Dhruv Darda IIT Gandhinagar

# Smile Classification

I'mbesidesyou Inc.

## Code:

https://colab.research.google.com/drive/15doWVU_9bCf-CZFW4zojd5UTT0aO8hSA?usp=sharing

## Ideology:

An image classification dataset is given and so training a convolutional neural network to classify was the first thought. Since training a whole neural network would take many resources( GPU and time) which was a constraint, I used a pre-trained Densenet-161 model as a feature extractor and trained the classification layer to classify smile.

## Methodology:

First, I analyzed the dataset and found that almost 70% of the dataset was biased to one class, 'NOT Smile'. To overcome this, there were three options available, oversampling, class-weighing, and a combination of oversampling and undersampling.
Training the model without using any of the above methods leads to model learning to predict 'NOT Smile' for any input to achieve 70% accuracy.
Using oversampling leads to overfitting of the data as the ratio of the number of examples in each of the classes is very high. The model predicts the other two classes with a total accuracy of 35% accuracy.
Now, using the class-weighing gives almost 68% accuracy.
I also tried using a combination of both oversampling and class-weighing with different weights but it did not give much difference in results from just using oversampling.
Apart from accuracy, recall and precision, recall and F1 score should also be considered as the dataset is biased. Sklearn's classification report is used for this purpose while

## Code Flow:

To start with, I decided which type of dataset is given from among the two types of dataset types, Map style and Iterable dataset. Since the given dataset was a map style dataset, I started

with creating a custom Dataset class that maps each image with its corresponding label in the CSV file.
After creating the dataset, the next step was to batchify the dataset, I used a custom Sampler for the dataloader which is similar to the 'WeightedRandomSampler' of PyTorch. I also used different transforms to create randomness in the dataset.
The next step was to choose a model, I choose a pre-trained Densenet 161 feature extraction so that there is no overfitting.
The next step was to create a training loop and compute losses and accuracy.

## Classification Report:

```
Best val Acc: 0.684668
                precision    recall  f1-score   support

     NOT smile       0.70      0.91      0.79      4484
positive smile       0.00      0.00      0.00       630
negative smile       0.20      0.09      0.13      1328

      accuracy                           0.65      6442
     macro avg       0.30      0.33      0.30      6442
  weighted avg       0.53      0.65      0.57      6442

--- 627.1707680225372 seconds ---
```

## Confusion Matrix: