

# Suivi des piétons basé sur CNN avec Keras et OpenCv

Rodrigo Viano  
PAO Deep Learning - INSA Rouen

# Plan

- Le contexte
- Les différentes approches
- L'approche utilisée
- Les solutions retenues
- Demo

# Contexte

Construire un modèle de réseaux de neurones capable de détecter et de suivre des piétons

# Différentes approches

- CNN avec sliding windows
- CNN avec OpenCv
- Recurrent Convolutional Neural Network (R-CNN)
- Fast Recurrent Convolutional Neural Network (Fast R-CNN)
- Faster Recurrent Convolutional Neural Network (Faster R-CNN)
- You Only Look Once (YOLO)

# Approche utilisée

## CNN avec OpenCv.

1. Construire et entraîner un CNN capable de classifier des images
2. Traitement d'image avec OpenCv
3. Obtenir les régions d'intérêts avec OpenCv
4. Classifier ces régions d'intérêts avec le CNN
5. Dessiner le bounding box.

# Solutions retenues

3 modèles construits

1. CNN avec des images de taille 150x150, entraîné avec Kitti dataset
2. CNN avec des images de taille 18x36, entraîné avec Daimler dataset
3. CNN avec des images de taille 64x64, entraîné avec Daimler dataset

# CNN 150x150 avec Kitti (1/2)

Image originale en RGB avec des piétons

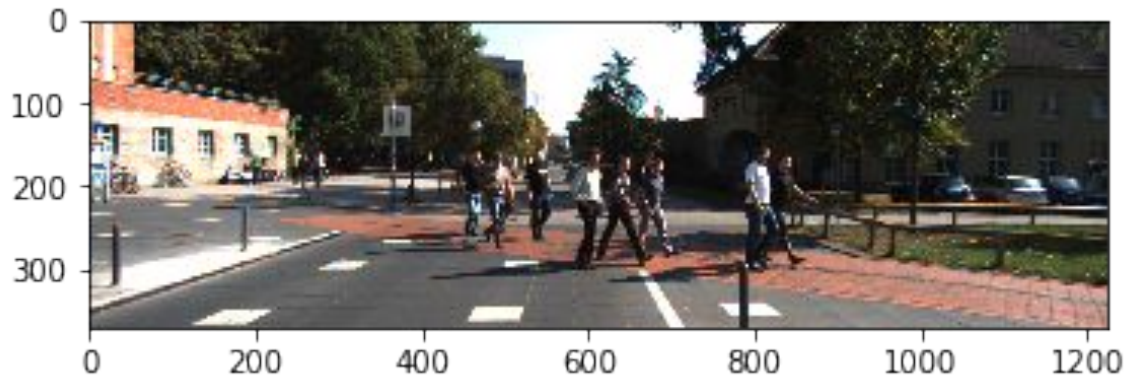
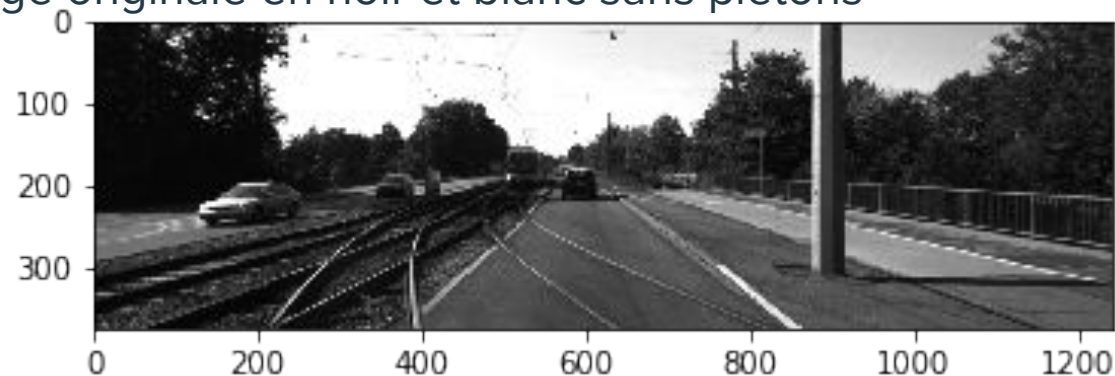


Image originale en noir et blanc sans piétons



# CNN 150x150 avec Kitti (2/2)

## Structure du modèle

File Edit View Bookmarks Settings Help

performance.

Loading weights from kitti/savedweightsvgg-Kitti.h5

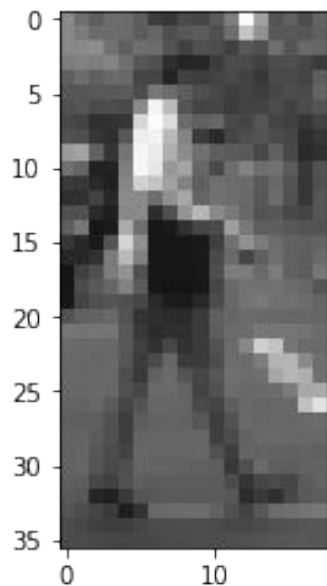
Loaded model and weights from disk

Layer (type)	Output Shape	Param #
=====		
conv2d_1 (Conv2D)	(None, 148, 148, 32)	896
activation_1 (Activation)	(None, 148, 148, 32)	0
max_pooling2d_1 (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_2 (Conv2D)	(None, 72, 72, 32)	9248
activation_2 (Activation)	(None, 72, 72, 32)	0
max_pooling2d_2 (MaxPooling2D)	(None, 36, 36, 32)	0
conv2d_3 (Conv2D)	(None, 34, 34, 64)	18496
activation_3 (Activation)	(None, 34, 34, 64)	0
max_pooling2d_3 (MaxPooling2D)	(None, 17, 17, 64)	0
flatten_1 (Flatten)	(None, 18496)	0
dense_1 (Dense)	(None, 64)	1183808
activation_4 (Activation)	(None, 64)	0
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 1)	65
activation_5 (Activation)	(None, 1)	0
=====		
Total params: 1,212,513		
Trainable params: 1,212,513		
Non-trainable params: 0		

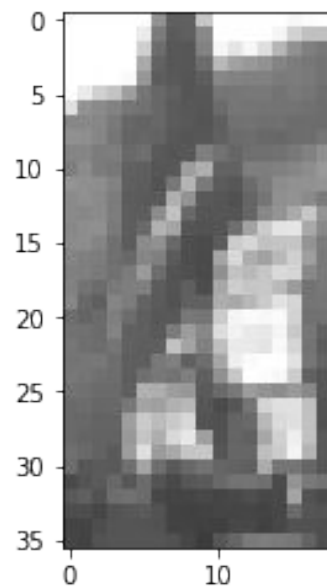


# CNN 18x36 avec Daimler (1/3)

Exemple piéton

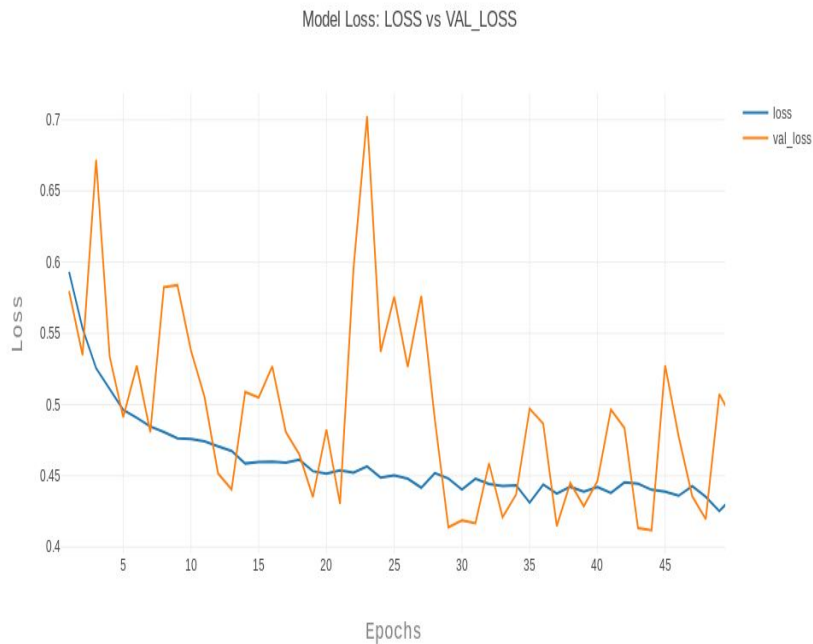


Exemple non-piéton

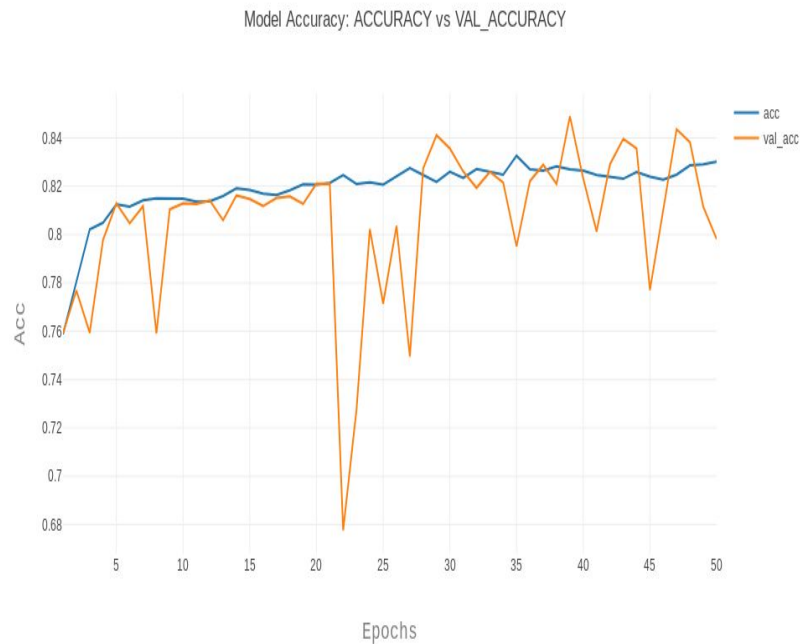


# CNN 18x36 avec Daimler (2/3)

## Perte



## Précision



# CNN 18x36 avec Daimler (3/3)

## Structure du modèle

File Edit View Bookmarks Settings Help

p\_parallelism\_threads for best performance.

Loading model from ped-18x36/modelcnn-18x36.json

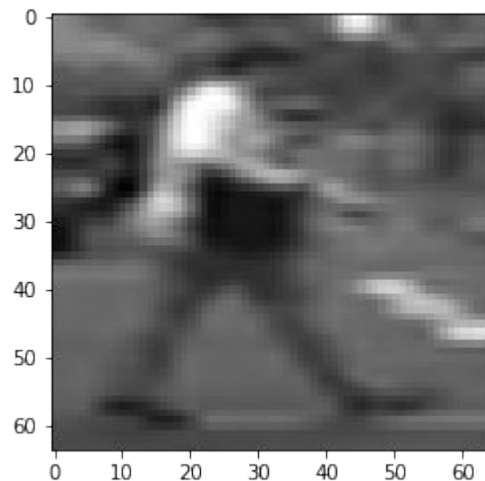
Loading weights from ped-18x36/weights-18x36.h5

Loaded model and weights from disk

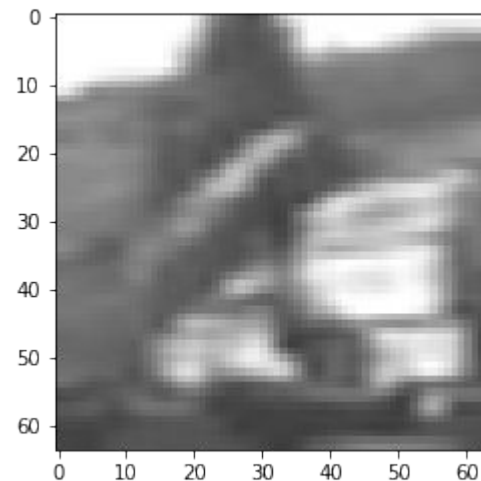
Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 18, 36, 16)	1216
batch_normalization_1 (Batch Normalization)	(None, 18, 36, 16)	64
leaky_re_lu_1 (LeakyReLU)	(None, 18, 36, 16)	0
max_pooling2d_1 (MaxPooling2D)	(None, 9, 18, 16)	0
conv2d_2 (Conv2D)	(None, 9, 18, 32)	12832
batch_normalization_2 (Batch Normalization)	(None, 9, 18, 32)	128
leaky_re_lu_2 (LeakyReLU)	(None, 9, 18, 32)	0
max_pooling2d_2 (MaxPooling2D)	(None, 4, 9, 32)	0
conv2d_3 (Conv2D)	(None, 4, 9, 64)	18496
batch_normalization_3 (Batch Normalization)	(None, 4, 9, 64)	256
leaky_re_lu_3 (LeakyReLU)	(None, 4, 9, 64)	0
max_pooling2d_3 (MaxPooling2D)	(None, 2, 4, 64)	0
flatten_1 (Flatten)	(None, 512)	0
dense_1 (Dense)	(None, 128)	65664
activation_1 (Activation)	(None, 128)	0
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 1)	129
activation_2 (Activation)	(None, 1)	0
Total params: 98,785		
Trainable params: 98,561		
Non-trainable params: 224		

# CNN 64x64 avec Daimler (1/3)

Exemple piéton



Exemple non-piéton



# CNN 64x64 avec Daimler (2/3)

## Précision et pertes

```
ped-64x64 : bash — Konsole
File Edit View Bookmarks Settings Help
245/245 [=====] - 215s 877ms/step - loss: 0.1432 - acc: 0.8026 - val_loss: 0.1275 - val_acc: 0.8526
Epoch 32/50
245/245 [=====] - 220s 899ms/step - loss: 0.1422 - acc: 0.8075 - val_loss: 0.1186 - val_acc: 0.8616
Epoch 33/50
245/245 [=====] - 228s 930ms/step - loss: 0.1384 - acc: 0.8091 - val_loss: 0.1185 - val_acc: 0.8700
Epoch 34/50
245/245 [=====] - 202s 823ms/step - loss: 0.1380 - acc: 0.8066 - val_loss: 0.1139 - val_acc: 0.8744
Epoch 35/50
245/245 [=====] - 235s 961ms/step - loss: 0.1374 - acc: 0.8079 - val_loss: 0.1138 - val_acc: 0.8658
Epoch 36/50
245/245 [=====] - 337s 1s/step - loss: 0.1329 - acc: 0.8172 - val_loss: 0.1114 - val_acc: 0.8601
Epoch 37/50
245/245 [=====] - 334s 1s/step - loss: 0.1296 - acc: 0.8223 - val_loss: 0.1108 - val_acc: 0.8627
Epoch 38/50
245/245 [=====] - 204s 832ms/step - loss: 0.1322 - acc: 0.8198 - val_loss: 0.1059 - val_acc: 0.8857
Epoch 39/50
245/245 [=====] - 195s 798ms/step - loss: 0.1294 - acc: 0.8209 - val_loss: 0.1140 - val_acc: 0.8652
Epoch 40/50
245/245 [=====] - 197s 802ms/step - loss: 0.1280 - acc: 0.8260 - val_loss: 0.1014 - val_acc: 0.8709
Epoch 41/50
245/245 [=====] - 216s 880ms/step - loss: 0.1281 - acc: 0.8221 - val_loss: 0.1017 - val_acc: 0.8711
Epoch 42/50
245/245 [=====] - 211s 862ms/step - loss: 0.1286 - acc: 0.8202 - val_loss: 0.1095 - val_acc: 0.8707
Epoch 43/50
245/245 [=====] - 244s 996ms/step - loss: 0.1263 - acc: 0.8291 - val_loss: 0.1041 - val_acc: 0.8742
Epoch 44/50
245/245 [=====] - 266s 1s/step - loss: 0.1267 - acc: 0.8241 - val_loss: 0.0998 - val_acc: 0.8822
Epoch 45/50
245/245 [=====] - 281s 1s/step - loss: 0.1263 - acc: 0.8226 - val_loss: 0.1015 - val_acc: 0.8796
Epoch 46/50
245/245 [=====] - 274s 1s/step - loss: 0.1263 - acc: 0.8241 - val_loss: 0.0969 - val_acc: 0.8878
Epoch 47/50
245/245 [=====] - 295s 1s/step - loss: 0.1259 - acc: 0.8245 - val_loss: 0.0988 - val_acc: 0.8856
Epoch 48/50
245/245 [=====] - 271s 1s/step - loss: 0.1247 - acc: 0.8259 - val_loss: 0.0963 - val_acc: 0.8848
Epoch 49/50
245/245 [=====] - 273s 1s/step - loss: 0.1216 - acc: 0.8362 - val_loss: 0.1070 - val_acc: 0.8771
Epoch 50/50
245/245 [=====] - 266s 1s/step - loss: 0.1207 - acc: 0.8356 - val_loss: 0.0966 - val_acc: 0.8969
ped-64x64 : bash
```

# CNN 64x64 avec Daimler (3/3)

## Structure du modèle

```
Loading weights from ped-64x64/weights-64x64.h5
Loaded model and weights from disk
```

Layer (type)	Output Shape	Param #
lambda_1_input (InputLayer)	(None, 64, 64, 3)	0
lambda_1 (Lambda)	(None, 64, 64, 3)	0
cv0 (Conv2D)	(None, 64, 64, 16)	448
dropout_1 (Dropout)	(None, 64, 64, 16)	0
cv1 (Conv2D)	(None, 64, 64, 32)	4640
dropout_2 (Dropout)	(None, 64, 64, 32)	0
cv2 (Conv2D)	(None, 64, 64, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 64)	0
dropout_3 (Dropout)	(None, 8, 8, 64)	0
fcn (Conv2D)	(None, 1, 1, 1)	4097
flatten_1 (Flatten)	(None, 1)	0
Total params: 27,681		
Trainable params: 27,681		
Non-trainable params: 0		

```
rodrigo@rodrigo-wesh:/media/rodrigo/Rodrigo/GIT/PAOS
```

# Demo

Merci pour votre attention, questions ?





# Sources

- <http://cs231n.stanford.edu/>
- <https://keras.io/>
- [https://www.researchgate.net/publication/4171908\\_Pedestrian\\_detection\\_with\\_convolutional\\_neural\\_networks](https://www.researchgate.net/publication/4171908_Pedestrian_detection_with_convolutional_neural_networks)