Facial Pain Detection

- **Dataset: The data used in this project is UNBC-McMaster shoulder pain expression which consists of nearly 48000 thousand frames extracted from videos of patient's facial expression during treatment, it's labeled from 0 to 10 as a pain scale.**

- **Data preparation process: The dataset consists of**

  - 400029 images of label 0 (No pain).

  -3128 images of label 1.
  -2351 images of label 2.
  -1412 images of label 3.
  -802 images of label 4.
  -242 images of label 5.
  -270 images of label 6.
  -53 images of label 7.
  -78 images of label 8.
  -32 images of label 9.
  -10 images of label 10.

  so clearly, it is not balanced data and to balanced it we grouped all the pain images (above 0 label) into one label as1 (8369 images) and cut the no pain images to 8471 images so it can be balanced and trainable, All images resized to 250*250to reduce ram usage during training on colab.

- **Model used: Regular CNN structural used with auto tuning on vgg16 neural network to boost model accuracy which reached 83% classifying 2classes (Pain or no Pain).**

- **Training and testing time: Training Tooks about 30 minute to train on 13 epochs and testing took about 2 seconds.**

- **Deploying part: The model deployed on desktop application made using Tkinter library with python.**

- **Colab notebook link: - https://colab.research.google.com/drive/1YGtCxL9L3EWq9bNehQv-hek71CGnkrWg?usp=share_link.**

- **Screen shoots: -**

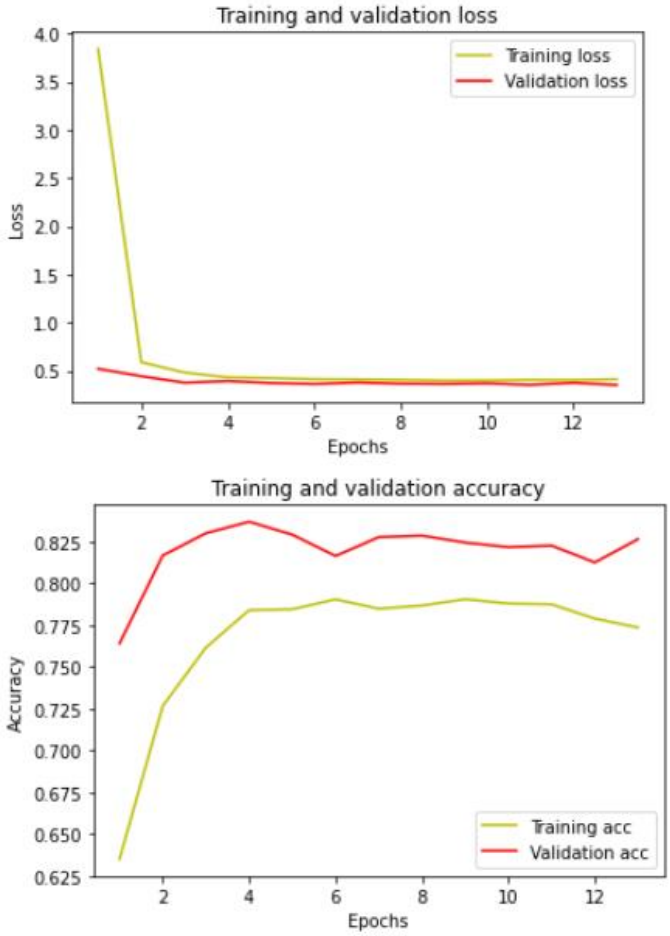
- Model structural

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
block1_conv1 (Conv2D)        (None, 250, 250, 64)      1792

block1_conv2 (Conv2D)        (None, 250, 250, 64)      36928

block1_pool (MaxPooling2D)   (None, 125, 125, 64)      0

block2_conv1 (Conv2D)        (None, 125, 125, 128)     73856

block2_conv2 (Conv2D)        (None, 125, 125, 128)     147584

block2_pool (MaxPooling2D)   (None, 62, 62, 128)       0

block3_conv1 (Conv2D)        (None, 62, 62, 256)       295168

block3_conv2 (Conv2D)        (None, 62, 62, 256)       590080

block3_conv3 (Conv2D)        (None, 62, 62, 256)       590080

block3_pool (MaxPooling2D)   (None, 31, 31, 256)       0

block4_conv1 (Conv2D)        (None, 31, 31, 512)       1180160

block4_conv2 (Conv2D)        (None, 31, 31, 512)       2359808

block4_conv3 (Conv2D)        (None, 31, 31, 512)       2359808

block4_pool (MaxPooling2D)   (None, 15, 15, 512)       0

block5_conv1 (Conv2D)        (None, 15, 15, 512)       2359808

block5_conv2 (Conv2D)        (None, 15, 15, 512)       2359808

block5_conv3 (Conv2D)        (None, 15, 15, 512)       2359808

block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0

flatten (Flatten)            (None, 25088)             0

dense (Dense)                (None, 512)               12845568

dropout (Dropout)            (None, 512)               0

dense_1 (Dense)              (None, 256)               131328

dropout_1 (Dropout)          (None, 256)               0

dense_2 (Dense)              (None, 128)               32896

dropout_2 (Dropout)          (None, 128)               0

dense_3 (Dense)              (None, 2)                 258

=================================================================
```

- **Training**

```
13472
(13472, 2)
3368
(3368, 2)
Train on 13472 samples, validate on 3368 samples
Epoch 1/13
13472/13472 [==============================] - ETA: 0s - loss: 3.8421 - acc: 0.6350/usr/local/lib/python3.7/dist-packages/keras/eng
  updates = self.state_updates
13472/13472 [==============================] - 127s 9ms/sample - loss: 3.8421 - acc: 0.6350 - val_loss: 0.5211 - val_acc: 0.7640
Epoch 2/13
13472/13472 [==============================] - 115s 9ms/sample - loss: 0.5890 - acc: 0.7265 - val_loss: 0.4449 - val_acc: 0.8165
Epoch 3/13
13472/13472 [==============================] - 115s 9ms/sample - loss: 0.4822 - acc: 0.7615 - val_loss: 0.3763 - val_acc: 0.8299
Epoch 4/13
13472/13472 [==============================] - 115s 9ms/sample - loss: 0.4328 - acc: 0.7838 - val_loss: 0.3938 - val_acc: 0.8367
Epoch 5/13
13472/13472 [==============================] - 115s 9ms/sample - loss: 0.4247 - acc: 0.7844 - val_loss: 0.3732 - val_acc: 0.8290
Epoch 6/13
13472/13472 [==============================] - 115s 9ms/sample - loss: 0.4137 - acc: 0.7903 - val_loss: 0.3627 - val_acc: 0.8162
Epoch 7/13
13472/13472 [==============================] - 115s 9ms/sample - loss: 0.4114 - acc: 0.7847 - val_loss: 0.3790 - val_acc: 0.8275
Epoch 8/13
13472/13472 [==============================] - 115s 9ms/sample - loss: 0.4068 - acc: 0.7866 - val_loss: 0.3660 - val_acc: 0.8284
Epoch 9/13
13472/13472 [==============================] - 115s 9ms/sample - loss: 0.3992 - acc: 0.7904 - val_loss: 0.3637 - val_acc: 0.8242
Epoch 10/13
13472/13472 [==============================] - 115s 9ms/sample - loss: 0.4005 - acc: 0.7878 - val_loss: 0.3693 - val_acc: 0.8216
Epoch 11/13
13472/13472 [==============================] - 115s 9ms/sample - loss: 0.4064 - acc: 0.7873 - val_loss: 0.3543 - val_acc: 0.8224
Epoch 12/13
13472/13472 [==============================] - 115s 9ms/sample - loss: 0.4049 - acc: 0.7788 - val_loss: 0.3755 - val_acc: 0.8124
Epoch 13/13
13472/13472 [==============================] - 115s 9ms/sample - loss: 0.4130 - acc: 0.7735 - val_loss: 0.3533 - val_acc: 0.8263
```

- **Model loss and accuracy**



- **How to run it**:-
  -open your visual code, open the project folder and inside your terminal write (`pip install -r requirements.txt`) to install the required library to run the project.

  -run the UI.py file.

- **NOTES:** to obtain good accuracy the uploaded images and the captured images must be in excellent quality, otherwise the results will be inaccurate and to capture an image press the ‘’q’’ button**.**