

PAC. Pràctica 2.

J. de Curtò i DíAz & I. de Zarzà i Cubero.

c@decurto.be z@dezarza.be

Tipologia i cicle de vida de les dades. Màster de Ciència de Dades.

Alumnes/as:

J. de Curtò i DíAz. decurto@uoc.edu

I. de Zarzà i Cubero. dezarza@uoc.edu

Pràctica 2. Neteja i anàlisi de dades.

Extensió del dataset CyZ, generat a la Pràctica 1.

CyZ: MARS Space Exploration Dataset.

<https://github.com/decurtoidiaz/cyz>

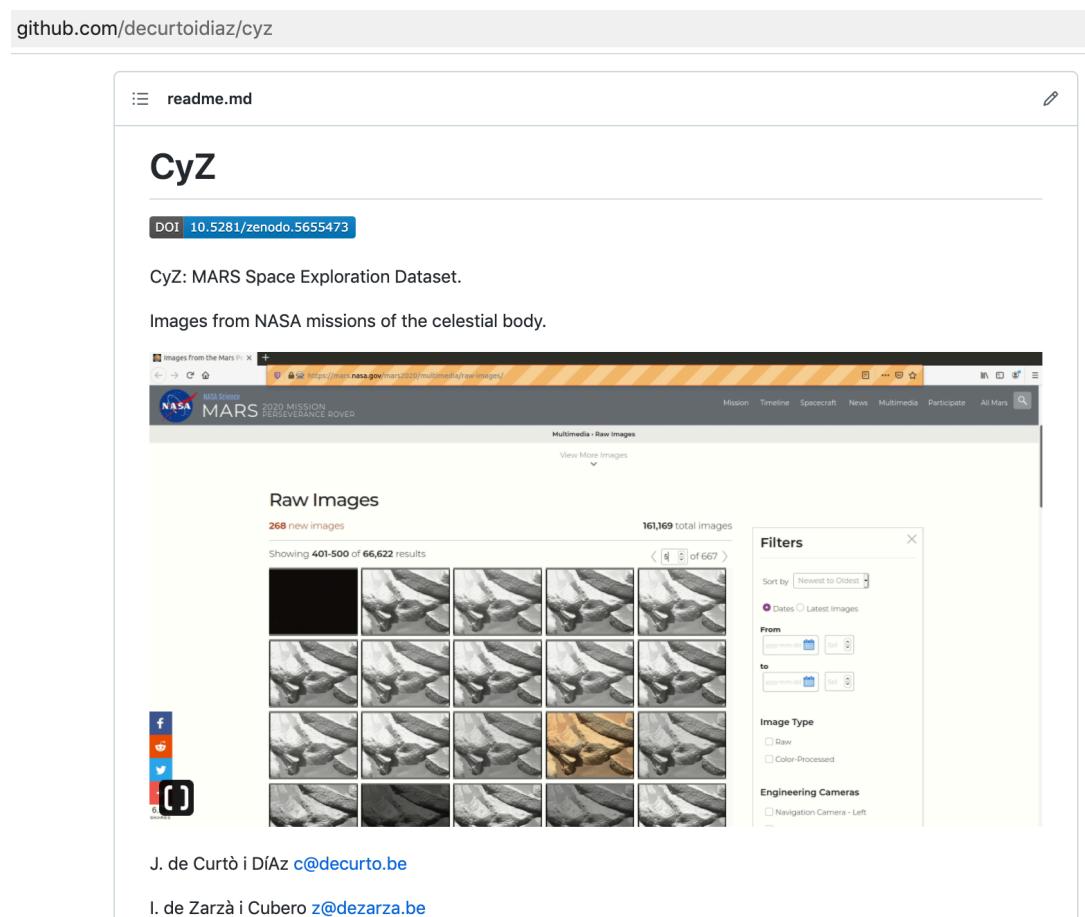


Figura 1. Repòsitori CyZ. Conté el dataset i els scripts introduïts a la pràctica 1; presenta al voltant de 60000 imatges de les missions Curiosity i Perseverance de la NASA.

DrCyZ: Techniques for analyzing and extracting useful information from CyZ.

<https://github.com/decurtoidiaz/drcyz>

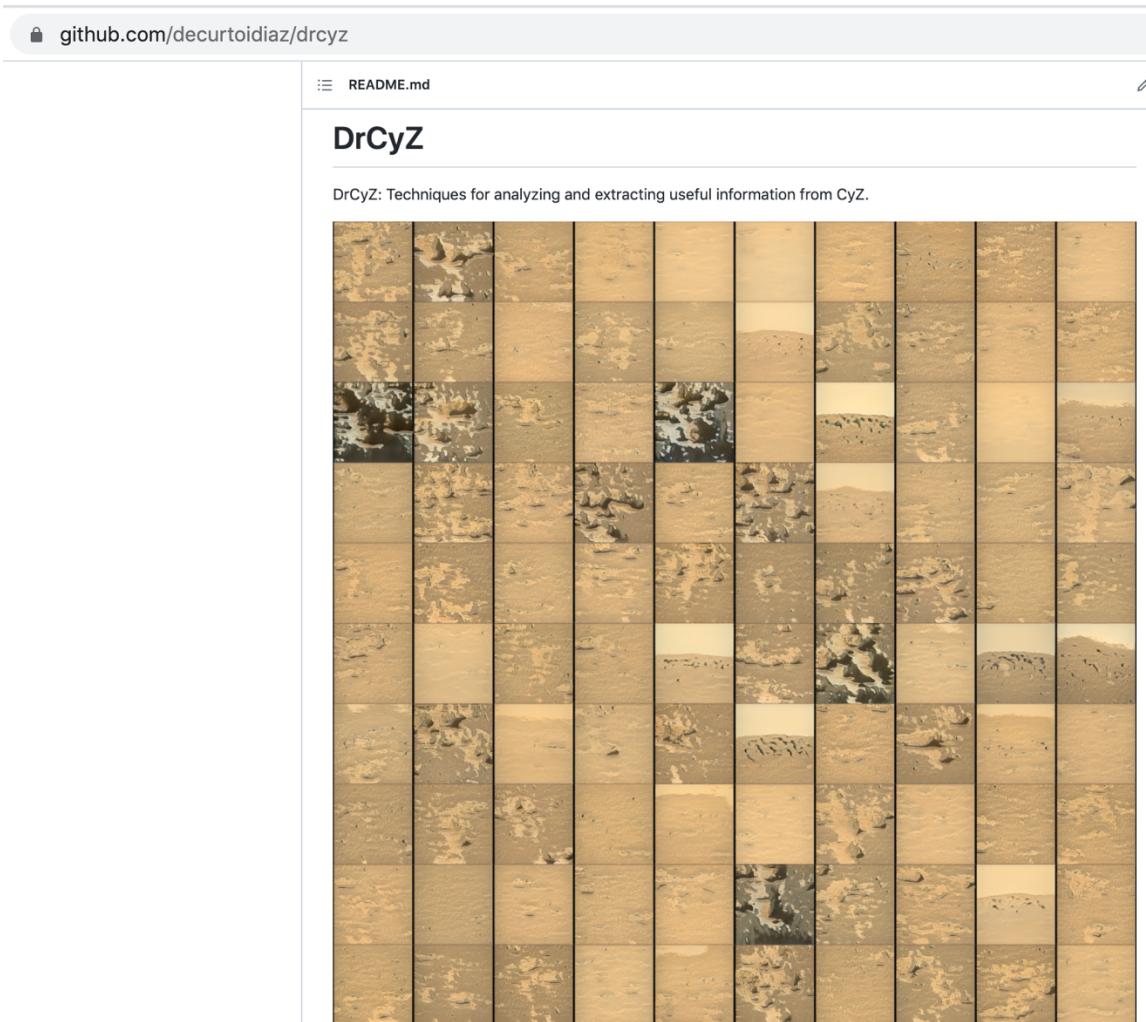


Figura 2. Repòsitori DrCyZ. Conté eines per analitzar i visualitzar el dataset CyZ. Permet visualitzar les dades mitjançant K-means Clustering i t-SNE (amb PCA), generar mostres sintètiques utilitzant una estructura Stylegan2-ada, fer càlculs estadístics per comparar la distribució original i la sintètica i fer segmentació semàntica de les mostres.

1. Descripció del dataset. Perquè és important i quina pregunta/problema pretén respondre?

El dataset [1] conté imatges de les missions Perseverance i Curiosity de la NASA [2,3,4,5,6] i està inspirat en datasets per dur a terme tasques de visió per computador en aplicacions espacials [7,8]. Va ser introduït prèviament a la pràctica 1 de web scraping d'aquesta assignatura, Figura 1. Aquest dataset presenta imatges que poden ésser molt útils per fer recerca en visió per computador i el nostre objectiu és netejar les dades, fer una anàlisi en profunditat d'aquests i estendre la informació proporcionada a la pràctica anterior, Figura 2.

2. Integració i selecció de les dades d'interès a analitzar.

El primer pas a dur a terme és una anàlisi mitjançant K-means Clustering per visualitzar les dades de manera que sigui possible dur a terme la seva neteja. També ens permet identificar els grups d'imatges que s'assemblen més entre ells i per tant, les càmeres que ens poden ser més útils a seleccionar donada una tasca de recerca posterior. Un cop seleccionades les dades que utilitzarem per fer les posteriors anàlisis, es pot optar primer per estudiar les dades de manera estadística i després per utilitzar mètodes més sofisticats d'aprenentatge

PAC. Pràctica 2.

J. de Curtò i DíAz & I. de Zarzà i Cubero.

c@decurto.be z@dezarza.be

automàtic i profund per una tasca concreta (classificació, segmentació, generació d'imatges sintètiques...).

```
▶ from sklearn.decomposition import PCA

features = np.array(features)
pca = PCA(2)

#Transform the data
df = pca.fit_transform(features)

n_cams = 8
kmeans = MiniBatchKMeans(n_clusters=n_cams)

#predict the labels of clusters.
label = kmeans.fit_predict(df)

#Getting unique labels
u_labels = np.unique(label)
```

Figura 3. Codi corresponent a l'algoritme K-means Clustering previa descomposició PCA en dues dimensions. *N_cams* determina el nombre de clusters (8 i 22, respectivament).

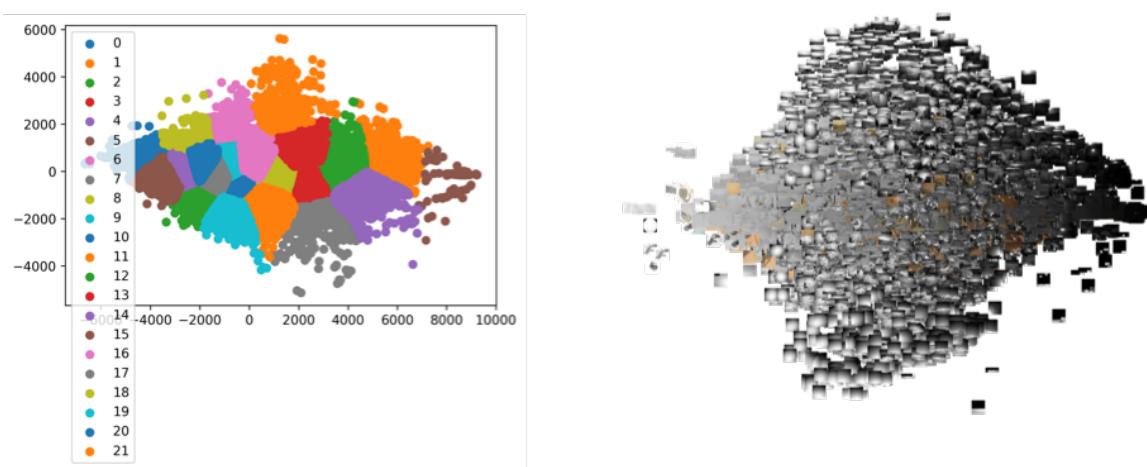
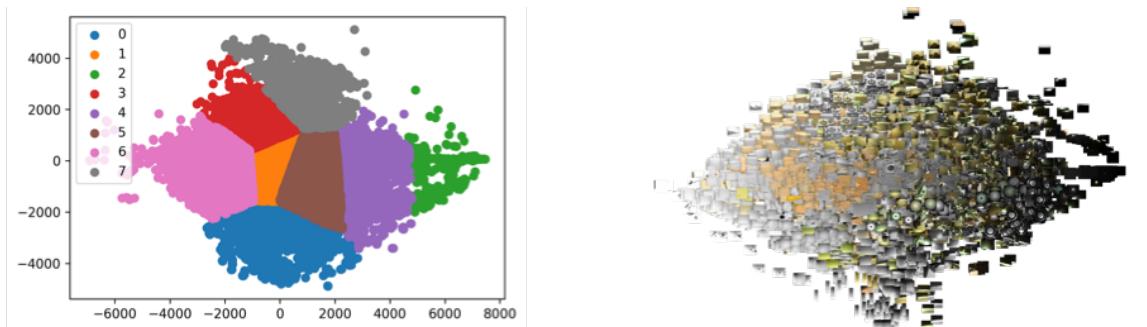


Figura 4. K-means Clustering. Imatges de la missió Curiosity (fila superior). Imatges de la missió Perseverance (fila inferior). Hyperparametre K seleccionat segons el nombre de càmeres usades per obtenir les mostres. Fem una projecció a 2 dimensions utilitzant el PCA prèviament al K-means Clustering.

PAC. Pràctica 2.

J. de Curtò i DíAz & I. de Zarzà i Cubero.

c@decurto.be z@dezarza.be

Com podem observar la visualització utilitzant K-means Clustering, Figures 3 i 4, ofereix una primera idea intuïtiva de la distribució en clústers de les imatges del dataset. Com es pot veure al codi, Figura 3, per poder aplicar aquest algoritme hem de fer una projecció en dues dimensions mitjançant la tècnica PCA. Donada la complexitat de la tasca, i el fet que aquesta primera eina de visualització es basa inherentment en mètodes lineals, proposem l'ús d'una tècnica més sofisticada (t-SNE) que ens ajudarà a fer una selecció de les dades, identificar valors extrems i imatges borroses o en un color fix (elements buits o nuls) que no aporten a la tasca a solucionar.

3. Neteja de les dades.

1. 3.1. Les dades contenen zeros o elements buits? Com gestionaries aquests casos?
2. 3.2. Identificació i tractament de valors extrems.

Apliquem K-means Clustering i visualitzem la informació per detectar valors a excloure, Figures 3 i 4. També fem un estudi per buscar imatges que no contenen informació o que estan borroses i no aporten dades significatives. A més a més, també plantegem l'aplicació de la tècnica Principal Components Analysis (PCA) [9] prèviament per entendre i interpretar la informació que aporta cadascuna de les càmeres en dues dimensions.

A continuació apliquem la tècnica t-SNE per dur a terme l'anàlisi visual de les dades i seleccionar un subconjunt per aplicar els algoritmes d'aprenentatge no supervisats i de segmentació semàntica, Figures 5-9.

```
▶ from sklearn.decomposition import PCA  
  
features = np.array(features)  
pca = PCA(n_components=0.99, svd_solver='full')  
pca.fit(features)  
pca_features = pca.transform(features)  
  
print(pca.explained_variance_)  
print(pca.explained_variance_ratio_)  
print(pca.explained_variance_ratio_.cumsum())  
print(pca.n_components_)  
  
↳ [3.32881889e+06 1.68059055e+06 1.08131421e+06 ... 1.01072405e+02  
1.01004857e+02 1.00862950e+02]  
[3.33581389e-01 1.68412206e-01 1.08358643e-01 ... 1.01284794e-05  
1.01217103e-05 1.01074898e-05]  
[0.33358139 0.5019936 0.61035224 ... 0.989998276 0.989999289 0.99000299]  
1919
```

Figura 5. Curiosity. Apliquem el mètode PCA amb una selecció del nombre de components (1919) que expliqui el 99% de la variància i computi les corresponents mesures estadístiques.

```
[ ] num_images_to_plot = 41425  
  
if len(images) > num_images_to_plot:  
    sort_order = sorted(random.sample(range(len(images)), num_images_to_plot))  
    images = [images[i] for i in sort_order]  
    pca_features = [pca_features[i] for i in sort_order]  
  
▶ X = np.array(pca_features)  
tsne = TSNE(n_components=2, learning_rate=150, perplexity=30, angle=0.2, verbose=2).fit_transform(X)
```

Figura 6. Curiosity. Computem l'algoritme t-SNE utilitzant 41425 imatges.

```
▶ from sklearn.decomposition import PCA

features = np.array(features)
pca = PCA(n_components=0.99, svd_solver='full')
pca.fit(features)
pca_features = pca.transform(features)

print(pca.explained_variance_)
print(pca.explained_variance_ratio_)
print(pca.explained_variance_ratio_.cumsum())
print(pca.n_components_)

[ ] [1.16016904e+06 9.08974590e+05 4.84503908e+05 ... 1.23337673e+02
     1.23188237e+02 1.23049858e+02]
[5.87204089e-01 7.45448316e-02 3.97340725e-02 ... 1.01148989e-05
     1.01026436e-05 1.00912952e-05]
[0.58720409 0.66174892 0.70148299 ... 0.98998802 0.98999812 0.99000821]
2031
```

Figura 7. Perseverance. Apliquem el mètode PCA amb una selecció del nombre de components (2031) que expliqui el 99% de la variància i computi les corresponents mesures estadístiques.

```
▶ num_images_to_plot = 22717

if len(images) > num_images_to_plot:
    sort_order = sorted(random.sample(range(len(images)), num_images_to_plot))
    images = [images[i] for i in sort_order]
    pca_features = [pca_features[i] for i in sort_order]

[ ] X = np.array(pca_features)
tsne = TSNE(n_components=2, learning_rate=150, perplexity=30, angle=0.2, verbose=2).fit_transform(X)
```

Figura 8. Curiosity. Computem l'algoritme t-SNE utilitzant 22717 imatges.

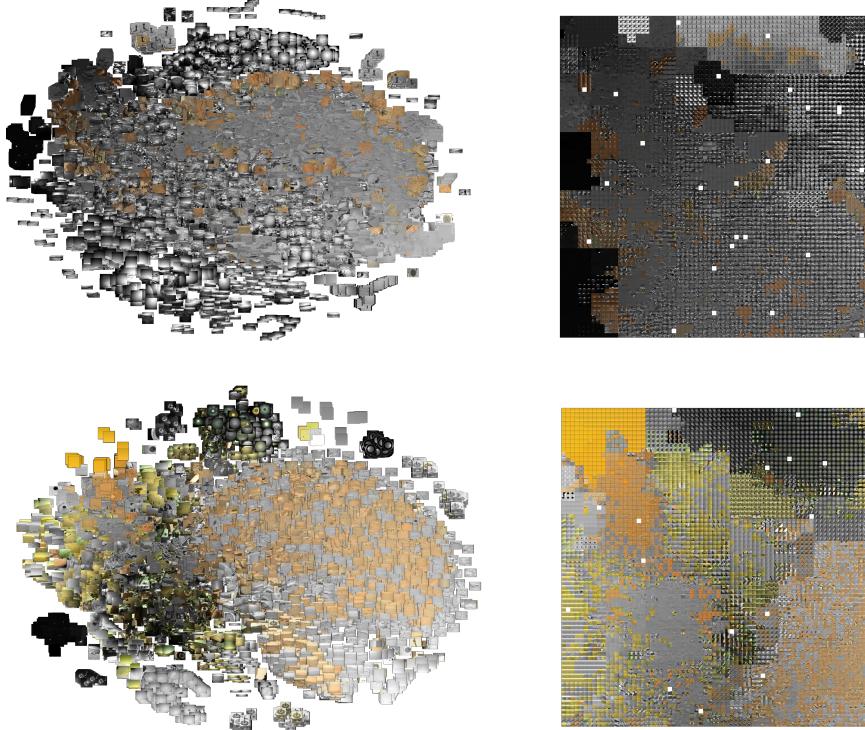


Figura 9. t-SNE. Imatges de la missió Curiosity (fila superior). Imatges de la missió Perseverance (fila inferior). S'aplica la tècnica PCA amb 99% de variància prèviament al mètode de visualització t-SNE. Esquerra representació en núvol. Dreta representació en forma de grid.

Tal com es pot observar a la Figura 9, la tècnica t-SNE, Figures 6 i 8, ens aporta una informació molt més rica on la visualització en clústers és molt més adequada i permet seleccionar les imatges que pertanyen a terreny marcià i identificar núvols de punts d'imatges nul·les o corresponents a elements fixos (e.g. tubs de mostra o parts del rover); del subconjunt total de càmeres. Per arribar a aquesta solució, és crucial seleccionar bé el nombre de components principals del PCA; en aquest cas, Figures 5 i 7, fem diferents experiments i observem com millora la distribució dels clústers i decidim seleccionar el nombre de components principals que expliquin el 99% de la variància. Un altre paràmetre important és la perplexitat del t-SNE; en aquest cas triem el valor 30.

4. Anàlisi de les dades.

- 4.1. Selecció dels grups de dades que es volen analitzar/comparar (planificació de les anàlisis a aplicar).
- 4.2. Comprovació de la normalitat i homogeneïtat de la variància.
- 4.3. Aplicació de proves estadístiques per comparar els grups de dades. En funció de les dades i de l'objectiu de l'estudi, aplicar proves de contrast d'hipòtesis, correlacions, regressions, etc. Aplicar almenys tres mètodes d'anàlisi diferents.

Apliquem la tècnica T-distributed Stochastic Neighbor Embedding (t-SNE) [10,11,12,13] per fer una visualització de les dades; es tracta d'una tècnica no-lineal de reducció de la dimensionalitat. Primer es construeix una distribució de probabilitats sobre parells d'imatges de tal manera que s'assigna probabilitat més alta a les instàncies semblants mentre que s'assigna baixa probabilitat a les mostres que no s'assemblen. Després es projecten aquests punts en un mapa de baixa dimensionalitat i es minimitza la KL divergence [14] entre les dues distribucions. Aquesta tècnica ha estat molt emprada per visualitzar imatges en el context de datasets per visió per computador i moltes altres aplicacions (genòmica, nlp, medicina,...).

De manera paral·lela, també adjuntem anàlisis de variància del conjunt de dades per seleccionar el nombre de components del PCA (Figura 5 i 7). Seleccionem el nombre de components del PCA que expliquin el 99% de la variància (Curiosity: 1919 components i Perseverance: 2031 components) prèviament a aplicar el mètode de visualització t-SNE per millorar la representació. A partir d'aquesta representació seleccionem 5025 imatges que contenen mostres del terreny marcià adequades per la generació sintètica i la segmentació semàntica. Proporcionem el dataset preprocessat en tamany 256, 512 i 1024 en format png i en format TFRecords (per ser importat a Tensorflow) al repositori de l'entrega per la ràpida importació i ús.

Utilitzem una tècnica d'aprenentatge no supervisat, Generative Adversarial Networks (GAN) [15,19] per generar a partir del dataset proporcionat noves imatges sintètiques i veure si la xarxa és capaç d'extreure la informació més significativa. Aquest procés ens permet entendre millor l'estructura de les mostres i la capacitat per generar noves dades donat un subconjunt de dades reals. Per entrenar la xarxa d'aprenentatge profund utilitzem una tècnica de data augmentation per reduir l'overfitting, en aquest cas fem un mirror vers l'eix horitzontal de totes les mostres. Generem un conjunt de 100, 1000 i 10000 mostres a partir del model entrenat en el subconjunt de dades escollit. L'entrenament es duu a terme durant unes 48h al núvol mitjançant una GPU NVIDIA Tesla P-100. El model entrenat i el codi emprat es poden trobar al repositori.

A continuació es pot veure la seqüència del codi i parts de l'execució més representatives.

PAC. Pràctica 2.

J. de Curtò i DíAz & I. de Zarzà i Cubero.

c@decurto.be z@dezarza.be

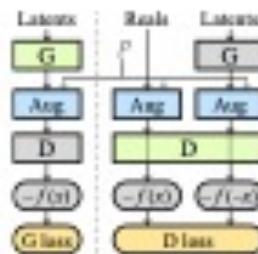
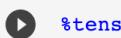


Figura 10. Arquitectura de l'estructura GAN utilitzada a Stylegan2-ada. Es pot observar la presència d'augmentacions probabilístiques adaptatives al discriminant. Manté l'estructura característica de GAN amb dues xarxes entrenades simultàniament: discriminant (D) i generador (G).



```
%tensorflow_version 1.x
!nvidia-smi
```

TensorFlow 1.x selected.
Thu Dec 30 06:02:27 2021

NVIDIA-SMI 495.44		Driver Version: 460.32.03		CUDA Version: 11.2	
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile Uncorr. ECC
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util Compute M.
					MIG M.
0	Tesla P100-PCIE...	Off	00000000:00:04.0	Off	0
N/A	36C	P0	27W / 250W	0MiB / 16280MiB	0% Default N/A

Processes:					
GPU	GI	CI	PID	Type	Process name
ID			ID		GPU Memory Usage
No running processes found					

Figura 11. Comanda nvidia-smi. Usem una GPU Tesla P-100.

```
[ ] dataset_name = 'dr_cyz'
datasets_source_path = project_path / 'datasets' / 'source' / (dataset_name + '.zip')
if datasets_source_path.is_dir():
    print("Dataset ready for import.")
else:
    print('Upload your images dataset as {}'.format(datasets_source_path))
```

Upload your images dataset as /content/drive/MyDrive/StyleGAN2-ADA/datasets/source/dr_cyz.zip

Figura 12. Importem una selecció de les dades (dr_cyz.zip) del subconjunt original de manera que sigui adequat per la generació sintètica d'imatges. En concret es tracta de 5025 imatges que provenen de la càmera esquerra i dreta Mastcam-Z de la missió Perseverance i que en el nivell t-SNE apareixen a la part central com a mostres de terreny.

PAC. Pràctica 2.

J. de Curtò i DíAz & I. de Zarzà i Cubero.

c@decurso.be z@dezarza.be

```
▶ local_original_images_path = local_dataset_path / 'c'
local_images_path = local_dataset_path / 'dr_c_y_z_512'
local_dataset_path /= 'tfr_512'
!apt install imagemagick

if (local_dataset_path).is_dir():
    print('\N{Heavy Exclamation Mark Symbol} Dataset already created \N{Heavy Exclamation Mark Symbol}')
    print('Delete current dataset folder ({})) to regenerate tfrecords.'.format(local_dataset_path))
else:
    !mkdir "local_images_path"
    !mogrify -path "{local_images_path}" -resize 512x512! "{local_original_images_path} / '*.png'"
    !mkdir "local_dataset_path"
    !python "{stylegan2_repo_path} / 'dataset_tool.py'" create_from_images \
        "{local_dataset_path}" "{local_images_path}"
```

Figura 13. Pre-processem les imatges fent un redimensionat a tamany 256x256, 512x512 i 1024x1024 pel posterior entrenament. També es converteixen les imatges a format TFRecords per ser importades a Tensorflow.

```
⌚ training_path = project_path / 'training' / dataset_name
if not training_path.is_dir():
    !mkdir "training_path"

#how often should the model generate samples and a .pkl file
snapshot_count = 2
#should the images be mirrored left to right?
mirrored = True
#should the images be mirrored top to bottom?
mirroredY = False
#metrics?
metric_list = None
#augments
augs = 'bgc'

resume_from = 'noresume'

!python "{stylegan2_repo_path} / 'train.py'" --outdir="{training_path}" \
    --data="{local_dataset_path}" --resume="{resume_from}" \
    --snap={snapshot_count} --augpipe={augs} \
    --mirror={mirrored} --mirrory={mirroredY} --cfg={'auto'} \
    --metrics={metric_list} #--dry-run
```

Figura 14. Entrenem el model GAN de manera que fa una selecció dels hyperparametres prèvia a l'entrenament (--cfg='auto') i que entra els pesos a aprendre des de zero (--resume='noresume').

```
⌚ from numpy import random
seed_init = random.randint(10000)
nbr_images = 100
|
generation_from =
'/content/drive/MyDrive/StyleGAN2-ADA/training/dr_cyz/00000-tfr-mirror-autol-bgc-noresume-256_de_curto_and_de_zarza/network-snapshot-000798.pkl'
!python "{stylegan2_repo_path} / 'generate.py'" generate-images \
    --outdir=(project_path / 'out' / 'dr_cyz_256_100') --trunc=0.7 \
    --seeds=(seed_init)-(seed_init+nbr_images-1) --create-grid \
    --network=(generation_from)
```

Figura 15. Generem imatges sintètiques a partir del model entrenat. En concret un set de 100, 1000 i 10000 a mida 256.

Generades les imatges sintètiques, plantegem una primera analisi de les mateixes mitjançant tècniques estadístiques. Per fer-ho, utilitzem el set de 1000 mostres; les re-escalem a dimensió 64x64 i les posem en forma de vector pla, Figura 16. Duem a terme dues versions, una amb les imatges RGB i l'altra amb les mostres en escala de grisos.

```

for filename in os.listdir(folder):
    image = cv2.imread(os.path.join(folder,filename))
    if image is not None:
        #image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        image = cv2.resize(image, (64,64))
        image = image.flatten()
        data.append([image, folder + filename])

for filename2 in os.listdir(folder2):
    image2 = cv2.imread(os.path.join(folder2,filename2))
    if image2 is not None:
        #image2 = cv2.cvtColor(image2, cv2.COLOR_BGR2GRAY)
        image2 = cv2.resize(image2, (64,64))
        image2 = image2.flatten()
        data2.append([image2, folder2 + filename2])

```

Figura 16. Per dur a terme una anàlisi comparativa de les mostres generades i les mostres originals, utilitzem el set de 1000 mostres generades a 256x256. Reescalem les imatges a 64x64 i les convertim a un vector pla. Fem dos tipus de còmput, amb les imatges en rgb i amb les imatges en escala de grisos.

Proposem computar la intensitat mitjana de les imatges per dur a terme la comparació, Figura 17; i posteriorment utilitzar els mètodes estadístics treballats durant el curs.

```

▶ mean_features = []
for c in features:
    mean_features.append(c.mean())

mean_features2 = []
for c2 in features2:
    mean_features2.append(c2.mean())

```

Figura 17. Per realitzar els testos estadístics proposem computar els valors mitjans de les intensitats dels pixels.

Així, es comprova la igualtat de variàncies fent servir el test de Levene i la normalitat de la distribució de les mostres. A continuació, donat que no es compleix normalitat utilitzem el test no paramètric de Kruskal-Wallis, Figures 18 (cas rgb) i 19 (cas escala de grisos).

```

▶ import scipy
print(scipy.stats.levene(mean_features, mean_features2)) #Levene
print(scipy.stats.levene(mean_features, mean_features2, center='mean')) #Brown-Forsyth
print(scipy.stats.bartlett(mean_features, mean_features2)) #Bartlett
print(scipy.stats.normaltest(mean_features)) #Normality of original samples
print(scipy.stats.normaltest(mean_features2)) #Normality of generated samples
print(scipy.stats.kruskal(mean_features,mean_features2)) #Kruskal-Wallis

⇨ LeveneResult(statistic=201.5582877437271, pvalue=5.050293281344776e-45)
LeveneResult(statistic=201.8255157607196, pvalue=4.434997564711846e-45)
BartlettResult(statistic=946.0806030817251, pvalue=9.434643098678378e-208)
NormaltestResult(statistic=2154.565562637841, pvalue=0.0)
NormaltestResult(statistic=1.265450664334759, pvalue=0.5311422875395507)
KruskalResult(statistic=29.23336319362794, pvalue=6.416479346217565e-08)

```

Figura 18. Imatges en color. Computem el test de Levene (i dues alternatives: Brown-Forsyth i Bartlett) per comprovar la igualtat de variàncies de les dues distribucions (rejectem la hipòtesi nul·la). També comprovem la normalitat de les mostres originals (rejectem la hipòtesi nul·la) i sintètiques (acceptem la hipòtesi nul·la). Finalment, apliquem el mètode de Kruskal-Wallis (rejectem la hipòtesi nul·la).

PAC. Pràctica 2.

J. de Curtò i DíAz & I. de Zarzà i Cubero.

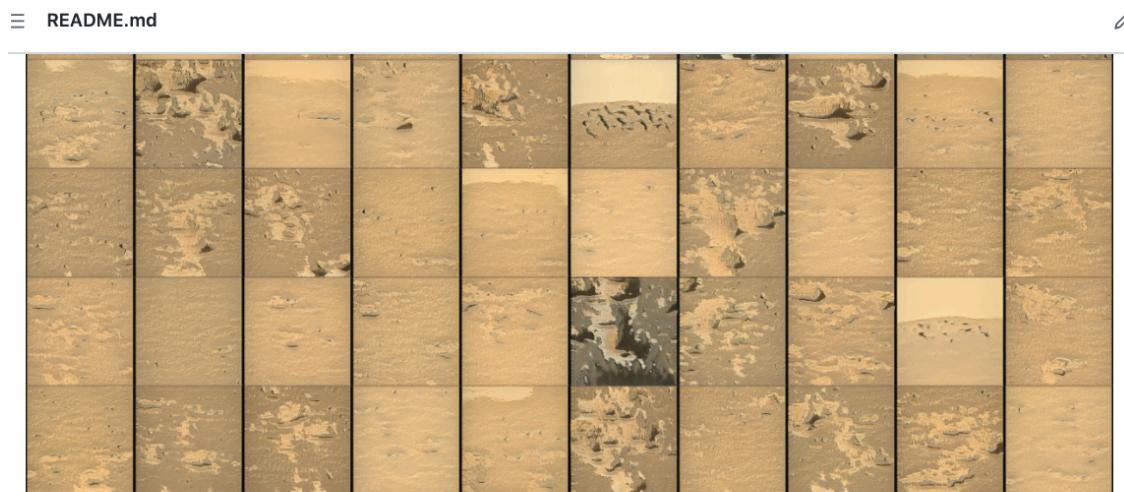
c@decurto.be z@dezarza.be

```
▶ import scipy
print(scipy.stats.levene(mean_features, mean_features2)) #Levene
print(scipy.stats.levene(mean_features, mean_features2, center='mean')) #Brown-Forsyth
print(scipy.stats.bartlett(mean_features, mean_features2)) #Bartlett
print(scipy.stats.normaltest(mean_features)) #Normality of original samples
print(scipy.stats.normaltest(mean_features2)) #Normality of generated samples
print(scipy.stats.kruskal(mean_features,mean_features2)) #Kruskal-Wallis
```

```
⇨ LeveneResult(statistic=156.12869754877337, pvalue=2.1729631188690332e-35)
LeveneResult(statistic=166.13819743144077, pvalue=1.611279115945387e-37)
BartlettResult(statistic=874.8317447750159, pvalue=2.905247248253299e-192)
NormaltestResult(statistic=2571.783381290372, pvalue=0.0)
NormaltestResult(statistic=0.11406195050626958, pvalue=0.9445648107792636)
KruskalResult(statistic=30.97421391554919, pvalue=2.6147941787875364e-08)
```

Figura 19. Imatges en escala de grisos. Computem el test de Levene (i dues alternatives: Brown-Forsyth i Bartlett) per comprovar la igualtat de variàncies de les dues distribucions (rejectem la hipòtesi nul·la). També comprovem la normalitat de les mostres originals (rejectem la hipòtesi nul·la) i sintètiques (acceptem la hipòtesi nul·la). Finalment, apliquem el mètode de Kruskal-Wallis (rejectem la hipòtesi nul·la).

Proporcionem al repositori el set de mostres utilitzades per entrenar les xarxes, les mostres sintètiques generades, el Network checkpoint resultant i els notebooks en python per reproduir els experiments, Figura 20.



[Link to synthetic samples from Perseverance generated using Stylegan2-ada. Set of 100. Size 256x256. Preview.](#)

[Link to synthetic samples from Perseverance generated using Stylegan2-ada. Set of 1000. Size 256x256. Preview.](#)

[Link to synthetic samples from Perseverance generated using Stylegan2-ada. Set of 10000. Size 256x256. Preview.](#)

[Subset of CyZ used to train Stylegan2-ada. Size 64x64. TFRecord.](#)

[Subset of CyZ used to train Stylegan2-ada. Size 128x128. TFRecord.](#)

[Subset of CyZ used to train Stylegan2-ada. Size 256x256. TFRecord.](#)

[Subset of CyZ used to train Stylegan2-ada. Size 512x512. TFRecord.](#)

[Subset of CyZ used to train Stylegan2-ada. Size 1024x1024. TFRecord.](#)

[Network checkpoint to generate the samples. Trained using 1 x NVIDIA Tesla P-100 at size 256x256 during 48h.](#)

Figura 20. Proporcionem el subconjunt de dades utilitzat per fer els experiments (5025 imatges) en format png, csv i TFRecords. També fem el release dels sets sintètics de 100, 1000 i 10000 mostres. Així com el Network checkpoint utilitzat per generar les mostres i els notebooks en python per reproduir els experiments.

5. Representació dels resultats a partir de taules i gràfiques.

Es realitza la presentació de les anàlisis de manera visual (Figura 4: K-means Clustering i Figura 9: t-SNE), i tot seguit també exemples pràctics dels algoritmes de segmentació semàntica (Figura 28) i generació sintètica d'imatges en el subconjunt de dades seleccionat (Figures 21 i 23).

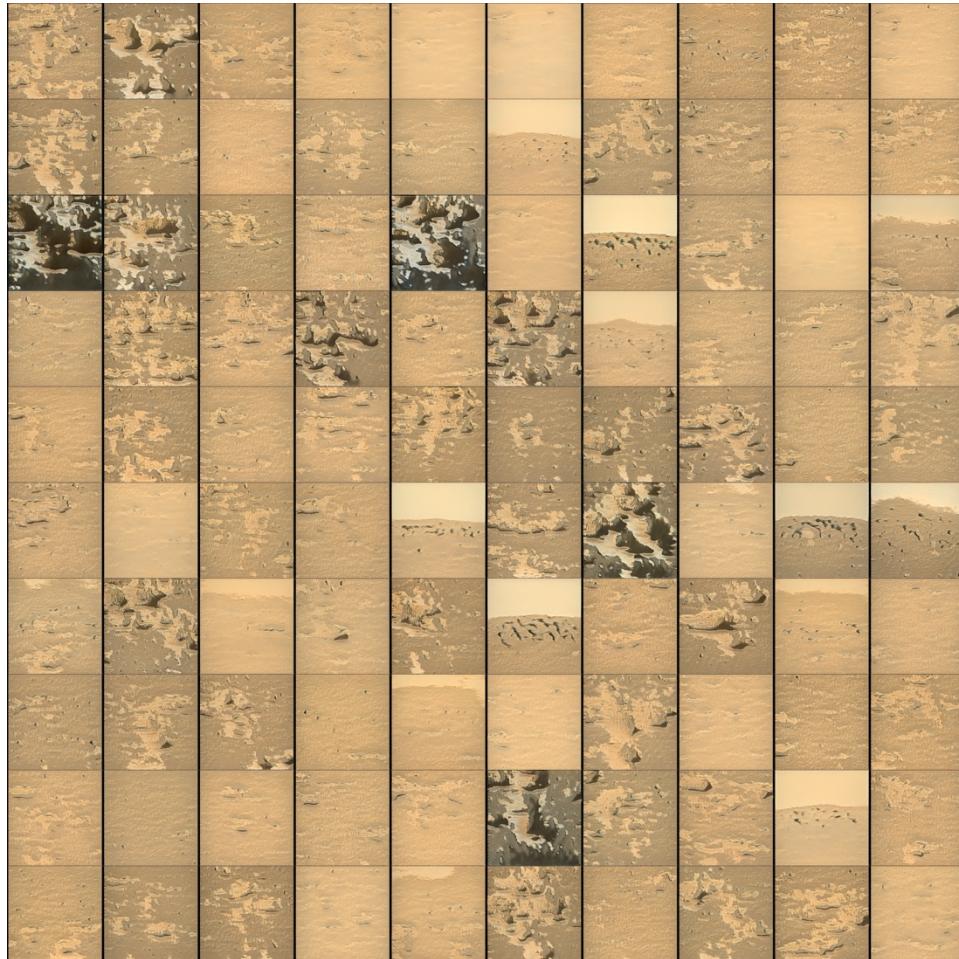


Figura 21. Generative Adversarial Networks. Grid de 100 imatges sintètiques entrenades amb una selecció de mostres de la missió Perseverance a mida 256x256.

També és possible fer una visualització en format vídeo explorant el latent space de la distribució generada.

```
❶ from numpy import random
walk_types = ['line', 'sphere', 'noiseloop', 'circularloop']
latent_walk_path = project_path / 'out' / 'latent_walk_sphere_cyz_256'
if not latent_walk_path.is_dir():
    mkdir(latent_walk_path)

explored_network =
    '/content/drive/MyDrive/StyleGAN2-ADA/training/dr_cyz/00000-tfr-mirror-autol-bgc-noresume-256_de.curto_and_de.zarza/network-snapshot-000798.pkl'

seeds = [random.randint(10000) for i in range(10)]
print(','.join(map(str, seeds)))
print("base seeds:", seeds)
!python "(stylegan2_repo_path / 'generate.py')" generate-latent-walk --network="(explored_network)" \
    --outdir="(latent_walk_path)" --trunc=0.7 --walk-type="(walk_types[1])" \
    --seeds=(','.join(map(str, seeds))) --frames {len(seeds)*20}
```

Figura 22. Exploració del latent space de la distribució generada a partir del model entrenat.



Figura 23. Frame del video de l'exploració en esfera del latent space.

Per últim, usarem una tècnica de segmentació semàntica [16,18]; Deeplab, Figura 20, entrenada en un dataset terrestre amb imatges de paisatges [17] per proporcionar màscares semàntiques aproximades de les imatges, Figura 21, 22, 23. Això ens permetrà entendre el contingut visual d'aquestes i detectar imatges que no aporten al conjunt i aquelles que són més significatives per una tasca concreta, Figura 24.

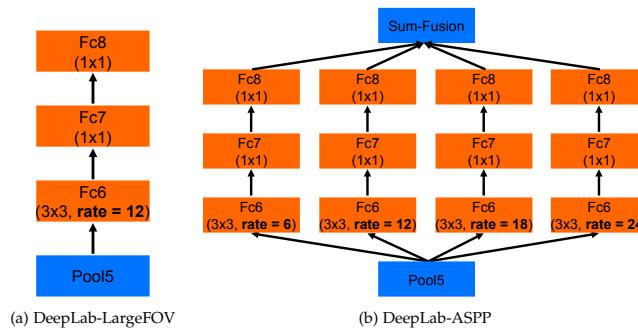


Figura 24. L'arquitectura de segmentació Deeplab utilitz a filters a diferents mides de manera que és capaç de capturar objectes i el context d'aquests a diferents escales.

```
#@title Downloading and extracting the model checkpoints
MODEL_NAME = 'deeplabv3_mnv2_ade20k_train_2018_12_03' # @param ['deeplabv3_mnv2_ade20k_train_2018_12_03']

DOWNLOAD_URL_PREFIX = 'http://download.tensorflow.org/models/'
MODEL_URLS = {
    'deeplabv3_mnv2_ade20k_train_2018_12_03':
        'deeplabv3_mnv2_ade20k_train_2018_12_03.tar.gz',
    'deeplabv3_xception_ade20k_train':
        'deeplabv3_xception_ade20k_train_2018_05_29.tar.gz',
}
MODEL_TAR = MODEL_URLS[MODEL_NAME]
MODEL_URL = DOWNLOAD_URL_PREFIX + MODEL_TAR

# Download
!wget -O {MODEL_TAR} {MODEL_URL}

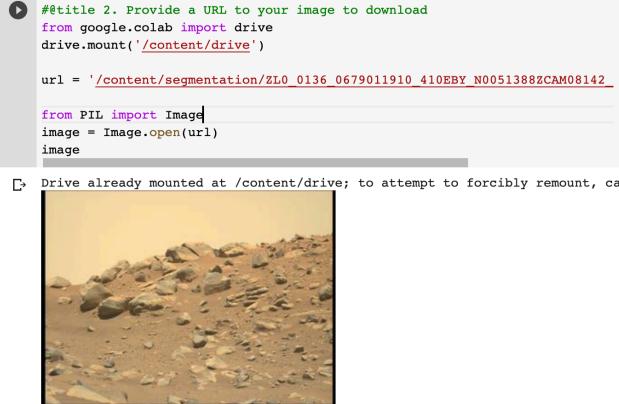
# Extract
MODEL_FILE = !tar -zxf {MODEL_TAR} --wildcards --no-anchored 'frozen_inference_graph.pb'
MODEL_FILE = MODEL_FILE[0].strip()
print('Frozen graph file path:', MODEL_FILE)
```

Figura 25. Descarreguem un model de Deeplab entrenat amb el dataset ADE20k, molt adequat per instance segmentation de paisatges.

PAC. Pràctica 2.

J. de Curtò i DíAz & I. de Zarzà i Cubero.

c@decurto.be z@dezarza.be



```
#title 2. Provide a URL to your image to download
from google.colab import drive
drive.mount('/content/drive')

url = '/content/segmentation/ZL0_0136_0679011910_410EBY_N00513882CAM08142'

from PIL import Image
image = Image.open(url)
image
```

2. Provide a URL to your image to download

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).



Figura 26. Seleccionem una mostra del subconjunt d'imatges del Perseverance.



```
vis_segmentation(cropped_image, seg_map)
```

Figura 27. Apliquem el model a la imatge seleccionada, previ redimensionament.

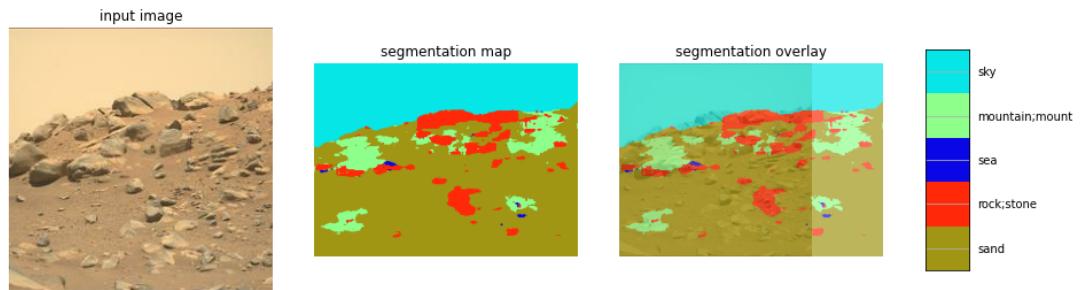


Figura 28. Exemple de segmentació semàntica amb una xarxa entrenada amb el dataset ADE20K. Es pot observar com es segmenta correctament alguns dels atributs corresponents a l'entorn del planeta Mart tals com el cel, les roques i el terreny.

6. Resolució del problema. A partir dels resultats obtinguts, quines són les conclusions?

Els resultats permeten respondre al problema?

La visualització t-SNE ens permet seleccionar un subconjunt d'imatges adequades per les tasques posteriors. En concret som capaços d'entrenar un model de generació sintètica d'imatges de Mart mitjançant una tècnica d'aprenentatge no supervisat (Generative Adversarial Networks) que ens permet generar un nou dataset amb subconjunts de 100, 1000 i 10000 imatges molt útils per entendre la distribució original de dades i per tasques posteriors com per exemple per fonamentar les textures d'un simulador 3D del planeta Mart. Aquestes noves dades es proporcionen al repositori. A més, també mostrem l'exemple pràctic per generar màscares de segmentació utilitzant un model pre-entrenat; essencialment útil per les tasques de percepció i localització d'un robot mòbil (e.g. SLAM).

7. Codi: Cal adjuntar el codi, preferiblement en R, amb el que s'ha realitzat la neteja, anàlisi i representació de les dades. Si ho preferiu, també podeu treballar en Python.

El codi en Python, les imatges sintètiques i els models entrenats es poden trobar al repositori:

DrCyZ: Techniques for analyzing and extracting useful information from CyZ.
<https://github.com/decurtoidiaz/drcyz>

Referències:

- [1] De Curtò and De Zarzà. CyZ: MARS Space Exploration Dataset. Zenodo. 2021.
<https://doi.org/10.5281/zenodo.5655473>
- [2] Spirit: <https://mars.nasa.gov/mer/gallery/all/spirit.html>
- [3] Opportunity: <https://mars.nasa.gov/mer/gallery/all/opportunity.html>
- [4] Curiosity: <https://mars.nasa.gov/msl/multimedia/raw-images/>
- [5] Perseverance: <https://mars.nasa.gov/mars2020/multimedia/raw-images/>
- [6] Maki et al. 2020. The Mars 2020 Engineering Cameras and Microphone on the Perseverance Rover: A Next-generation Imaging System for Mars Exploration.
<https://link.springer.com/article/10.1007/s11214-020-00765-9>
- [7] Lamarre et al. 2020. The Canadian Planetary Emulation Terrain Energy-Aware Rover Navigation Dataset.
<https://starslab.ca/enav-planetary-dataset/>
- [8] ESA Robotics Dataset. 2015. Katwijk Beach Planetary Rover Dataset.
<https://robotics.estec.esa.int/datasets/katwijk-beach-11-2015/>
- [9] Shlens. A Tutorial on Principal Component Analysis. 2005.
<https://www.cs.cmu.edu/~elaw/papers/pca.pdf>
- [10] Van der Maaten. Accelerating t-SNE using Tree-Based Algorithms. Journal of Machine Learning Research 15(Oct):3221-3245, 2014.
- [11] Van der Maaten and G.E. Hinton. Visualizing Non-Metric Similarities in Multiple Maps. Machine Learning 87(1):33-55, 2012.
- [12] Van der Maaten. Learning a Parametric Embedding by Preserving Local Structure. In Proceedings of the Twelfth International Conference on Artificial Intelligence & Statistics (AI-STATS), JMLR W&CP 5:384-391, 2009.
- [13] Van der Maaten and Hinton. Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research 9(Nov):2579-2605, 2008.
- [14] https://en.wikipedia.org/wiki/Kullback-Leibler_divergence
- [15] Goodfellow et al. Generative Adversarial Networks. NIPS. 2014.
- [16] He et al. Mask R-cnn. ICCV. 2017.
- [17] Zhou et al. Semantic Understanding of Scenes through the ADE20K Dataset. IJCV. 2016.
- [18] Chen et al. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. TPAMI. 2016.
- [19] Karras et al. Training Generative Adversarial Networks with Limited Data. NeurIPS. 2020.

Contribucions	Signatura
Investigació prèvia	JDC, IDZ
Redacció de les respostes	JDC, IDZ
Desenvolupament del codi	JDC, IDZ