Non ci siamo concentrati sull'aspetto di come funziona l'algoritmo ma abbiamo voluto portare una cosa diversa dal solito, usandolo e cercando di trarre delle conclusioni interessanti dal suo utilizzo. Più importante per noi è dimostrare che il dolore non è come le altre espressioni ma ha una componente psicologica molto piú evidente. Si noterá infatti che le reazioni cambiano molto in base alla cultura, all'esperienza ed al soggetto che lo sta provando

Software
1- clonazione della repository da github.

```
[ ]  !git clone https://github.com/misbah4064/facial_expressions.git

     Cloning into 'facial_expressions'...
     remote: Enumerating objects: 29, done.
     remote: Counting objects: 100% (29/29), done.
     remote: Compressing objects: 100% (29/29), done.
     remote: Total 14243 (delta 12), reused 0 (delta 0), pack-reused 14214
     Receiving objects: 100% (14243/14243), 240.06 MiB | 34.46 MiB/s, done.
     Resolving deltas: 100% (235/235), done.
     Checking out files: 100% (14004/14004), done.
```

2- accesso alla cartella di lavoro e creazione delle cartelle di supporto all'algoritmo.

```
%cd facial_expressions/
%mkdir -p data_set/{anger,happy,neutral,sad}

/content/facial_expressions
```

3-smistamento delle immagini nel database principale nelle cartelle create nel passaggio precedente.

```
[ ]  import cv2
     with open('happy.txt','r') as f:
         img = [line.strip() for line in f]
     for image in img:
         loadedImage = cv2.imread("images/"+image)
         cv2.imwrite("data_set/happy/"+image,loadedImage)
     print("done writing")

     done writing
```

4-individuazione volti nelle foto smistate e salvataggio volti con identificativo assegnato. Salvataggio nella cartella dataset.

```
[ ]  import cv2

     with open('happy.txt','r') as f:
         images = [line.strip() for line in f]

     face_detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

     # For each Emotion, enter one numeric face id
     face_id = input('\n Enter Emotion id end press <return> ==>  ')

     count = 0

     for image in images:
         img = cv2.imread("data_set/happy/"+image)
         gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
         faces = face_detector.detectMultiScale(gray, 1.3, 5)

         for (x,y,w,h) in faces:

             cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2)
             count += 1

             # Save the captured image into the datasets folder
             cv2.imwrite("dataset/User." + str(face_id) + '.' + str(count) + ".jpg", gray[y:y+h,x:x+w])

     print("\n Done creating face data")
```

```
 Enter user id end press <return> ==>  1

 Done creating face data
```

5 - Fase di training. Avviare questa fase solo dopo che tutte le espressioni sono state preparate nelle fasi precedenti 3 e 4. Il modello verrà salvato nella cartella “trainer”. Se tutto andrà a buon fine si vedrà in output la frase ”exiting program”

```python
import cv2
import numpy as np
from PIL import Image
import os

# Path for face image database
path = 'dataset'

recognizer = cv2.face.LBPHFaceRecognizer_create()
detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml");

# function to get the images and label data
def getImagesAndLabels(path):

    imagePaths = [os.path.join(path,f) for f in os.listdir(path)]
    faceSamples=[]
    ids = []

    for imagePath in imagePaths:

        PIL_img = Image.open(imagePath).convert('L') # convert it to grayscale
        img_numpy = np.array(PIL_img,'uint8')

        id = int(os.path.split(imagePath)[-1].split(".")[1])
        faces = detector.detectMultiScale(img_numpy)

        for (x,y,w,h) in faces:
            faceSamples.append(img_numpy[y:y+h,x:x+w])
            ids.append(id)

    return faceSamples,ids

print ("\n [INFO] Training faces....")
faces,ids = getImagesAndLabels(path)
recognizer.train(faces, np.array(ids))

# Save the model into trainer/trainer.yml
recognizer.write('trainer/trainer.yml')

# Print the numer of Emotions trained and end program
print("\n [INFO] {0} Emotions trained. Exiting Program".format(len(np.unique(ids))))
```

```
 [INFO] Training faces....

 [INFO] 2 faces trained. Exiting Program
```

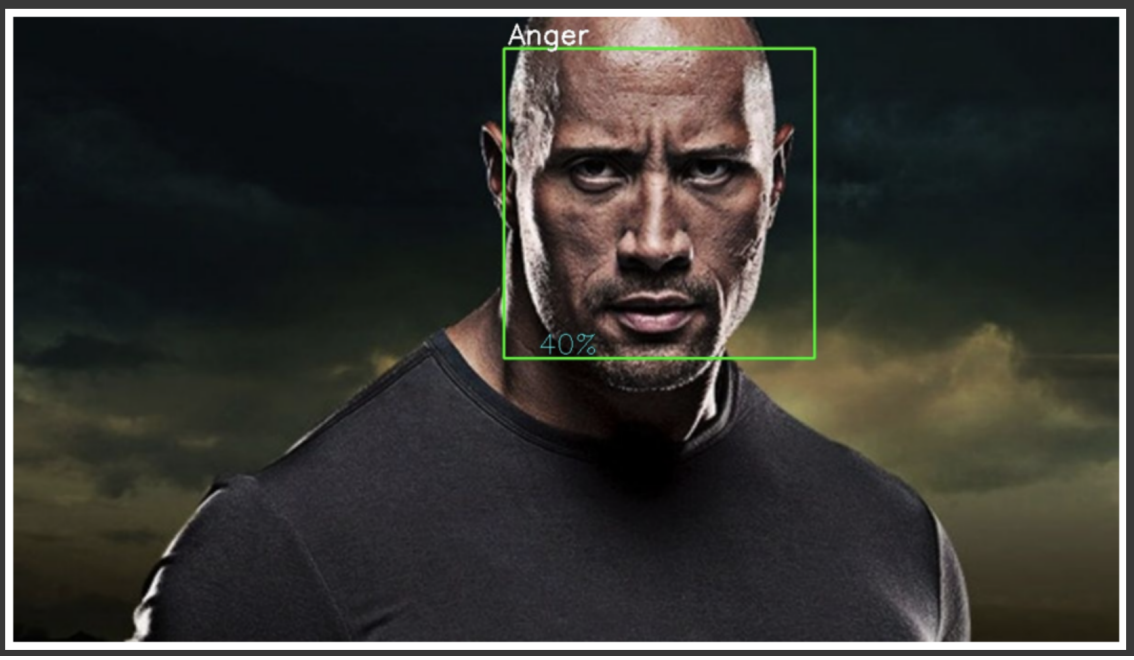
6- inserimento immagine per testare le capacitá di riconoscimento dell’algoritmo.

```python
import cv2
import numpy as np
import os

recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read('trainer/trainer.yml')
cascadePath = "haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascadePath);

font = cv2.FONT_HERSHEY_SIMPLEX

#iniciate id counter
id = 0

# Emotions related to ids: example ==> Anger: id=0,  etc
names = ['Anger', 'Happy', 'None', 'None', 'None', 'None']

# Initialize and start realtime video capture
cam = cv2.VideoCapture(0)
cam.set(3, 640) # set video widht
cam.set(4, 480) # set video height

# Define min window size to be recognized as a face
minW = 0.1*cam.get(3)
minH = 0.1*cam.get(4)

# ret, img =cam.read()
img = cv2.imread("dwayne.jpg")
# img = cv2.flip(img, -1) # Flip vertically

gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

faces = faceCascade.detectMultiScale(
    gray,
    scaleFactor = 1.2,
    minNeighbors = 5,
    minSize = (int(minW), int(minH)),
    )

for(x,y,w,h) in faces:

    cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)

    id, confidence = recognizer.predict(gray[y:y+h,x:x+w])

    # Check if confidence is less them 100 ==> "0" is perfect match
    if (confidence < 100):
        id = names[id]
        confidence = "  {0}%".format(round(100 - confidence))
    else:
        id = "unknown"
        confidence = "  {0}%".format(round(100 - confidence))

    cv2.putText(img, str(id), (x+5,y-5), font, 1, (255,255,255), 2)
    cv2.putText(img, str(confidence), (x+5,y+h-5), font, 1, (255,255,0), 1)

cv2.imwrite("dwayne_johnson.jpg",img)

print("\n [INFO] Done detecting and Image is saved")
cam.release()
cv2.destroyAllWindows()
```

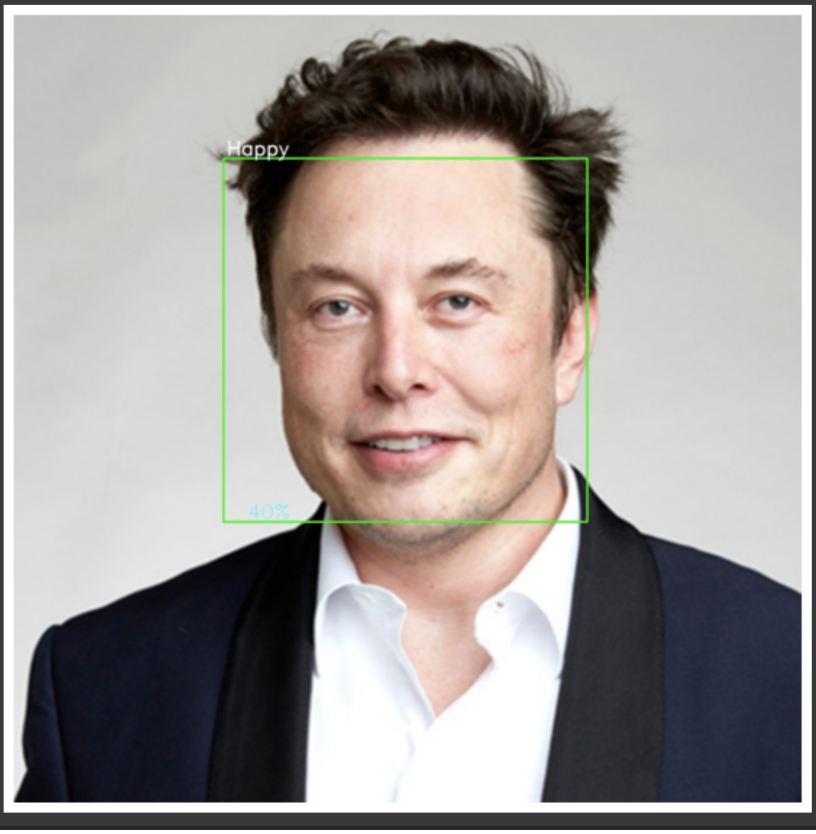
```
 [INFO] Done detecting and Image is saved
```

7- porzione di codice necessaria per mostrare il risultato ingrandito.

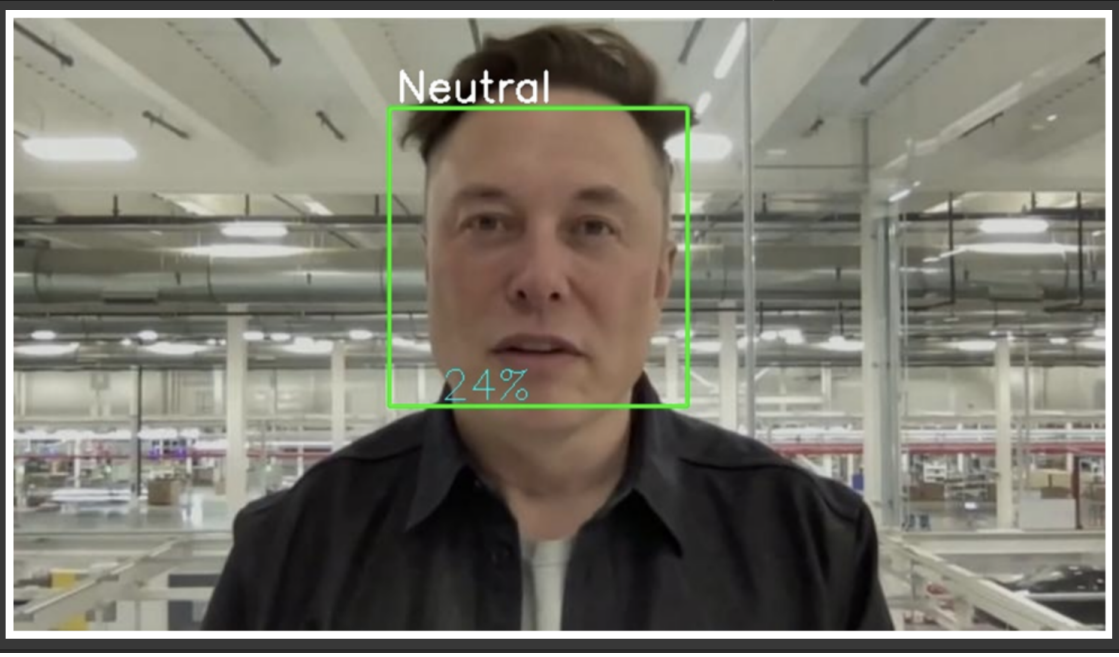
```python
import cv2
import matplotlib.pyplot as plt
%matplotlib inline

image = cv2.imread("dwayne_johnson.jpg")
height, width = image.shape[:2]
resized_image = cv2.resize(image,(3*width, 3*height), interpolation = cv2.INTER_CUBIC)

fig = plt.gcf()
fig.set_size_inches(18, 10)
plt.axis("off")
plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
plt.show()
```



```
import cv2
import matplotlib.pyplot as plt
%matplotlib inline

image = cv2.imread("elon_musk.jpg")
height, width = image.shape[:2]
resized_image = cv2.resize(image,(3*width, 3*height), interpolation = cv2.INTER_CUBIC)

fig = plt.gcf()
fig.set_size_inches(18, 10)
plt.axis("off")
plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
plt.show()
```



```
import cv2
import matplotlib.pyplot as plt
%matplotlib inline

image = cv2.imread("neutral_result.jpg")
height, width = image.shape[:2]
resized_image = cv2.resize(image,(3*width, 3*height), interpolation = cv2.INTER_CUBIC)

fig = plt.gcf()
fig.set_size_inches(18, 10)
plt.axis("off")
plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
plt.show()
```



La situazione si diventa complicata ed interessante quando si prova ad insegnare all'algoritmo un'espressione per la quale non è stato testato: l'espressione del dolore.

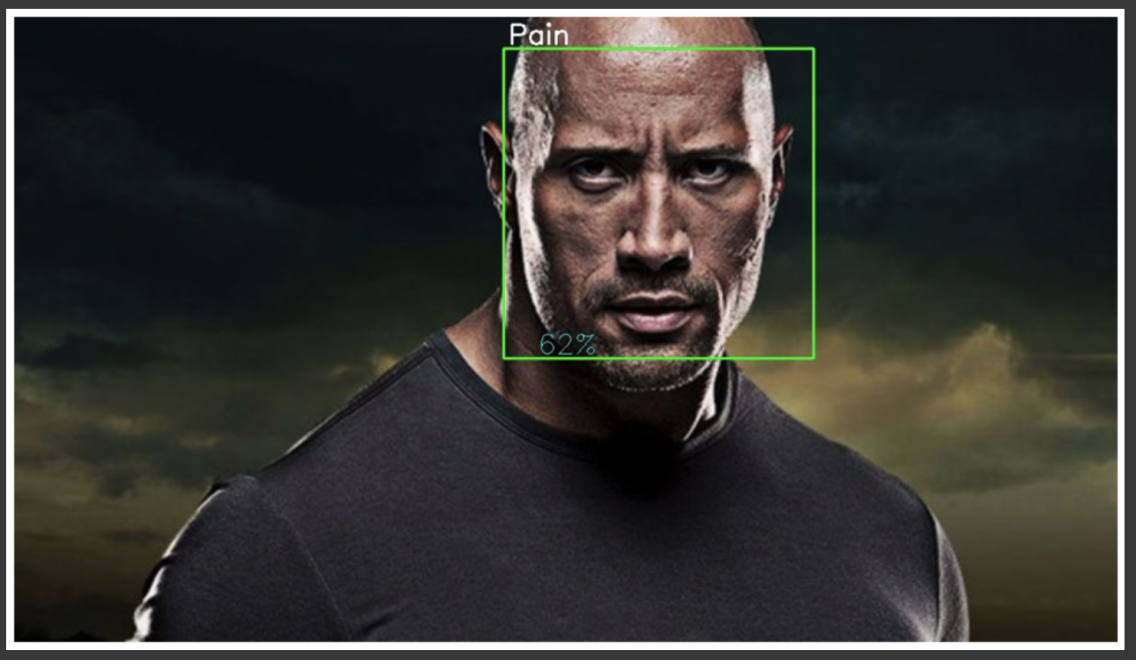
Database

Per insegnare una nuova espressione è necessario innanzitutto trovare un buon database. Dopo aver trovato le immagini delle espressioni che si vogliono insegnare, si procede creando un piccolo programma in java che estrae tutti i nomi delle immagini contenute nel database. Il risultato servirà per creare il file di testo che serve per smistare il database principale nelle varie cartelle delle espressioni. Si importa creando un file pain.txt e vi si inseriscono tutti i nomi delle immagini del database. Si aggiungono, poi, le immagini stesse al database interno al codice.

Test Riconoscimento Dolore

Provando ad avviare le immagini usando lo stesso algoritmo prima e dopo il riconoscimento del dolore, si può notare che questo comincerà a confondersi e qualsiasi espressione gli si presenti, verrà riconosciuta come espressione di dolore.

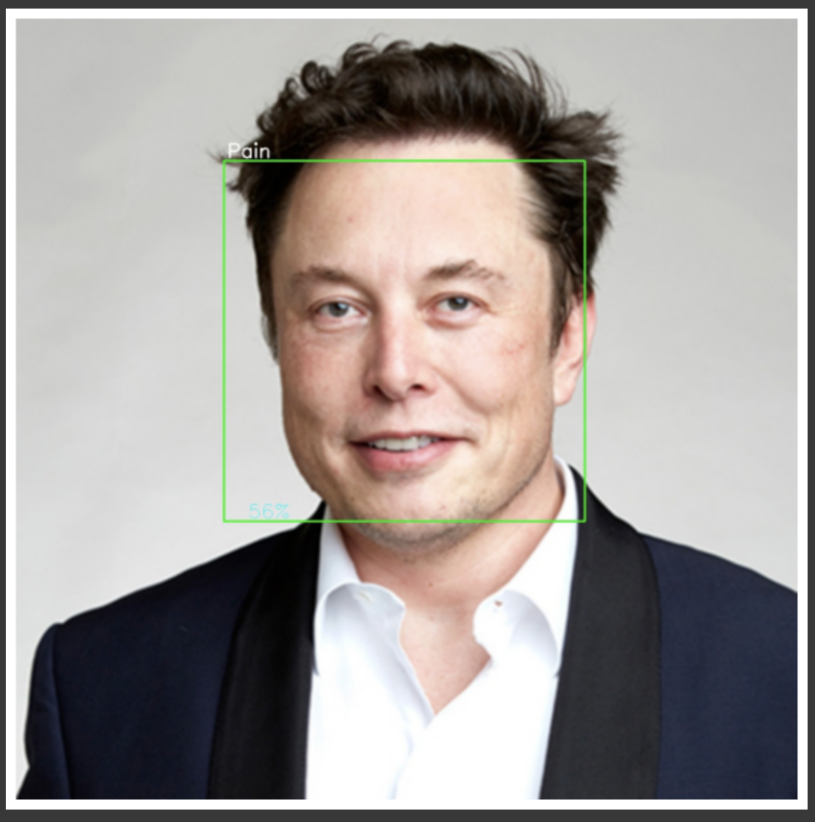
```
import cv2
import matplotlib.pyplot as plt
%matplotlib inline

image = cv2.imread("dwayne_johnson.jpg")
height, width = image.shape[:2]
resized_image = cv2.resize(image,(3*width, 3*height), interpolation = cv2.INTER_CUBIC)

fig = plt.gcf()
fig.set_size_inches(18, 10)
plt.axis("off")
plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
plt.show()
```



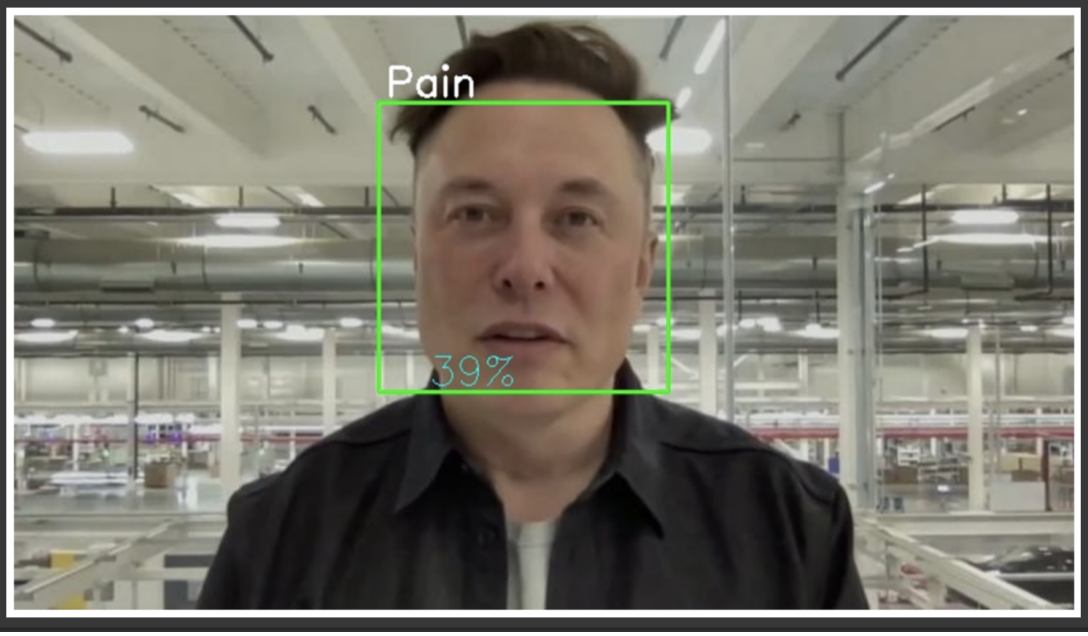
```
import cv2
import matplotlib.pyplot as plt
%matplotlib inline

on_musk.jpg")
shape[:2]
ize(image,(3*width, 3*height), interpolation = cv2.INTER_CUBIC)

fig.set_size_inches(18, 10)
plt.axis("off")
plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
plt.show()
```

Esegui cella (⌘/Ctrl+Enter)
cella eseguita dall'ultima modifica

eseguita da Diego Turri
11:33 (0 minuti fa)
esecuzione completata tra 3.265 s



```
import cv2
import matplotlib.pyplot as plt
%matplotlib inline

image = cv2.imread("neutral_result.jpg")
height, width = image.shape[:2]
resized_image = cv2.resize(image,(3*width, 3*height), interpolation = cv2.INTER_CUBIC)

fig = plt.gcf()
fig.set_size_inches(18, 10)
plt.axis("off")
plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
plt.show()
```



Si pensava che questo problema fosse dovuto dall'impurità dei dati interni al database. Se si andasse a guardare le immagini del database preso in considerazione, si può vedere con facilità che i soggetti a volte ridono ed a volte sembrano arrabbiati. Questo non avviene a causa di un impurità nel database poiché i soggetti stavano provando effettivamente dolore in quel momento, ma deriva più da una complessità evidente nell'espressione di dolore. Infatti anche un essere umano faticherebbe nel riconoscere come dolore alcune espressioni presenti del database. Eccone alcuni esempi:

Si può dire che nel campo del dolore è difficile trovare molte informazioni, infatti non è possibile trovare un algoritmo che rilevi direttamente il dolore ma è possibile addestrarne uno per il riconoscimento delle espressioni meno complesse come felicità e tristezza. Inoltre il database non contiene immagini di persone che stanno provando dolore in quel momento ma piuttosto immagini di persone che lo stanno simulando, ed è l'unico database accessibile di questo tipo. Probabilmente la simulazione del dolore introduce ulteriore rumore nell'espressione da riconoscere, ma il fatto che l'algoritmo non abbia funzionato ha permesso di notare due cose importanti:

- il dolore non è univoco, ma dipende molto da tanti fattori intrecciati tra loro e perciò anche l'espressione del dolore risulta confusa a volte. Anche se esiste un'espressione universale del dolore, le persone reagiscono in maniera molto diversificata agli stimoli o comunque pensano alla reazione simulata in maniera molto diversa tra loro.

- l'importanza assoluta che hanno i dati con cui si addestra l'algoritmo. Infatti è probabile che si confonda dopo avergli insegnato a riconoscere il dolore a causa dell'impurità dei dati ricevuti come input.

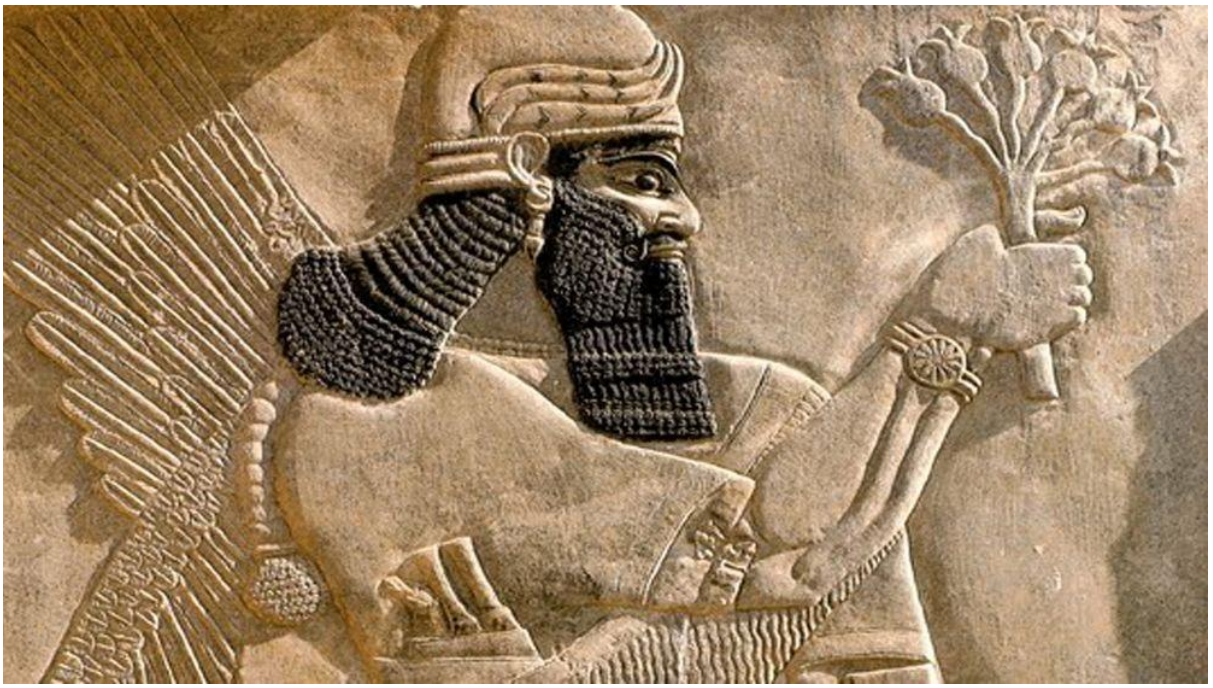
è possibile notare come sia relativamente semplice riconoscere e separare espressioni come felicità e tristezza rispetto alla difficoltà che si incontra nel separare la tristezza dal dolore, la rabbia dal dolore o addirittura perfino la felicità dal dolore basti pensare alle foto nel database di espressioni di dolore simulato in cui molte persone avevano una

espressione sorridente. Questo è il motivo centrale del perché non è stato possibile trovare algoritmi che riguardassero il riconoscimento del dolore. Probabilmente essendo più difficile da riconoscere, più difficile da separare dalle altre espressioni e meno popolare tra le persone, finisce per non avere la stessa diffusione e popolarità rispetto ad altre espressioni più comuni.

Analogia con la Storia

Un altra cosa curiosa emersa da questo “fallimento” è l’analogia tra la difficoltà riscontrata nell’affrontare l’argomento del dolore e la difficoltà che hanno riscontrato le varie culture nel affrontarlo durante il corso della storia

Si parte dalla visione del dolore della cultura babilonese che veniva guarito attraverso la “magia” e miscugli di erbe naturali (infatti i medici babilonesi erano chiamati medici-maghi)



alla visione del dolore come una punizione divina che hanno sempre avuto religioni tipo l’ebraismo in cui il curatore era a tutti gli effetti il sacerdote

Si arriva alle culture asiatiche dove il dolore nella storia non viene combattuto ma viene accettato come parte della vita. Ad esempio, nel buddismo, il superamento del dolore avveniva attraverso il raggiungimento della pace interiore.

Quindi ogni popolo e cultura per nei confronti del dolore reagiva in modo diverso ma sempre in modo abbastanza astratto. Fino all’arrivo di Ippocrate che nel quarto secolo avanti cristo descrisse il dolore finalmente come uno squilibrio fisico e quindi una cosa che poteva essere fisicamente spiegata e curata.

La storia poi come sappiamo ha fatto il suo corso, le invasioni barbariche portarono all'avvento nel medioevo e l'illuminismo fece progredire la scienza fino ai giorni nostri. Oggi infatti sappiamo che tutti loro avevano una parte di ragione nel descrivere il dolore poichè non è ne una cosa prettamente psicologica ne prettamente fisica ma è appunto un complicato intreccio tra le due cose. Quindi esattamente è esattamente ciò con cui

Abbiamo visto quindi come il nostro sforzo nell'affrontare un argomento complesso come il dolore non è un avvenimento isolato ma riecheggia durante tutto il corso della storia

Algoritmo Finale

Infine c'è una buona notizia per il futuro del Machine Learning nel campo del dolore
A san Diego è stato sviluppato un algoritmo di apprendimento automatico per il riconoscimento delle espressioni chiamato CERT (Computer Expression Recognition Toolbox) il quale é stato testato per riconoscere l'espressione di dolore reale nelle persone da un espressione di dolore che viene simulata. I test effettuati hanno dato come risultato che un essere umano seppure sia naturalmente in grado di percepire correttamente le espressioni ed emozioni dei suoi simili non supera il 50% di accuratezza quando si parla di riconoscimento del dolore simulato. Inoltre anche se addestrato la sua capacità di individuare espressioni di dolore reale rimane inferiore al 60%. Utilizzando l'algoritmo CERT si è dimostrata la potenzialità del Machine Learning. L'algoritmo ha ottenuto una precisione dell'85% sul riconoscimento del dolore simulato. Siamo ancora agli albori di queste tecnologie e giá in questo specifico campo l'algoritmo CERT ottiene una precisione largamente maggiore rispetto ad un essere umano. Questo risultato dimostra anche che il futuro spingerà molto in questa direzione e quindi che algoritmi come CERT avranno un enorme effetto sulla società ed il modo in cui vivremo nei prossimi anni