# CAP 5627 Affective Computing

## Project Deliverable 2 - Report

# Graduate Group – 3

**Team members and contributions:**

***Heamnath Balasubramanian:***
*Worked on image pre-processing and contributed equally in the paper and report documentation.*

***Lalithabhinaya Mallu:***
*Worked on transfer learning and fine tuning the models. Contributed equally in the paper and report documentation.*

***Mathan Kumar Venkateswaran:***
*Worked on CNN and fine-tuning models. Contributed equally in the paper and report documentation.*

***Padmavathi Siddi:***
*Worked on transfer learning and fine tuning the models. Contributed equally in the paper and report documentation.*

***Sravya Kodali:***
*Worked on manual data Augmentation and contributed equally in paper and report.*

# FACIAL CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORKS

Below is the report which shows the output of script for classification of data using CNN.
In the terminal output, Pain is represented as 0 and No pain is represented as 1.

## 1. Output:

The Confusion matrix, classification accuracy, precision, recall and FI Score values are shown below.

```
Found 3495 images belonging to 2 classes.
Accuracy =  0.685550790031751
loss =  0.6134912153085073
Confusion Matrix
[[ 161 1134]
 [ 187 2013]]
Classification Report
              precision    recall  f1-score   support

        Pain       0.46      0.12      0.20      1295
     No Pain       0.64      0.92      0.75      2200

   micro avg       0.62      0.62      0.62      3495
   macro avg       0.55      0.52      0.47      3495
weighted avg       0.57      0.62      0.55      3495
```

Figure 1: Output of CNN

**Final Model**

| **Layer (type)** | **Output Shape** | **Param #** |
|---|---|---|
| conv2d_1 (Conv2D) | (None, 160, 160, 32) | 320 |
| batch_normalization_1(Batch) | (None, 160, 160, 32) | 128 |
| re_lu_1 (ReLU) | (None, 160, 160, 32) | 0 |
| max_pooling2d_1 (MaxPooling2 ) | (None, 80, 80, 32) | 0 |
| dropout_1 (Dropout) | (None, 80, 80, 32) | 0 |
| conv2d_2 (Conv2D) | (None, 80, 80, 64) | 18496 |
| batch_normalization_2 (Batch) | (None, 80, 80, 64) | 256 |

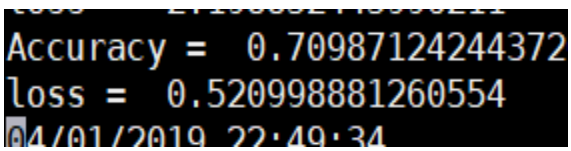
| | | |
|---|---|---|
| re_lu_2 (ReLU) | (None, 80, 80, 64) | 0 |
| max_pooling2d_2 (MaxPooling2) | (None, 40, 40, 64) | 0 |
| dropout_2 (Dropout) | (None, 40, 40, 64) | 0 |
| conv2d_3 (Conv2D) | (None, 40, 40, 128) | 73856 |
| batch_normalization_3 (Batch) | (None, 40, 40, 128) | 512 |
| re_lu_3 (ReLU) | (None, 40, 40, 128) | 0 |
| max_pooling2d_3 (MaxPooling2) | (None, 20, 20, 128) | 0 |
| dropout_3 (Dropout) | (None, 20, 20, 128) | 0 |
| flatten_1 (Flatten) | (None, 51200) | 0 |
| dense_1 (Dense) | (None, 16) | 819216 |
| batch_normalization_4 (Batch) | (None, 16) | 64 |
| re_lu_4 (ReLU) | (None, 16) | 0 |
| dropout_4 (Dropout) | (None, 16) | 0 |
| dense_2 (Dense) | (None, 1) | 17 |

Total params: 912,865
Trainable params: 912,385
Non-trainable params: 480
_____

## 2. Extra Credit: (Method and Accuracy of the model)

Since there is an imbalance in the data we have the accuracy that is fluctuating, and we have submitted the average accuracy (68.55) for the model. The accuracy is in the range 65.55-70.98 as shown in the figure below.

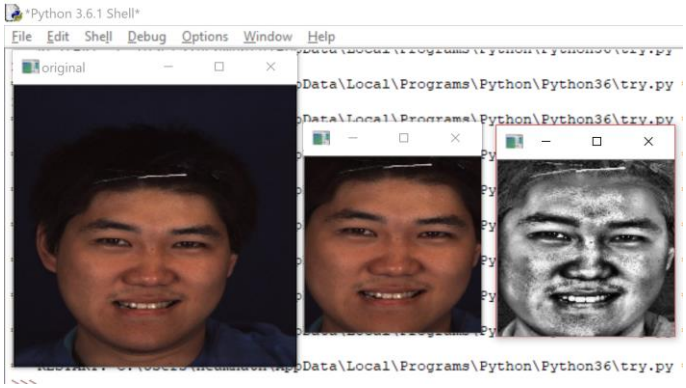
**Highest accuracy** for the same model.



Another significant Models' Output:

| Different Parameters | Accuracy | loss |
|---|---|---|
| SGD & Binary mode classes | 60.88 | 0.9177 |
| RMS & Binary mode classes | 62.48 | 0.7289 |
| RMS & Categorical Loss | 64.03 | 0.6275 |
| SGD& Categorical Loss | 62.94 | 2.1988 |

## 3. Original Face Vs Cropped Face:

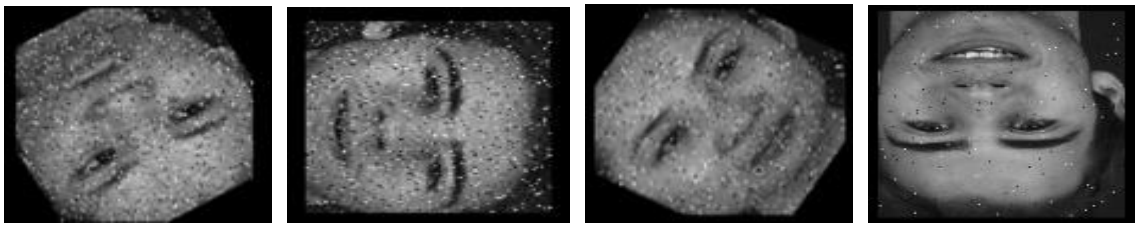

## 4. Accuracy of proposed model and justification

### Accuracy: 68.55%

A Fine-tuned CNN model included 3 convolutional layers one dense with RMSprop as optimizer with Binary Cross-entropy as the loss function. Dropout layer was used at each layer along with batch normalization. In this model, early-stopping was used with patience=100 with epoch=200 with a learning rate of 0.001 and decay rate as 1e-6.

## 5. Manual Augmentation

Manual augmentation is done using OpenCV. For each original image, rotation, translation was applied, and some noise is applied. Varying noise level, randomizer level, border mode affects the output images. Noise level lies between 0 and 1. Randomizer level indicates the level of change (when compared to input image) in the output image. Few augmented images were shown below.



We have done the augmentation manually and saved the images on to the disk. We doubled the training data for both pain and No pain classes. However, we are not using this data to train our model as we are not getting good accuracy with this data. We are using inbuilt augmentation functions for training the model. The following functions were used in the manual data augmentation code for rotation and translation.

**cv2.getRotationMatrix2D** function is used to calculate affine matrix of two-dimensional rotation. Center of the rotation in the original image, rotation angle in degrees and scale factor are the parameters for this function in python.

**cv2.warpAffine** function is used to apply affine transformation to the original image. For this function the parameters are original image, M (2x3 transformation matrix), size of the output image. This function returns an output image.

**Random.uniform** function is used to draw samples from uniform distribution. Any value in the given interval (low, high) can be picked up by the uniform function.