

REVISION

1. Premier pipeline 3D CNN
2. Retour au 2D CNN, pipeline simplifié

TELECHARGEMENTS & INSTALLATION

Deux contenus sont téléchargeables :

- les datasets,
- les scripts Python.

Tous les scripts Python, et les modèles générés, sont hébergés par GitHub et gérés en configuration.

Les datasets ne sont pas hébergés par GitHub compte tenu de la quantité de données. **Pour éviter la synchronisation automatique de ces datasets sur GitHub, ils doivent être stockés dans un dossier local dédié indépendant du repo GitHub local géré en configuration.**

TELECHARGEMENT DES DATASET

Les datasets sont disponibles en téléchargement via **OneDrive** :

<https://1drv.ms/u/s!AgPL1p8-kSaRg9RDySAEe-KTSa6sng>

Ces fichiers compressés contiennent les images, et les labelles (direction, gaz et position ligne) correspondant.

INSTALLATION LOCALE DES DATASETS

- 1) **Créer un dossier local pour stocker les datasets** et les vidéos. Exemple : c:/datasets
- 2) **Décompresser les datasets téléchargés** dans ce dossier local.
- 3) **Créer les dossiers « train_valid_dataset » et « video »** dans ce même dossier local.
- 4) Le contenu du dossier local doit être le suivant :

Nom	Modifié le	Type	Taille
dataset_001	10/09/2019 23:58	Dossier de fichiers	
dataset_002	14/09/2019 11:58	Dossier de fichiers	
dataset_003	12/09/2019 14:29	Dossier de fichiers	
dataset_004	12/09/2019 14:06	Dossier de fichiers	
dataset_005	11/09/2019 21:08	Dossier de fichiers	
dataset_006	11/09/2019 23:30	Dossier de fichiers	
dataset_007	12/09/2019 14:17	Dossier de fichiers	
dataset_008	12/09/2019 14:23	Dossier de fichiers	
train_valid_dataset	14/09/2019 15:11	Dossier de fichiers	
video	14/09/2019 19:13	Dossier de fichiers	

SCRIPTS PYTHON

Les scripts Python sont disponibles sur GitHub :

GitHub/pat92fr/Carre92/02 - Logiciel PC/04 - CNN Pipeline Line Detection/

<https://github.com/pat92fr/Carre92/tree/master/02%20-%20Logiciel%20PC/04%20-%20CNN%20Pipeline%20Line%20Detection>

- 1) **Utiliser GitHub Desktop** pour télécharger et synchroniser les scripts en local.

Note : Sinon, copier tous les scripts dans un dossier local et créer le dossier « model ».

- 2) Le contenu du dossier local doit être le suivant :

Nom	Modifié le	Type	Taille
model	14/09/2019 20:43	Dossier de fichiers	
01_Prepate_Dataset_4_Training.py	14/09/2019 15:49	Python File	3 Ko
02_Train.py	14/09/2019 18:35	Python File	8 Ko
10_Dataset_Labeling_Tool.py	14/09/2019 15:07	Python File	5 Ko
11_Labels_2_Video.py	14/09/2019 18:30	Python File	4 Ko
12_Predict_2_Video.py	14/09/2019 20:27	Python File	6 Ko
dos.bat	14/09/2019 16:43	Fichier de comma...	1 Ko
my_constants.py	14/09/2019 16:56	Python File	2 Ko
my_datasettools.py	14/09/2019 11:50	Python File	1 Ko
my_modeltools.py	14/09/2019 15:52	Python File	3 Ko
my_parameters.py	14/09/2019 20:46	Python File	2 Ko
notice.docx	15/09/2019 12:15	Document Micros...	398 Ko
notice.pdf	07/09/2019 19:46	Adobe Acrobat D...	338 Ko
README.txt	14/09/2019 12:07	Document texte	1 Ko

- 3) **Modifier « my_parameters.py »** pour indiquer le chemin vers le dossier local des datasets

```
my_parameters.py x
1  ## PARAMETERS #####
2
3  # inputs/outputs
4  base_dataset_directory = "c:/GitHub/datasets/" # change to your local directory where datasets are decompressed
```

FICHIER DE CONFIGURATION

Le fichier « **my_parameters.py** » contient toutes les données de configuration de toutes les étapes.

PIPELINE (2 ETAPES)

ETAPE 1 : SYNTHÈSE DU DATASET D'APPRENTISSAGE



- 1) Sélectionner les datasets à utiliser pour l'apprentissage, en modifiant la variable « **train_valid_dataset_list** » du fichier de configuration « **my_parameters.py** » :

```
my_parameters.py x
1  ## PARAMETERS #####
2
3  # inputs/outputs
4  base_dataset_directory = "c:/GitHub/datasets/" # change to your local directory where datasets are decompressed
5  train_and_valid_dataset_list = [ "dataset_003", "dataset_004", "dataset_007", "dataset_008" ]
```

- 2) Lancer « **01_Prepare_Dataset_4_Training.py** » pour générer le dataset d'apprentissage.

```
C:\GitHub\Carre92\02 - Logiciel PC\04 - CNN Pipeline Line Detection>01_Prepare_Dataset_4_Training.py
dataset_003
Loading label file...
Done.
Parsing label file...
m=10813 examples
Done.
Building dataset...
Done.
dataset_004
Loading label file...
Done.
Parsing label file...
m=23032 examples
Done.
Building dataset...
Done.
dataset_007
Loading label file...
Done.
Parsing label file...
m=19134 examples
Done.
Building dataset...
Done.
dataset_008
Loading label file...
Done.
Parsing label file...
m=12926 examples
Done.
Building dataset...
Done.
(131810, 90, 160, 1)
(131810,)
Done.
Splitting whole dataset into train and cross-validation datasets...
Xtrain, size: 125219
Ytrain, size: 125219
Xvalid, size: 6591
Yvalid, size: 6591
Done.
Compressing and writing file to disk...
Done.
C:\GitHub\Carre92\02 - Logiciel PC\04 - CNN Pipeline Line Detection>
```

- 3) Le fichier généré est stocké localement dans le dossier « **train_valid_dataset** », du dossier local où se trouve tous les datasets. Note : le précédent dataset d'apprentissage est automatiquement sauvegardé (.bak).

Nom	Modifié le	Type	Taille
 train_and_valid_dataset.npz	15/09/2019 12:35	Fichier NPZ	1 854 609 Ko
 train_and_valid_dataset.npz.bak	15/09/2019 12:35	Fichier BAK	1 157 105 Ko

ETAPE 2 : APPRENTISSAGE DU CNN

- 1) **Définir l'architecture du CNN**, en modifiant la variable « **layers** » du fichier de configuration « **my_parameters.py** » :

```
my_parameters.py x
7
8 # CNN parameters
9 layers = [
10     ('crop2D', 26), ## do not change
11     ('norm', 0), ## do not change
12     ('conv2D', 8, (5,5), (2,2), 'relu'), ## filters, kernel_size, stride, activation
13     ('conv2D', 8, (5,5), (2,2), 'relu'), ## filters, kernel_size, stride, activation
14     ('maxpooling2D', (2,2), (2,2)), ## size, stride
15     ('dropout', 0.2), ## %
16     ('flatten', 0), ## do not change ## CONV to FC transition
17     ('fc', 8, 'tanh', 0.0001), ## units, activation, l2
18     ('dropout', 0.2), ## %
19     ('fc', 1, 'linear', 0.0001), ## units, activation, l2
20     # do not change unit and activation of the last layer (one output, no activation = linear)
21 ]
```

- La couche **crop2D** supprime les 26 premières lignes de l'image (160x90) de telle manière que le CNN traite une image tronquée (160x64).
- Le couche **norm** normalise les valeurs RGB de l'image dans la plage [0.0 , 1.0]
- Une couche de filtres de convolution **conv2D** prend en argument le nombre de filtres, la taille des filtres (h,w), le déplacement des filtres (h,w) et la fonction d'activation.
- La couche **maxpooling** prend en argument la taille (h,w), et le déplacement (h,w),
- La couche **dropout** prend en argument le % de suppression.
- La couche de neurones **fc** prend en argument le nombre d'unités, la fonction d'activation et lambda (régularisation L2).

- 2) **Ajuster les hyper paramètres d'apprentissage du CNN**, en modifiant la variable « **hyp...** » du fichier de configuration « **my_parameters.py** » :

```
my_parameters.py x
24 # hyperparameters
25 hyp_train_valid_dataset_ratio = 0.05
26 hyp_batch_size = 256
27 hyp_epoch = 100
28 hyp_lr = 0.0001
29 hyp_lr_decay = 0.0
30 hyp_min_delta=0.0002
31 hyp_patience=12
```

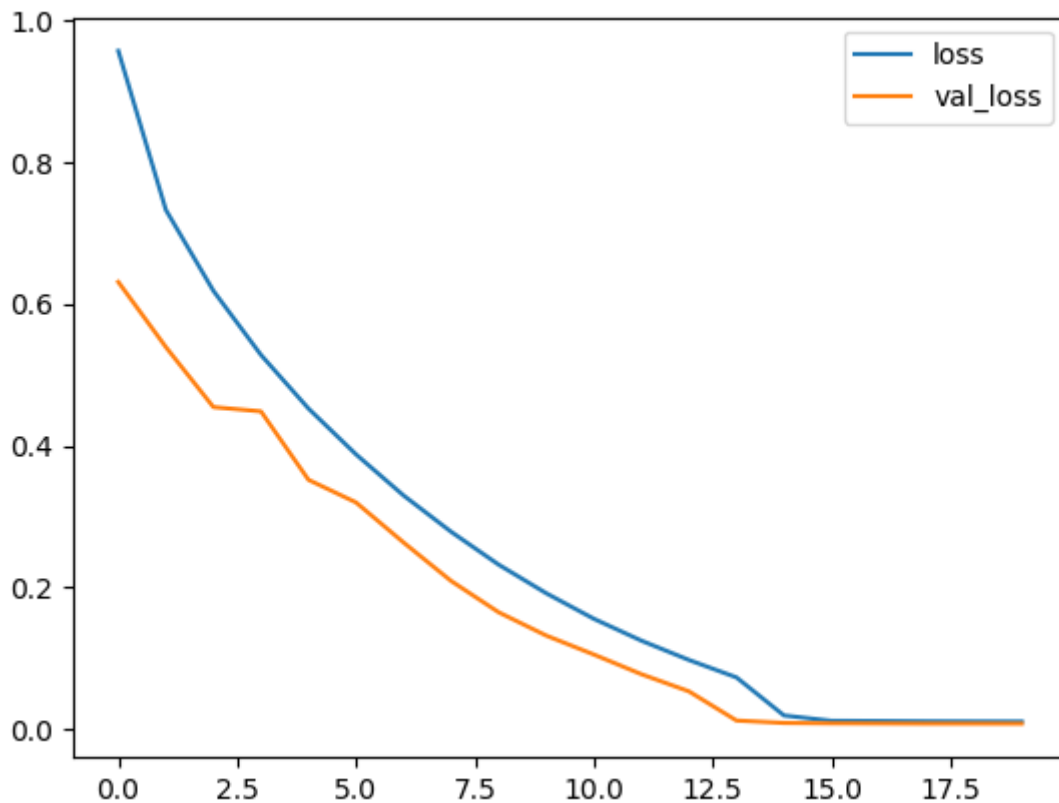
3) Lancer « **02_Train.py** » pour lancer l'apprentissage.

L'architecture modèle est affichée au début de l'apprentissage :

Layer (type)	Output Shape	Param #
cropping2d_1 (Cropping2D)	(None, 64, 160, 1)	0
lambda_1 (Lambda)	(None, 64, 160, 1)	0
conv2d_1 (Conv2D)	(None, 30, 78, 8)	208
batch_normalization_1 (Batch Normalization)	(None, 30, 78, 8)	32
activation_1 (Activation)	(None, 30, 78, 8)	0
conv2d_2 (Conv2D)	(None, 13, 37, 8)	1608
batch_normalization_2 (Batch Normalization)	(None, 13, 37, 8)	32
activation_2 (Activation)	(None, 13, 37, 8)	0
max_pooling2d_1 (MaxPooling2D)	(None, 6, 18, 8)	0
dropout_1 (Dropout)	(None, 6, 18, 8)	0
flatten_1 (Flatten)	(None, 864)	0
dense_1 (Dense)	(None, 8)	6920
batch_normalization_3 (Batch Normalization)	(None, 8)	32
activation_3 (Activation)	(None, 8)	0
dropout_2 (Dropout)	(None, 8)	0
dense_2 (Dense)	(None, 1)	9
batch_normalization_4 (Batch Normalization)	(None, 1)	4
activation_4 (Activation)	(None, 1)	0
Total params: 8,845		
Trainable params: 8,795		
Non-trainable params: 50		
Training model...		

Les métriques sont affichées au fil de l'apprentissage (sortie standard et graphique) :

```
Epoch 18/100
- 17s - loss: 0.0113 - mean_squared_error: 0.0109 - val_loss: 0.0080 - val_mean_squared_error: 0.0077
Epoch 19/100
- 17s - loss: 0.0111 - mean_squared_error: 0.0108 - val_loss: 0.0080 - val_mean_squared_error: 0.0077
Epoch 20/100
- 17s - loss: 0.0109 - mean_squared_error: 0.0106 - val_loss: 0.0079 - val_mean_squared_error: 0.0077
Epoch 21/100
- 17s - loss: 0.0107 - mean_squared_error: 0.0105 - val_loss: 0.0078 - val_mean_squared_error: 0.0076
```



- 4) A la fin de l'apprentissage, le modèle généré est stocké localement dans le dossier « **model** » dans un fichier « .h5 » daté et dans un fichier générique « model.h5 ». Note : le précédent modèle est automatiquement sauvegardé (.bak).

Nom	Modifié le	Type	Taille
history_Sat_Sep_14_18-53-46_2019.png	14/09/2019 18:53	Fichier PNG	27 Ko
history_Sat_Sep_14_20-43-02_2019.png	14/09/2019 20:43	Fichier PNG	26 Ko
model.h5	14/09/2019 20:43	Fichier H5	172 Ko
model.h5.bak	14/09/2019 20:43	Fichier BAK	172 Ko
model.h5_Sat_Sep_14_18-53-46_2019.h5	14/09/2019 18:53	Fichier H5	172 Ko
model.h5_Sat_Sep_14_20-43-01_2019.h5	14/09/2019 20:43	Fichier H5	172 Ko
report_Sat_Sep_14_18-53-47_2019.txt	14/09/2019 18:53	Document texte	1 Ko
report_Sat_Sep_14_20-43-03_2019.txt	14/09/2019 20:43	Document texte	1 Ko

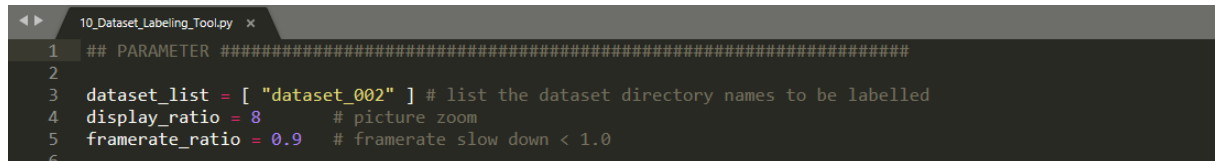
- 5) Le graphique des métriques est stocké localement dans le dossier « model » dans un fichier daté « .png ».
- 6) Un rapport d'apprentissage est stocké localement dans le dossier « model » dans un fichier daté « .txt ». Ce rapport rappelle succinctement l'architecture du CNN et les métriques finales pour le dataset de validation (Loss et MSE).

OUTILS

OUTIL DE LABELISATION D'UN DATASET SOURCE

Le script Python de labélisation est « 10_Dataset_Labeling_Tool.py ». Il génère un fichier « label.txt » dans le dossier du dataset sélectionné.

- 1) Sélectionner le dataset à labéliser, en modifiant la variable « **dataset_list** » du script.



```
10_Dataset_Labeling_Tool.py x
1  ## PARAMETER #####
2
3  dataset_list = [ "dataset_002" ] # list the dataset directory names to be labelled
4  display_ratio = 8               # picture zoom
5  framerate_ratio = 0.9           # framerate slow down < 1.0
6
```

- 2) Lancer le script et labéliser à la souris.

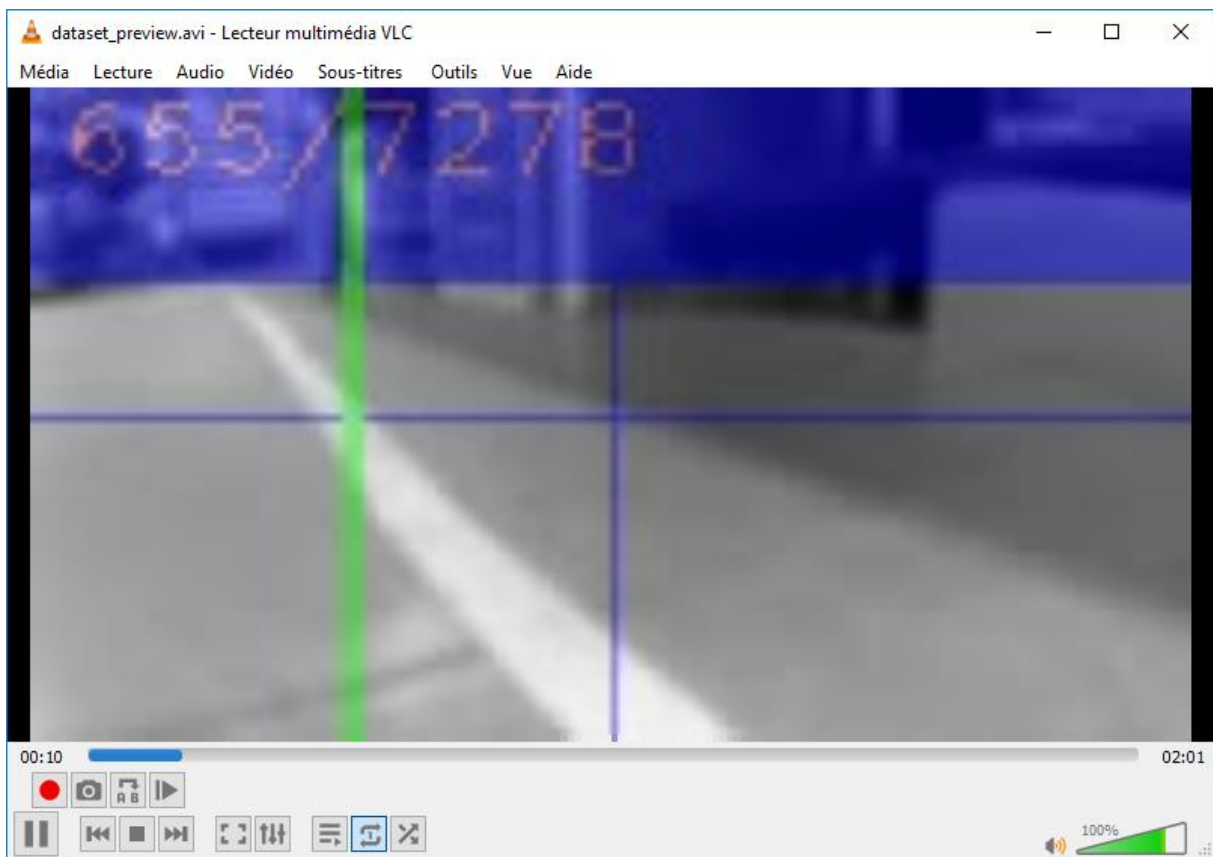
OUTIL DE VISUALISATION D'UN DATASET SOURCE LABELISES

Le script Python de visualisation est « 11_Labels_2_Video.py ». Il génère un fichier video « dataset_preview.avi » dans le dossier « video » du dossier local contenant les datasets.

- 1) Sélectionner le dataset à visualiser, en modifiant la variable « **dataset_list** » du script.

```
11_Labels_2_Video.py x
1  ## PARAMETER #####
2
3  dataset_list = [ "dataset_002" ] # list the dataset directory names to be visualized
4
```

- 2) Lancer le script.
- 3) Ouvrir la vidéo générée.



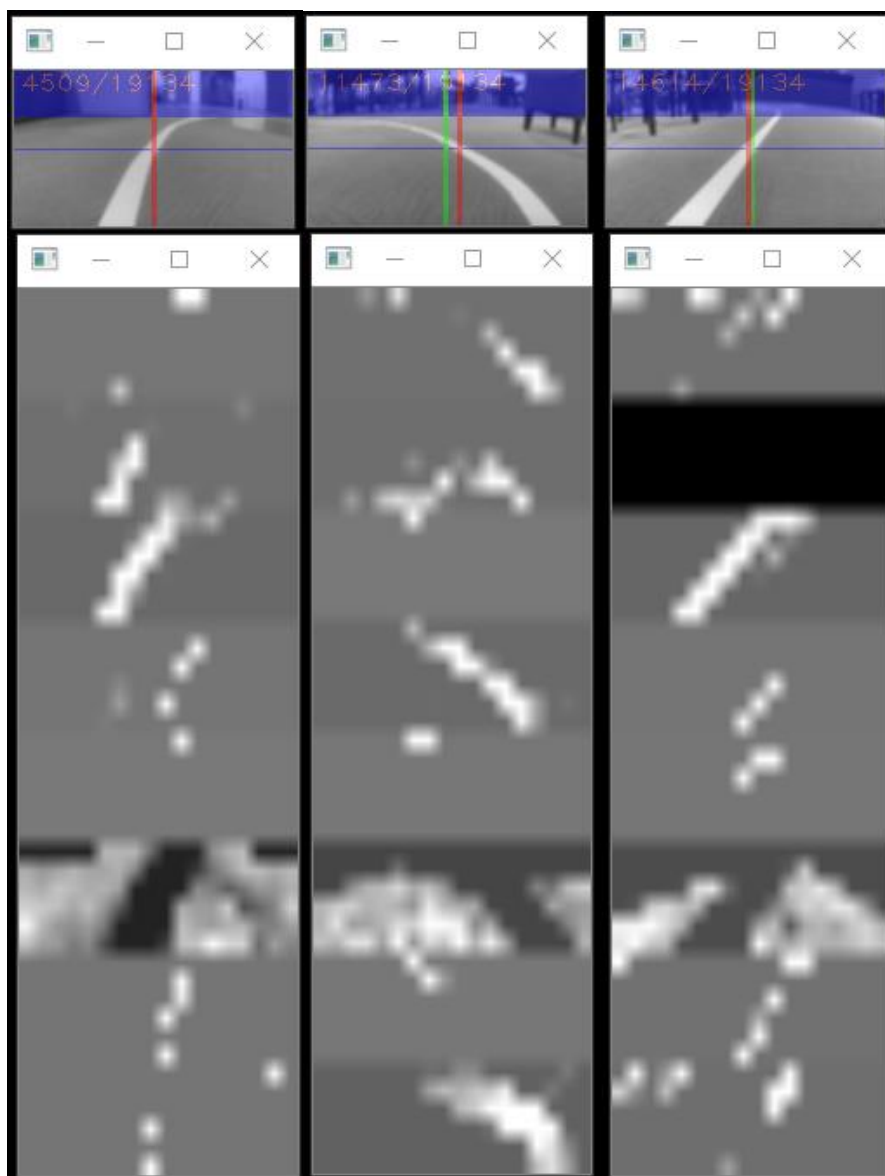
OUTIL DE VISUALISATION D'UN DATASET SOURCE LABELISES AVEC PREDICTION PAR CNN

Le script Python de visualisation est « 12_Predict_2_Video.py ». Il génère un fichier video « dataset_prediction.avi » dans le dossier « video » du dossier local contenant les datasets.

- 1) Sélectionner le dataset à visualiser et la couche du CNN à visualiser, en modifiant les variables du script.

```
12_Predict_2_Video.py x
1 1 ## PARAMETER #####
2
3 dataset_list = [ "dataset_007" ] # list the dataset directory names to be visualized
4 show_layer = 2+7 # bypass crop and lambda layers (add2)
5
```

- 2) Lancer le script. La visualisation de la couche interne complète la visualisation du dataset.



- 3) Ouvrir la vidéo générée.
- 4)

