# 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

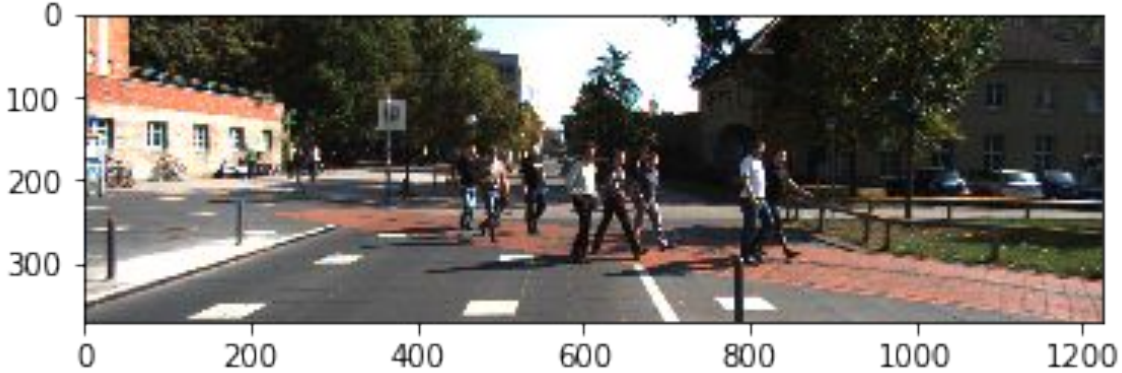
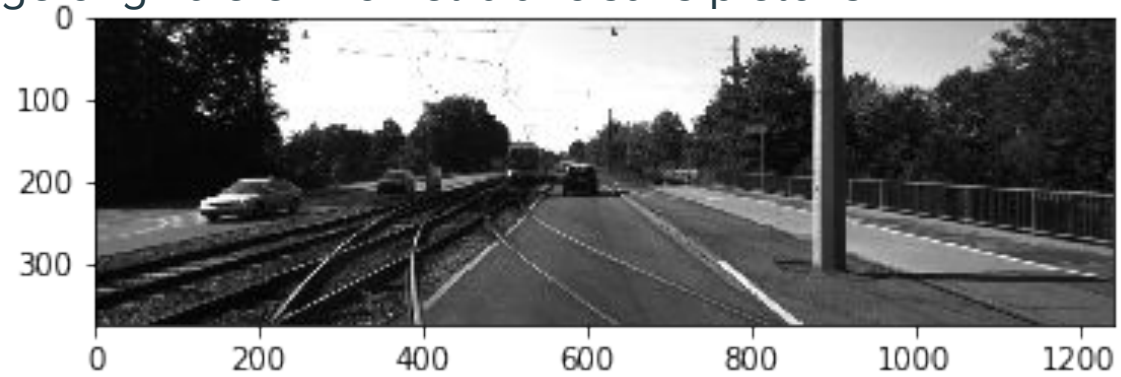
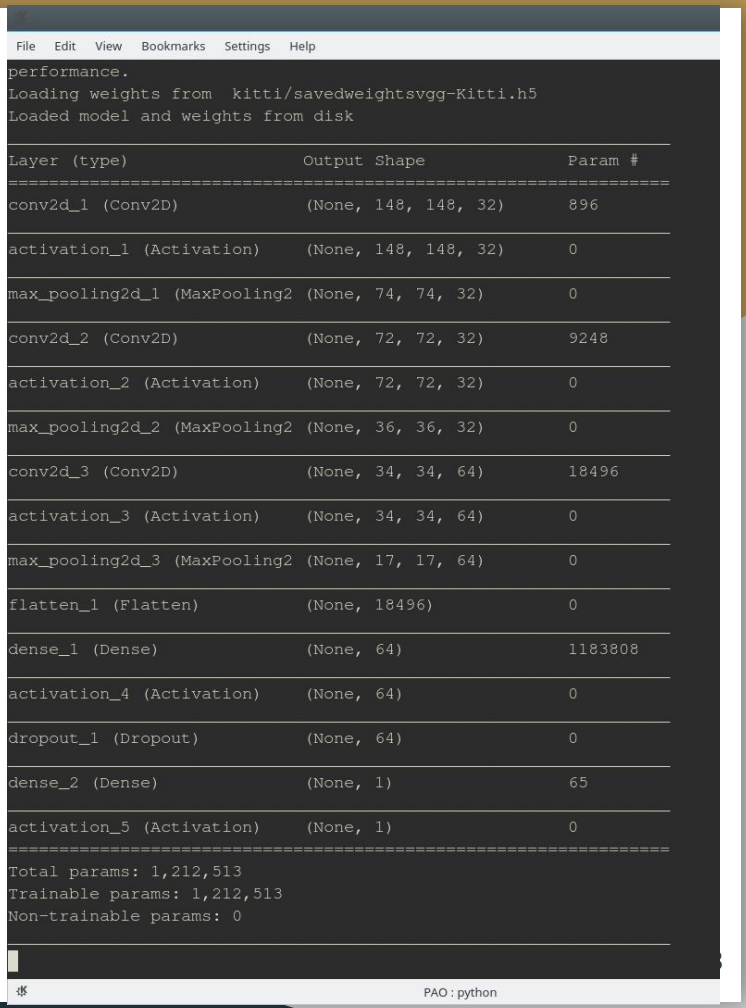



Image originale en noir et blanc sans piétons



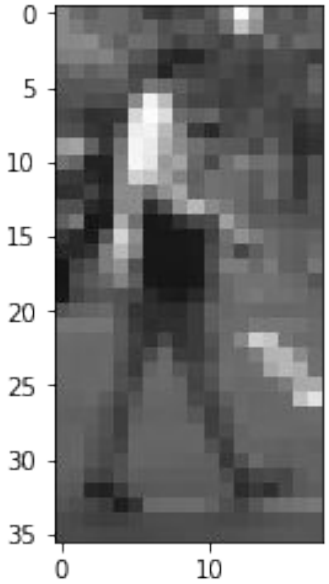
# CNN 150x150 avec Kitti (2/2)

Structure du modèle

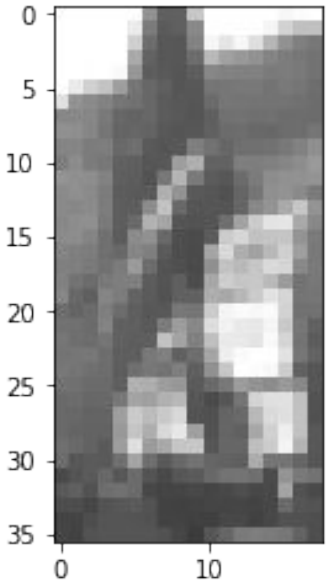


# CNN 18x36 avec Daimler (1/3)

## Example piéton

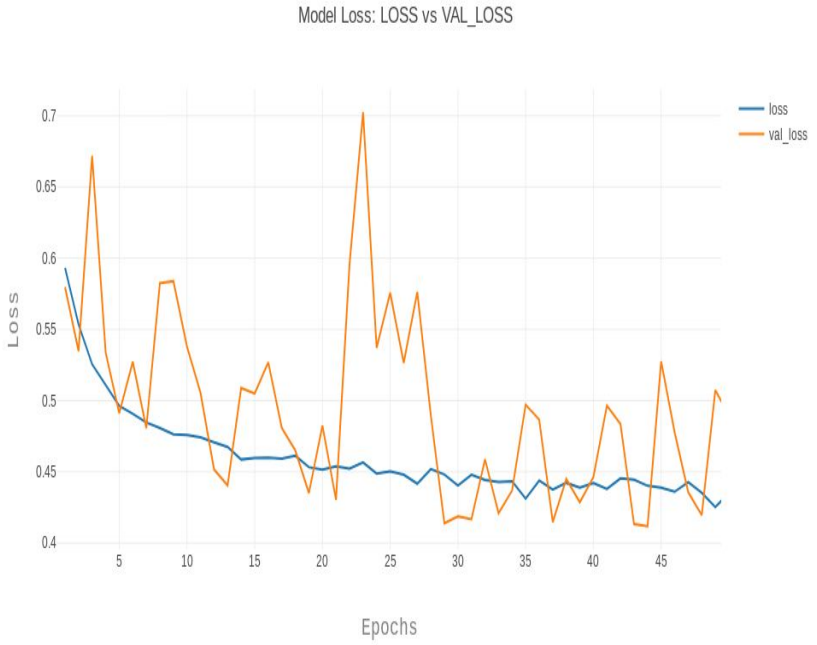



## Example non-piéton

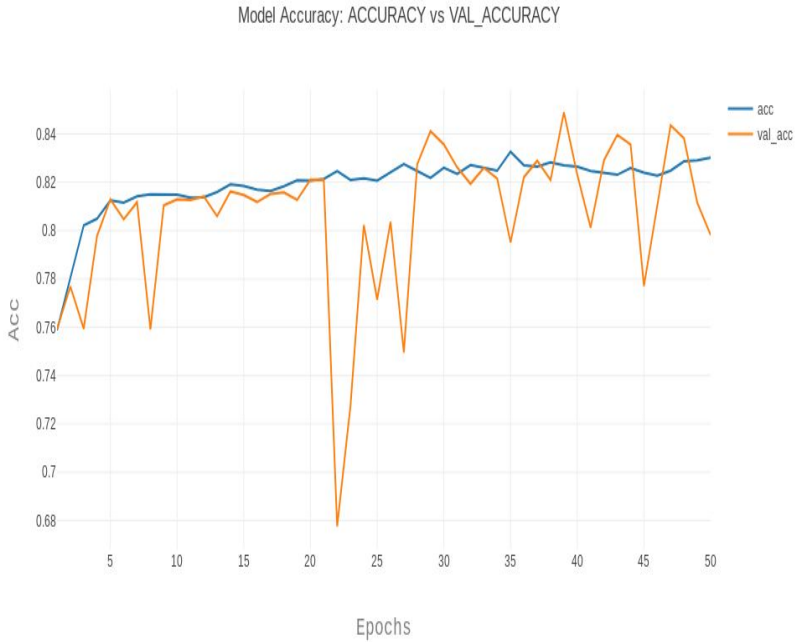


# CNN 18x36 avec Daimler (2/3)

## Perte

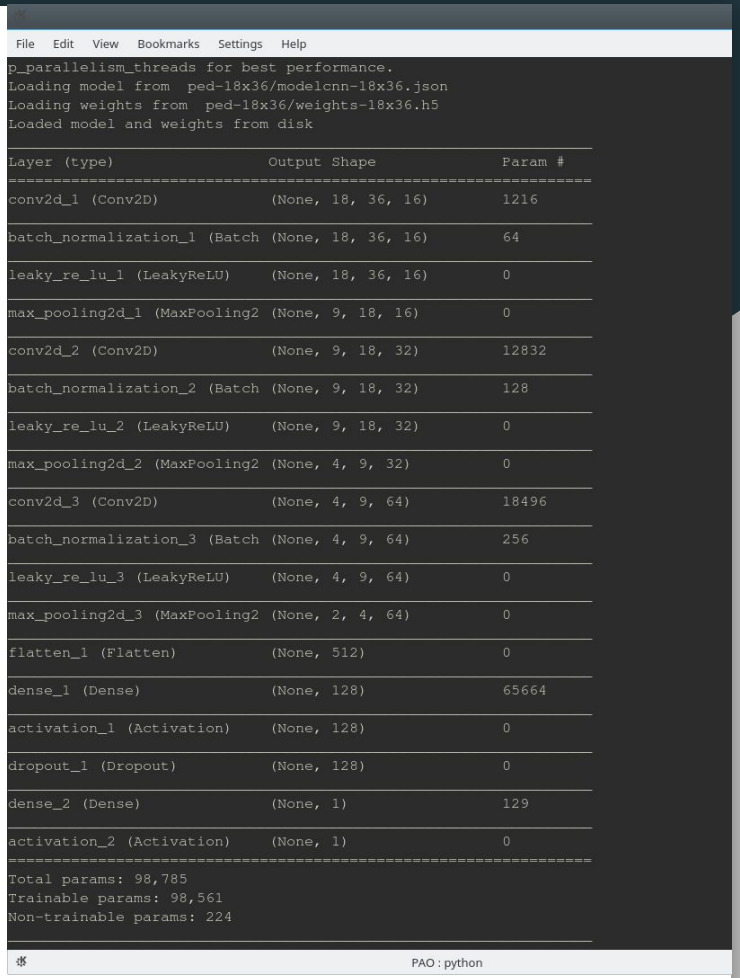



## Précision



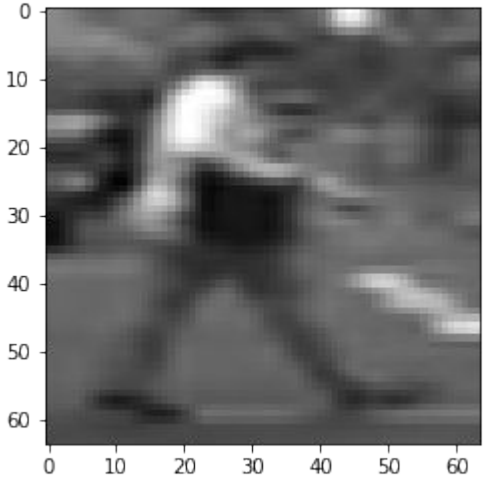
# CNN 18x36 avec Daimler (3/3)

Structure du modèle

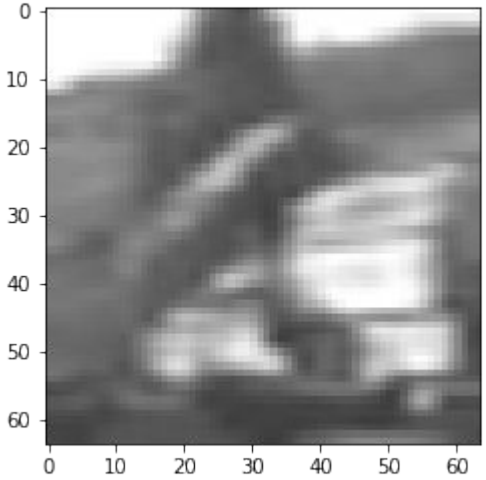


# CNN 64x64 avec Daimler (1/3)

## Example piéton

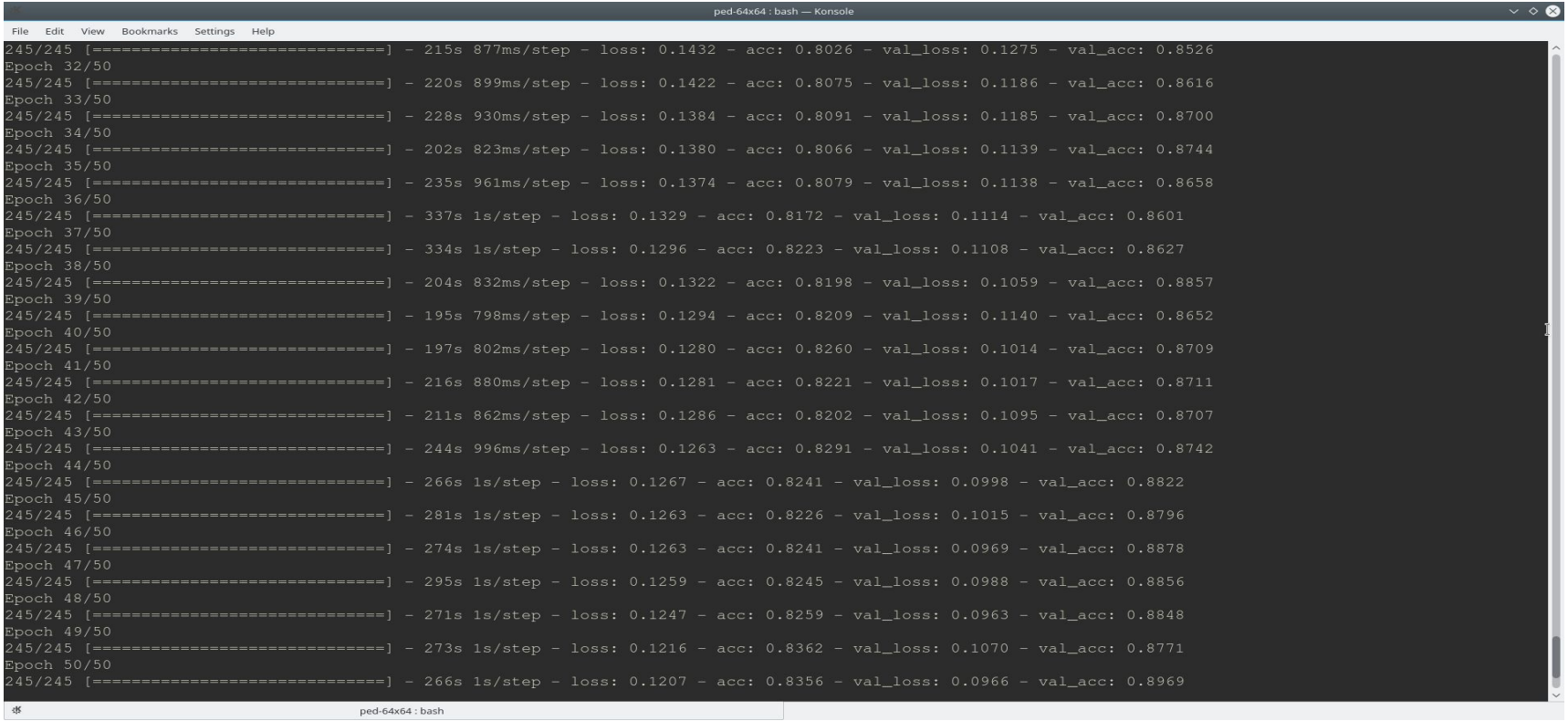



## Example non-piéton



# CNN 64x64 avec Daimler (2/3)

## Précision et pertes



# 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 (MaxPooling2 (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/PAO$
```

PAO : bash

# 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