Data Preprocessing

It is a data mining technique that transforms raw data into an understandable format. Raw data (real world data) is always incomplete and that data cannot be sent through a model. That would cause certain errors. That is why we need to preprocess data before sending through a model.

**Steps in Data Preprocessing**

Here are the steps we have followed;  
1. Import libraries  
2. Read data  
3. Augmentations

**1. Import libraries**

As main libraries, We are using Numpy and openCV;  
**openCV**: Use for reading the images, resizing and gray scaling.  
**Numpy**: a fundamental package for scientific computing with Python.

We used **sklearn** libraries to split my data into train and test.

We used **tensorflow** libraries to apply augmentation on my train data.

**2. Read Data**

We have a function called path\_split which extracts image paths existing in the dataset and then splits to train and test sets.

: PARAM dataset\_path: the path of dataset

: PARAM seed: the seed required to random shuffle files

: return: train\_path, test\_path

Then, we have data\_extractor function firstly, with the help of openCV libraries we read, resize the images and gray scale them. After that we label our data. Now we need to normalize the data, since when using the image as it is and passing through a Deep Neural Network, the computation of high numeric values may become more complex. To reduce this, we can normalize the values to range from 0 to 1.

In this way, the numbers will be small and the computation becomes easier and faster.  
As the pixel values range from 0 to 256, apart from 0 the range is 255. So dividing all the values by 255 will convert it to range from 0 to 1.

: PARAM image\_paths: the input image path

: PARAM height: resized image height

: PARAM width: resized image width

: return: image and label arrays

data= []

labels = []

for imagepath in image\_paths:

image = cv2.imread(imagepath)

if gray:

image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

image = cv2.resize(image, (img\_height, img\_width), interpolation=cv2.INTER\_AREA)

label = imagepath.split(os.sep)[-2]

label = int(label)

labels.append(label)

data.append(image)

return np.array(data, dtype='float') / 255.0, np.array(labels)

**Let’s look at the data**

So the dataset is labeled as 0s and 1s

* 0 = non Smile
* 1 = Smile

**3. Augmentations**

Data augmentation is a technique to boost the performance of your neural network model by creating extra or fake data. The more examples the model has to learn from, the better it can recognize the difference in images, in other words more data helps the model generalize better.

One easy way to getting more data is **data Augmentation** which transforms the data you already have in ways, so that preserve the image class. For example, whether a car facing left or right doesn’t change the fact that it is a car and not a truck or bus, and that’s the whole idea behind data augmentation; add in some extra fake data that looks like the real data and your model will improve.

There are many types of transformations applied on dataset when augmenting such as rotating, adjusting color or contrast, warping the image and many more things, usually applied in combination. In our code we have used random horizontal flip, contrast adjustment, rotation, width and height translation.

You can do data augmentation in two ways.

* Include it in a data pipeline with a function like ImageDataGenerator
* Include it in a model using preprocessing layers

Data augmentation is usually done online, when the images are being fed into the network for training, since the image transformations will be computed on the GPU instead of CPU, speeding up the training, this is why we took the second approach.

def augmentation\_layer(imgs\_height, imgs\_width, n\_channels):

return tf.keras.Sequential([

tf.keras.layers.RandomFlip('horizontal', input\_shape=(imgs\_height, imgs\_width, n\_channels)),

preprocessing.RandomContrast(factor=0.3),

preprocessing.RandomWidth(factor=0.15),

preprocessing.RandomRotation(factor=0.20),

preprocessing.RandomTranslation(height\_factor=0.1, width\_factor=0.1)])

We have defined a function (augmentation\_layer) which returns a Keras Sequential model containing:

* RandomFlip: switches the images horizontally from left to right or vice versa
* RandomContrast: adjust randomly the images contrast with factor of 0.3
* RandomWidth: transforms randomly the images width with factor of 0.15
* RandomRotaion: rotates randomly the images with factor of 0.2
* RandomTranslatation: randomly transforms the images height and width with factor of 0.1

We have applied all of them in combination to whole dataset. A sample output has shown in the following.



Source: <https://towardsdatascience.com/data-pre-processing-techniques-you-should-know-8954662716d6>