

```
from google.colab import drive
!ls
drive.mount('/content/drive')

sample_data
Mounted at /content/drive

# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras

from keras.applications.vgg16 import VGG16
import numpy as np
import matplotlib.pyplot as plt

train_dict_path = '/content/drive/Shareddrives/Projet-ML-M2-PLS/data/positions/train'
valid_dict_path = '/content/drive/Shareddrives/Projet-ML-M2-PLS/data/positions/validation'

# On a utilisé les layer preprocessing VGG16 de Keras
# avec demension de 640*480 de l'image et 3 pour RGB
vgg = VGG16(input_shape=[640, 480] + [3], weights='imagenet', include_top=False)

for layer in vgg.layers:
    layer.trainable = False

x = keras.layers.Flatten()(vgg.output)

# on definit 6 classe du dernier layer (face, gauche, droite, haut, bas, absent)
prediction = keras.layers.Dense(6, activation='softmax')(x)

# creation du model
model = keras.Model(inputs=vgg.input, outputs=prediction)

# On compile avec l'optimizer Adam
model.compile(
    optimizer=keras.optimizers.Adam(),
    loss=[keras.losses.CategoricalCrossentropy()],
    metrics=['accuracy']
)

# afficher la structure du modele
model.summary()

# On utilise la classe ImageDataGenerator pour traiter les image
from keras.preprocessing.image import ImageDataGenerator

train_img_datagen = ImageDataGenerator(rescale = 1./255,
                                       shear_range = 0.2,
                                       zoom_range = 0.2,
                                       horizontal_flip = True)

test_img_datagen = ImageDataGenerator(rescale = 1./255)
```

```
# on utilise la fonction (flow_from_directory) pour definir le training set avec ces classes
training_set = train_img_datagen.flow_from_directory(train_dict_path,
                                                    target_size = (640, 480),
                                                    batch_size = 32,
                                                    class_mode = 'categorical')

test_set = test_img_datagen.flow_from_directory(valid_dict_path,
                                                target_size = (640, 480),
                                                batch_size = 32,
                                                class_mode = 'categorical')

# fit le model avec epochs=5 comme un debut
history = model.fit(
    training_set,
    validation_data=test_set,
    epochs=5,
    steps_per_epoch=len(training_set),
    validation_steps=len(test_set)
)
```

=====		
input_2 (InputLayer)	[(None, 640, 480, 3)]	0
block1_conv1 (Conv2D)	(None, 640, 480, 64)	1792
block1_conv2 (Conv2D)	(None, 640, 480, 64)	36928
block1_pool (MaxPooling2D)	(None, 320, 240, 64)	0
block2_conv1 (Conv2D)	(None, 320, 240, 128)	73856
block2_conv2 (Conv2D)	(None, 320, 240, 128)	147584
block2_pool (MaxPooling2D)	(None, 160, 120, 128)	0
block3_conv1 (Conv2D)	(None, 160, 120, 256)	295168
block3_conv2 (Conv2D)	(None, 160, 120, 256)	590080
block3_conv3 (Conv2D)	(None, 160, 120, 256)	590080
block3_pool (MaxPooling2D)	(None, 80, 60, 256)	0
block4_conv1 (Conv2D)	(None, 80, 60, 512)	1180160
block4_conv2 (Conv2D)	(None, 80, 60, 512)	2359808
block4_conv3 (Conv2D)	(None, 80, 60, 512)	2359808
block4_pool (MaxPooling2D)	(None, 40, 30, 512)	0
block5_conv1 (Conv2D)	(None, 40, 30, 512)	2359808
block5_conv2 (Conv2D)	(None, 40, 30, 512)	2359808
block5_conv3 (Conv2D)	(None, 40, 30, 512)	2359808
block5_pool (MaxPooling2D)	(None, 20, 15, 512)	0

flatten (Flatten)	(None, 153600)	0
dense (Dense)	(None, 6)	921606
=====		
Total params: 15,636,294		
Trainable params: 921,606		
Non-trainable params: 14,714,688		

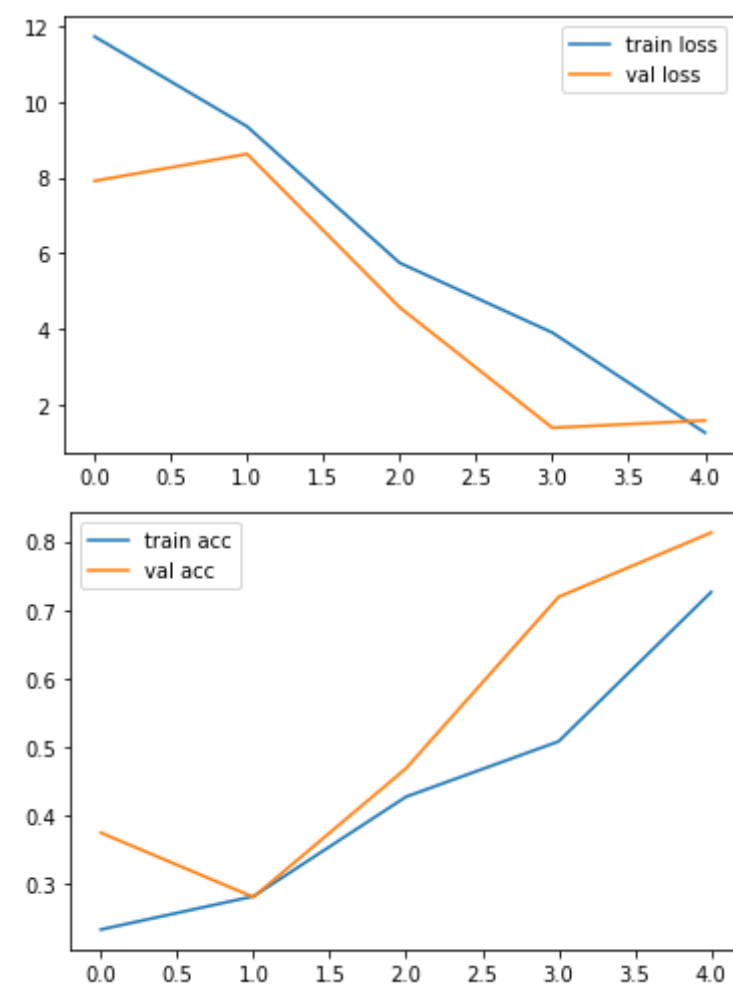
Found 124 images belonging to 6 classes.
Found 32 images belonging to 6 classes.
Epoch 1/5
4/4 [=====] - 489s 129s/step - loss: 9.1340 - accuracy: 0.2008 - val_loss: 7.9118 - val_accuracy: 0.3750
Epoch 2/5
4/4 [=====] - 500s 128s/step - loss: 9.5011 - accuracy: 0.3031 - val_loss: 8.6218 - val_accuracy: 0.2812
Epoch 3/5
4/4 [=====] - 505s 134s/step - loss: 6.7225 - accuracy: 0.3961 - val_loss: 4.5744 - val_accuracy: 0.4688
Epoch 4/5
4/4 [=====] - 498s 132s/step - loss: 4.3296 - accuracy: 0.4918 - val_loss: 1.3998 - val_accuracy: 0.7188
Epoch 5/5
4/4 [=====] - 506s 137s/step - loss: 1.3638 - accuracy: 0.7134 - val_loss: 1.5920 - val_accuracy: 0.8125

```
# afficher les different valeur du loss
plt.plot(history.history['loss'], label='train loss')
plt.plot(history.history['val_loss'], label='val loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')
```

```
# afficher les different valeur du l'accuracy
plt.plot(history.history['accuracy'], label='train acc')
plt.plot(history.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
plt.savefig('AccVal_acc')
```

```
model.save('positions_model.h5')
```





<Figure size 432x288 with 0 Axes>