

Développement d'un système de reconnaissance des expressions faciales

Réalisé par : ALLAL Yahia , AIT ICHOU Mustapha , AMMOURI Yassire

Faculté des Sciences de Rabat

16 Juillet 2024



Plan

- 1 Problématique
- 2 La base de données FER-2013
- 3 Les outils utilisés & L'environnement
- 4 Architecture de CNN et Resultat
- 5 Démonstration
- 6 Conclusion

Problématique

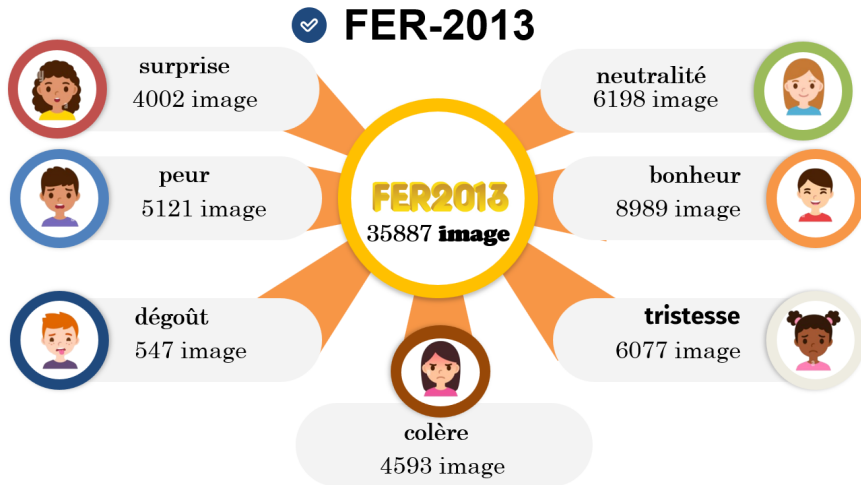
Problématique



- 1 La reconnaissance des expressions faciales est essentielle dans différents domaines, notamment les interactions homme-machine, la surveillance et la médecine.
- 2 Certaines expressions ne sont pas détectées par l'humain de façon optimale, ce qui présente des limitations en termes de précision et d'efficacité.
- 3 Comment nous pouvons améliorer la détection des expressions faciales à l'aide de techniques avancées comme les réseaux de neurones convolutifs ?

La base de données FER-2013

La base de données FER-2013



La base de données FER-2013

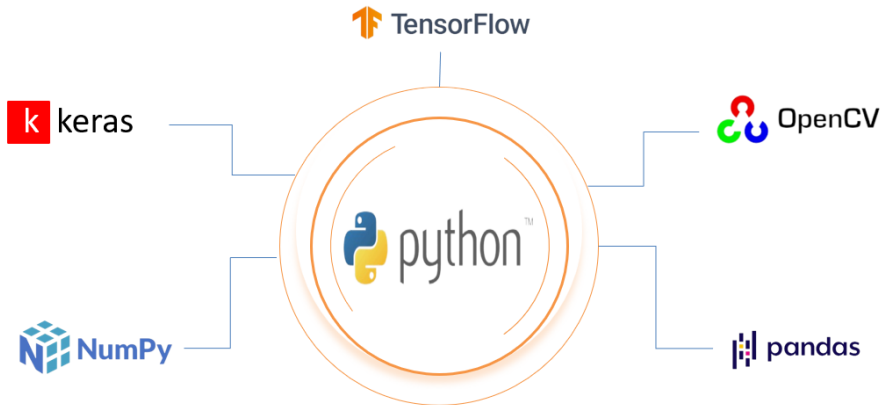
Exemples d'images du dataset



Les outils utilisés & L'environnement

Les outils utilisés & L'environnement

✔ les logiciels & les bibliothèques informatiques



✓ le matériel



*processeur :
intel core i7 8^{ème}*

*carte graphique :
intel (R) Graphics 650*

*mémoire vive : **16 Go***

*système d'exploitation:
Windows 11*

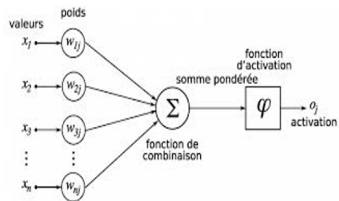
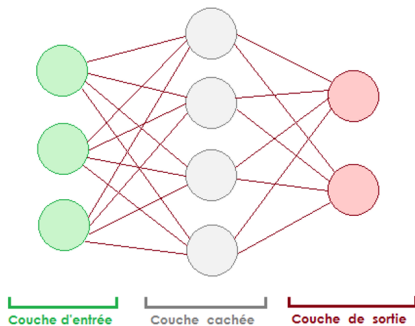
Pré-traitement

```
train_datagen = ImageDataGenerator(  
    rescale=1./255,  
    rotation_range=30,  
    shear_range=0.3,  
    zoom_range=0.3,  
    horizontal_flip=True,  
    fill_mode='nearest')  
  
validation_datagen = ImageDataGenerator(rescale=1./255)  
  
train_generator = train_datagen.flow_from_directory(  
    train_data_dir,  
    color_mode='grayscale',  
    target_size=(48, 48),  
    batch_size=32,  
    class_mode='categorical',  
    shuffle=True)  
  
validation_generator = validation_datagen.flow_from_directory(  
    validation_data_dir,  
    color_mode='grayscale',  
    target_size=(48, 48),  
    batch_size=32,  
    class_mode='categorical',  
    shuffle=True)
```

Architecture de CNN et Resulat

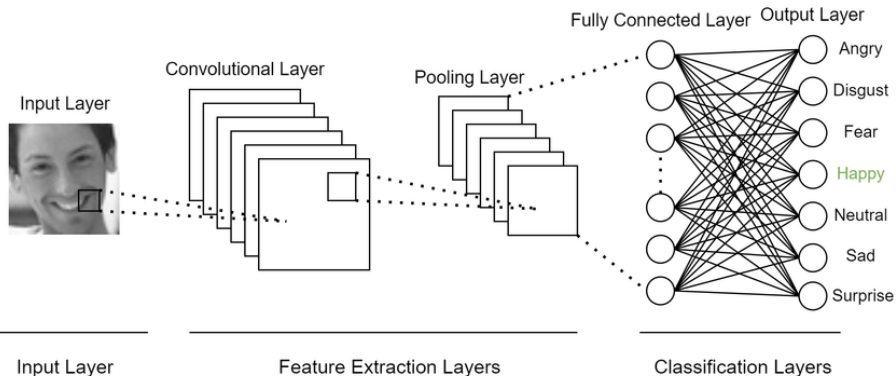
Architecture de ANN

✓ réseau de neurone



Neurone artificiel

Architecture de CNN



Script du model

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(48,48,1)))

model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.1))

model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.1))

model.add(Conv2D(256, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Dropout(0.1))

model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(7, activation='softmax'))
model.compile(optimizer = 'adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

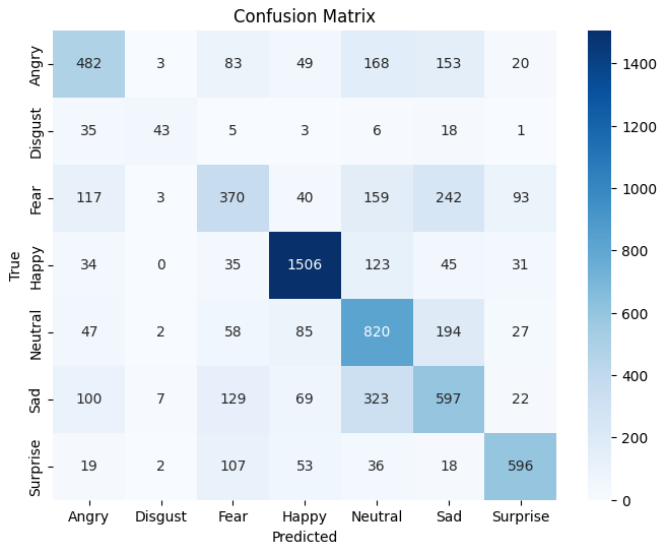
Model Summary

```

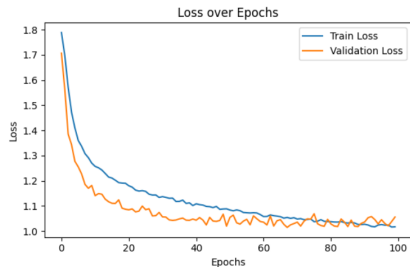
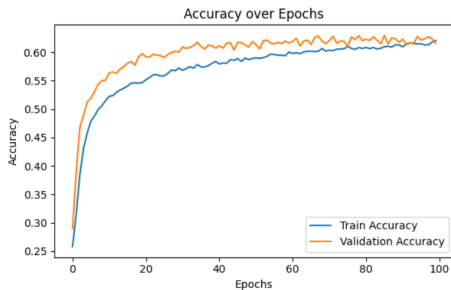
Layer (type)                Output Shape                Param #
=====
conv2d_8 (Conv2D)           (None, 46, 46, 32)         320
conv2d_9 (Conv2D)           (None, 44, 44, 64)         18496
max_pooling2d_6 (MaxPoolin  (None, 22, 22, 64)         0
g2D)
dropout_8 (Dropout)         (None, 22, 22, 64)         0
conv2d_10 (Conv2D)          (None, 20, 20, 128)        73856
max_pooling2d_7 (MaxPoolin  (None, 10, 10, 128)        0
g2D)
dropout_9 (Dropout)         (None, 10, 10, 128)        0
conv2d_11 (Conv2D)          (None, 8, 8, 256)          295168
max_pooling2d_8 (MaxPoolin  (None, 4, 4, 256)          0
...)
Epoch 99/100
897/897 [=====] - 34s 38ms/step - loss: 1.0163 - accuracy: 0.6176 - val_loss: 1.0378 - val_accuracy: 0.6232
Epoch 100/100
897/897 [=====] - 37s 41ms/step - loss: 1.0172 - accuracy: 0.6203 - val_loss: 1.0560 - val_accuracy: 0.6152

```


Confusion matrix



The metrics results



Transfer Learning

```
from keras.applications import VGG16

# Load the VGG16 model with pretrained weights
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(48, 48, 3))

# Freeze the layers of the base model
for layer in base_model.layers:
    layer.trainable = False

# Build the model
model = Sequential()
model.add(base_model)
```

```
Epoch 97/100
897/897 [=====] - 33s 37ms/step - loss: 1.4383 - accuracy: 0.4381 - val_loss: 1.4726 - val_accuracy: 0.4413
Epoch 98/100
897/897 [=====] - 33s 37ms/step - loss: 1.4453 - accuracy: 0.4424 - val_loss: 1.4916 - val_accuracy: 0.4321
Epoch 99/100
897/897 [=====] - 34s 38ms/step - loss: 1.4399 - accuracy: 0.4412 - val_loss: 1.4886 - val_accuracy: 0.4329
Epoch 100/100
897/897 [=====] - 33s 36ms/step - loss: 1.4340 - accuracy: 0.4442 - val_loss: 1.4670 - val_accuracy: 0.4503
```

Démonstration

Conslusion

Conclusion

- ❶ Bien que nous augmentions le nombre d'époques, l'accuracy reste insuffisante.
- ❷ Mais malgré cela, notre modèle peut estimer un grand nombre d'expressions faciales
- ❸ Complexité des Émotions.
- ❹ Performance en Temps Réel.
- ❺ Variabilité des Conditions d'Éclairage et d'Angles de Vue.

Merci pour votre attention !

