina de desenvolupament donat l’alt nivell d’abstracció de les llibreries que ofereixen. Tanmateix, de vegades ha sigut difícil compatibilitzar l’entorn python necessari per a connectar amb l’entorn de la PdC (API de AirSim) amb l’entorn utilitzat per a desenvolupar la xarxa convolucional profunda (pytorch).

¹⁰ Link Omniverse, etc

8.5 – Consideracions addicionals al respecte de l'abast, objectius i compromisos establerts durant la planificació inicial d'aquest projecte