```
from google.colab import drive
drive.mount('/content/gdrive', force_remount= True)
```

```
Mounted at /content/gdrive
```

```
# !pwd
%cd gdrive/My Drive/face/face
# !ls
```

```
/content/gdrive/My Drive/face/face
```

First of all, let's import necessary Packages

```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
%matplotlib inline
from math import sin, cos, pi
import cv2
from tqdm.notebook import tqdm

from keras.layers.advanced_activations import LeakyReLU
from keras.models import Sequential, Model, load_model
from keras.layers import Activation, Convolution2D, MaxPooling2D, BatchNormalization, Flatten
from keras.callbacks import ReduceLROnPlateau, ModelCheckpoint
from keras.optimizers import Adam
```

Input data

```
train_data = pd.read_csv('training.csv')
test_data = pd.read_csv('test.csv')
idlookup_data = pd.read_csv('IdLookupTable.csv')
```

```
# Expplore the data
train_data.head().T
```

	0	1	2	3	4
left_eye_center_x	66.0336	64.3329	65.0571	65.2257	66.7253
left_eye_center_y	39.0023	34.9701	34.9096	37.2618	39.6213
right_eye_center_x	30.227	29.9493	30.9038	32.0231	32.2448
right_eye_center_y	36.4217	33.4487	34.9096	37.2618	38.042
left_eye_inner_corner_x	59.5821	58.8562	59.412	60.0033	58.5659
left_eye_inner_corner_y	39.6474	35.2743	36.321	39.1272	39.6213
left_eye_outer_corner_x	73.1303	70.7227	70.9844	72.3147	72.5159
left_eye_outer_corner_y	39.97	36.1872	36.321	38.381	39.8845
right_eye_inner_corner_x	36.3566	36.0347	37.6781	37.6186	36.9824
right_eye_inner_corner_y	37.3894	34.3615	36.321	38.7541	39.0949
right_eye_outer_corner_x	23.4529	24.4725	24.9764	25.3073	22.5061
right_eye_outer_corner_y	37.3894	33.1444	36.6032	38.0079	38.3052
left_eyebrow_inner_end_x	56.9533	53.9874	55.7425	56.4338	57.2496
left_eyebrow_inner_end_y	29.0336	28.2759	27.5709	30.9299	30.6722
left_eyebrow_outer_end_x	80.2271	78.6342	78.8874	77.9103	77.7629
left_eyebrow_outer_end_y	32.2281	30.4059	32.6516	31.6657	31.7372
right_eyebrow_inner_end_x	40.2276	42.7289	42.1939	41.6715	38.0354
right_eyebrow_inner_end_y	29.0023	26.146	28.1355	31.05	30.9354
right_eyebrow_outer_end_x	16.3564	16.8654	16.7912	20.458	15.9259
right_eyebrow_outer_end_y	29.6475	27.0589	32.0871	29.9093	30.6722
nose_tip_x	44.4206	48.2063	47.5573	51.8851	43.2995
nose_tip_y	57.0668	55.6609	53.5389	54.1665	64.8895
mouth_left_corner_x	61.1953	56.4214	60.8229	65.5989	60.6714
mouth_left_corner_y	79.9702	76.352	73.0143	72.7037	77.5232
mouth_right_corner_x	28.6145	35.1224	33.7263	37.2455	31.1918
mouth_right_corner_y	77.389	76.0477	72.732	74.1955	76.9973
mouth_center_top_lip_x	43.3126	46.6846	47.2749	50.3032	44.9627
mouth_center_top_lip_y	72.9355	72.9666	72.1912	72.9017	73.7074

```
test_data.head()
```

	ImageId	Image
0	1	182 183 182 182 180 180 176 169 156 137 124 10...
1	2	76 87 81 72 65 59 64 76 69 42 31 38 49 58 58 4...
2	3	177 176 174 170 169 169 168 166 166 166 161 14...
3	4	176 174 174 175 174 174 176 176 175 171 165 15...

```
idlookup_data.head().T
```

	0	1	2	3	
RowId	1	2	3	4	
ImageId	1	1	1	1	
FeatureName	left_eye_center_x	left_eye_center_y	right_eye_center_x	right_eye_center_y	left_eye
Location	NaN	NaN	NaN	NaN	

Check for any images with missing pixel values

```
print("Length of train data: {}".format(len(train_data)))
print("Number of Images with missing pixel values: {}".format(len(train_data) - int(train_dat
```

```
Length of train data: 7049
Number of Images with missing pixel values: 0
```

Find columns having Null values and their counts

```
train_data.isnull().sum()
```

```
left_eye_center_x              10
left_eye_center_y              10
right_eye_center_x             13
right_eye_center_y             13
left_eye_inner_corner_x      4778
left_eye_inner_corner_y      4778
left_eye_outer_corner_x      4782
left_eye_outer_corner_y      4782
right_eye_inner_corner_x     4781
right_eye_inner_corner_y     4781
right_eye_outer_corner_x     4781
right_eye_outer_corner_y     4781
left_eyebrow_inner_end_x     4779
left_eyebrow_inner_end_y     4779
left_eyebrow_outer_end_x     4824
left_eyebrow_outer_end_y     4824
right_eyebrow_inner_end_x    4779
right_eyebrow_inner_end_y    4779
```

```
right_eyebrow_outer_end_x    4813
right_eyebrow_outer_end_y    4813
nose_tip_x                      0
nose_tip_y                      0
mouth_left_corner_x          4780
mouth_left_corner_y          4780
mouth_right_corner_x         4779
mouth_right_corner_y         4779
mouth_center_top_lip_x       4774
mouth_center_top_lip_y       4774
mouth_center_bottom_lip_x      33
mouth_center_bottom_lip_y      33
Image                           0
dtype: int64
```

Define a function to plot facial keypoints with images

```
def plot_sample(image, keypoint, axis, title):
    image = image.reshape(96,96)
    axis.imshow(image, cmap='gray')
    axis.scatter(keypoint[0::2], keypoint[1::2], marker='x', s=20)
    plt.title(title)
```

Separate data into clean & unclean subsets

```
%%time

clean_train_data = train_data.dropna()
print("clean_train_data shape: {}".format(np.shape(clean_train_data)))

# https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.fillna.html
unclean_train_data = train_data.fillna(method = 'ffill')
print("unclean_train_data shape: {}\n".format(np.shape(unclean_train_data)))
```

```
clean_train_data shape: (2140, 31)
unclean_train_data shape: (7049, 31)

CPU times: user 14.1 ms, sys: 5.28 ms, total: 19.4 ms
Wall time: 21.2 ms
```

```
include_unclean_data = True    # Whether to include samples with missing keypoint values. Not
sample_image_index = 20    # Index of sample train image used for visualizing various augment
```

```
%%time

def load_images(image_data):
    images = []
    for idx, sample in image_data.iterrows():
        image = np.array(sample['Image'].split(' '), dtype=int)
```

```
            image = np.reshape(image, (96,96,1))
            images.append(image)
        images = np.array(images)/255.
        return images

    def load_keypoints(keypoint_data):
        keypoint_data = keypoint_data.drop('Image',axis = 1)
        keypoint_features = []
        for idx, sample_keypoints in keypoint_data.iterrows():
            keypoint_features.append(sample_keypoints)
        keypoint_features = np.array(keypoint_features, dtype = 'float')
        return keypoint_features

    clean_train_images = load_images(clean_train_data)
    print("Shape of clean_train_images: {}".format(np.shape(clean_train_images)))
    clean_train_keypoints = load_keypoints(clean_train_data)
    print("Shape of clean_train_keypoints: {}".format(np.shape(clean_train_keypoints)))
    test_images = load_images(test_data)
    print("Shape of test_images: {}".format(np.shape(test_images)))

    train_images = clean_train_images
    train_keypoints = clean_train_keypoints
    fig, axis = plt.subplots()
    plot_sample(clean_train_images[sample_image_index], clean_train_keypoints[sample_image_index]

    if include_unclean_data:
        unclean_train_images = load_images(unclean_train_data)
        print("Shape of unclean_train_images: {}".format(np.shape(unclean_train_images)))
        unclean_train_keypoints = load_keypoints(unclean_train_data)
        print("Shape of unclean_train_keypoints: {}\n".format(np.shape(unclean_train_keypoints)))
        train_images = np.concatenate((train_images, unclean_train_images))
        train_keypoints = np.concatenate((train_keypoints, unclean_train_keypoints))
```

```
Shape of clean_train_images: (2140, 96, 96, 1)
Shape of clean_train_keypoints: (2140, 30)
Shape of test_images: (1783, 96, 96, 1)
Shape of unclean_train_images: (7049, 96, 96, 1)
```

We can observe that approx. 68% of data is missing for several keypoints, so we need to clean the data.

```
Wall time: 19.9 s
```

```
%%time

clean_train_data = train_data.dropna()
print("clean_train_data shape: {}".format(np.shape(clean_train_data)))

# https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.fillna.html
unclean_train_data = train_data.fillna(method = 'ffill')
print("unclean_train_data shape: {}\n".format(np.shape(unclean_train_data)))
```

```
clean_train_data shape: (2140, 31)
unclean_train_data shape: (7049, 31)

CPU times: user 12.6 ms, sys: 0 ns, total: 12.6 ms
Wall time: 12.3 ms
```

Data Augmentation

There are various augmentation choices

```
horizontal_flip = False
rotation_augmentation = True
brightness_augmentation = True
shift_augmentation = True
random_noise_augmentation = True


rotation_angles = [12]    # Rotation angle in degrees (includes both clockwise & anti-clockwi
pixel_shifts = [12]    # Horizontal & vertical shift amount in pixels (includes shift from al

NUM_EPOCHS = 80
BATCH_SIZE = 64
```

1. Performing Horizontal Flipping for Data Augmentation

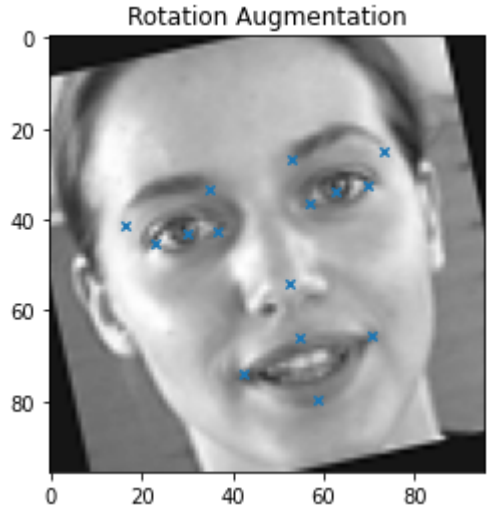
```
def left_right_flip(images, keypoints):
    flipped_keypoints = []
    flipped_images = np.flip(images, axis=2)   # Flip column-wise (axis=2)
    for idx, sample_keypoints in enumerate(keypoints):
```

```
        for idx, sample_keypoints in enumerate(keypoints):
            flipped_keypoints.append([96.-coor if idx%2==0 else coor for idx,coor in enumerate(sa
        return flipped_images, flipped_keypoints

    if horizontal_flip:
        flipped_train_images, flipped_train_keypoints = left_right_flip(clean_train_images, clean
        print("Shape of flipped_train_images: {}".format(np.shape(flipped_train_images)))
        print("Shape of flipped_train_keypoints: {}".format(np.shape(flipped_train_keypoints)))
        train_images = np.concatenate((train_images, flipped_train_images))
        train_keypoints = np.concatenate((train_keypoints, flipped_train_keypoints))
        fig, axis = plt.subplots()
        plot_sample(flipped_train_images[sample_image_index], flipped_train_keypoints[sample_imag
```

2. Performing Rotation Augmentation

```
    def rotate_augmentation(images, keypoints):
        rotated_images = []
        rotated_keypoints = []
        print("Augmenting for angles (in degrees): ")
        for angle in rotation_angles:    # Rotation augmentation for a list of angle values
            for angle in [angle,-angle]:
                print(f'{angle}', end='  ')
                M = cv2.getRotationMatrix2D((48,48), angle, 1.0)
                angle_rad = -angle*pi/180.     # Obtain angle in radians from angle in degrees (n
                # For train_images
                for image in images:
                    rotated_image = cv2.warpAffine(image, M, (96,96), flags=cv2.INTER_CUBIC)
                    rotated_images.append(rotated_image)
                # For train_keypoints
                for keypoint in keypoints:
                    rotated_keypoint = keypoint - 48.    # Subtract the middle value of the image
                    for idx in range(0,len(rotated_keypoint),2):
                        # https://in.mathworks.com/matlabcentral/answers/93554-how-can-i-rotate-a
                        rotated_keypoint[idx] = rotated_keypoint[idx]*cos(angle_rad)-rotated_keyp
                        rotated_keypoint[idx+1] = rotated_keypoint[idx]*sin(angle_rad)+rotated_ke
                    rotated_keypoint += 48.   # Add the earlier subtracted value
                    rotated_keypoints.append(rotated_keypoint)

        return np.reshape(rotated_images,(-1,96,96,1)), rotated_keypoints

    if rotation_augmentation:
        rotated_train_images, rotated_train_keypoints = rotate_augmentation(clean_train_images, c
        print("\nShape of rotated_train_images: {}".format(np.shape(rotated_train_images)))
        print("Shape of rotated_train_keypoints: {}\n".format(np.shape(rotated_train_keypoints)))
        train_images = np.concatenate((train_images, rotated_train_images))
        train_keypoints = np.concatenate((train_keypoints, rotated_train_keypoints))
        fig, axis = plt.subplots()
        plot_sample(rotated_train_images[sample_image_index], rotated_train_keypoints[sample_imag
```

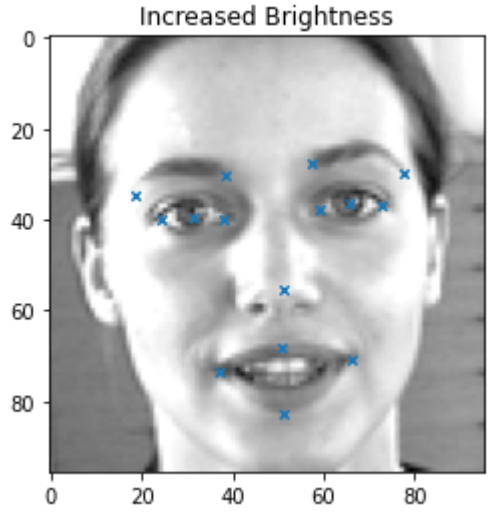
```
Augmenting for angles (in degrees):
12  -12
Shape of rotated_train_images: (4280, 96, 96, 1)
Shape of rotated_train_keypoints: (4280, 30)
```

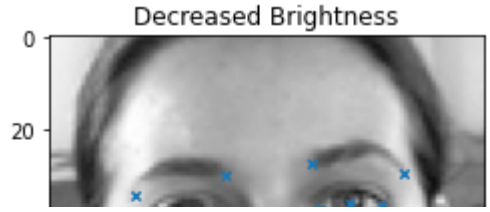


3. Performing Brightness Alteration for Data Augmentation

```
def alter_brightness(images, keypoints):
    altered_brightness_images = []
    inc_brightness_images = np.clip(images*1.2, 0.0, 1.0)    # Increased brightness by a fact
    dec_brightness_images = np.clip(images*0.6, 0.0, 1.0)    # Decreased brightness by a fact
    altered_brightness_images.extend(inc_brightness_images)
    altered_brightness_images.extend(dec_brightness_images)
    return altered_brightness_images, np.concatenate((keypoints, keypoints))

if brightness_augmentation:
    altered_brightness_train_images, altered_brightness_train_keypoints = alter_brightness(cl
    print(f"Shape of altered_brightness_train_images: {np.shape(altered_brightness_train_imag
    print(f"Shape of altered_brightness_train_keypoints: {np.shape(altered_brightness_train_k
    train_images = np.concatenate((train_images, altered_brightness_train_images))
    train_keypoints = np.concatenate((train_keypoints, altered_brightness_train_keypoints))
    fig, axis = plt.subplots()
    plot_sample(altered_brightness_train_images[sample_image_index], altered_brightness_train
    fig, axis = plt.subplots()
    plot_sample(altered_brightness_train_images[len(altered_brightness_train_images)//2+sampl
```

```
Shape of altered_brightness_train_images: (4280, 96, 96, 1)
Shape of altered_brightness_train_keypoints: (4280, 30)
```

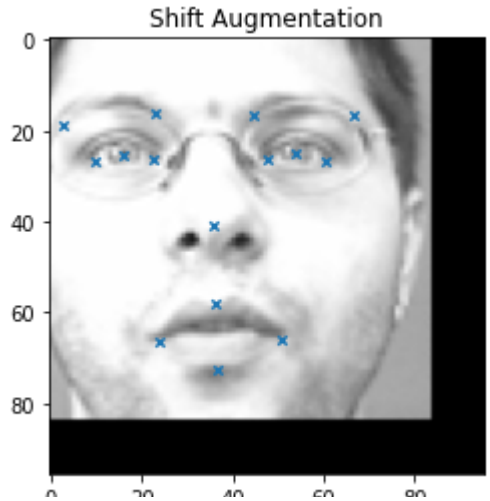




4. Performing Horizontal & Vertical shift

```
def shift_images(images, keypoints):
    shifted_images = []
    shifted_keypoints = []
    for shift in pixel_shifts:    # Augmenting over several pixel shift values
        for (shift_x,shift_y) in [(-shift,-shift),(-shift,shift),(shift,-shift),(shift,shift)
            M = np.float32([[1,0,shift_x],[0,1,shift_y]])
            for image, keypoint in zip(images, keypoints):
                shifted_image = cv2.warpAffine(image, M, (96,96), flags=cv2.INTER_CUBIC)
                shifted_keypoint = np.array([(point+shift_x) if idx%2==0 else (point+shift_y)
                if np.all(0.0<shifted_keypoint) and np.all(shifted_keypoint<96.0):
                    shifted_images.append(shifted_image.reshape(96,96,1))
                    shifted_keypoints.append(shifted_keypoint)
    shifted_keypoints = np.clip(shifted_keypoints,0.0,96.0)
    return shifted_images, shifted_keypoints

if shift_augmentation:
    shifted_train_images, shifted_train_keypoints = shift_images(clean_train_images, clean_tr
    print(f"Shape of shifted_train_images: {np.shape(shifted_train_images)}")
    print(f"Shape of shifted_train_keypoints: {np.shape(shifted_train_keypoints)}")
    train_images = np.concatenate((train_images, shifted_train_images))
    train_keypoints = np.concatenate((train_keypoints, shifted_train_keypoints))
    fig, axis = plt.subplots()
    plot_sample(shifted_train_images[sample_image_index], shifted_train_keypoints[sample_imag
```

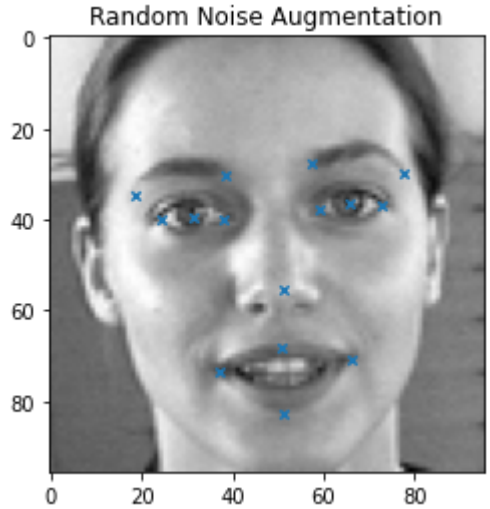
```
Shape of shifted_train_images: (6350, 96, 96, 1)
Shape of shifted_train_keypoints: (6350, 30)
```



5. Adding Random Noise for Data Augmentation

```
def add_noise(images):
    noisy_images = []
    for image in images:
        noisy_image = cv2.add(image, 0.008*np.random.randn(96,96,1))    # Adding random norma
        noisy_images.append(noisy_image.reshape(96,96,1))
    return noisy_images

if random_noise_augmentation:
    noisy_train_images = add_noise(clean_train_images)
    print(f"Shape of noisy_train_images: {np.shape(noisy_train_images)}")
    train_images = np.concatenate((train_images, noisy_train_images))
    train_keypoints = np.concatenate((train_keypoints, clean_train_keypoints))
    fig, axis = plt.subplots()
    plot_sample(noisy_train_images[sample_image_index], clean_train_keypoints[sample_image_in
```

```
Shape of noisy_train_images: (2140, 96, 96, 1)
```



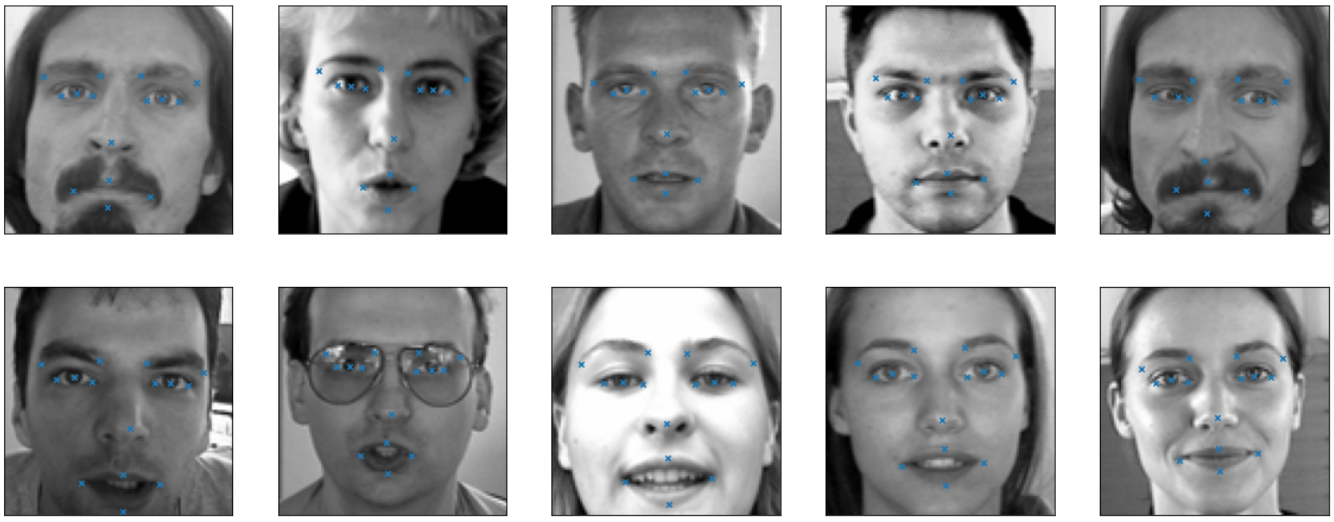
Visualize Train images & corresponding Keypoints

```
print("Shape of final train_images: {}".format(np.shape(train_images)))
print("Shape of final train_keypoints: {}".format(np.shape(train_keypoints)))

print("\n Clean Train Data: ")
fig = plt.figure(figsize=(20,8))
for i in range(10):
    axis = fig.add_subplot(2, 5, i+1, xticks=[], yticks=[])
    plot_sample(clean_train_images[i], clean_train_keypoints[i], axis, "")
plt.show()

if include_unclean_data:
    print("Unclean Train Data: ")
    fig = plt.figure(figsize=(20,8))
    for i in range(10):
        axis = fig.add_subplot(2, 5, i+1, xticks=[], yticks=[])
        plot_sample(unclean_train_images[i], unclean_train_keypoints[i], axis, "")
    plt.show()

if horizontal_flip:
    print("Horizontal Flip Augmentation: ")
    fig = plt.figure(figsize=(20,8))
    for i in range(10):
        axis = fig.add_subplot(2, 5, i+1, xticks=[], yticks=[])
        plot_sample(flipped_train_images[i], flipped_train_keypoints[i], axis, "")
    plt.show()

if rotation_augmentation:
    print("Rotation Augmentation: ")
    fig = plt.figure(figsize=(20,8))
    for i in range(10):
        axis = fig.add_subplot(2, 5, i+1, xticks=[], yticks=[])
        plot_sample(rotated_train_images[i], rotated_train_keypoints[i], axis, "")
    plt.show()

if brightness_augmentation:
    print("Brightness Augmentation: ")
    fig = plt.figure(figsize=(20,8))
    for i in range(10):
        axis = fig.add_subplot(2, 5, i+1, xticks=[], yticks=[])
        plot_sample(altered_brightness_train_images[i], altered_brightness_train_keypoints[i]
    plt.show()

if shift_augmentation:
    print("Shift Augmentation: ")
    fig = plt.figure(figsize=(20,8))
    for i in range(10):
        axis = fig.add_subplot(2, 5, i+1, xticks=[], yticks=[])
        plot_sample(shifted_train_images[i], shifted_train_keypoints[i], axis, "")
```

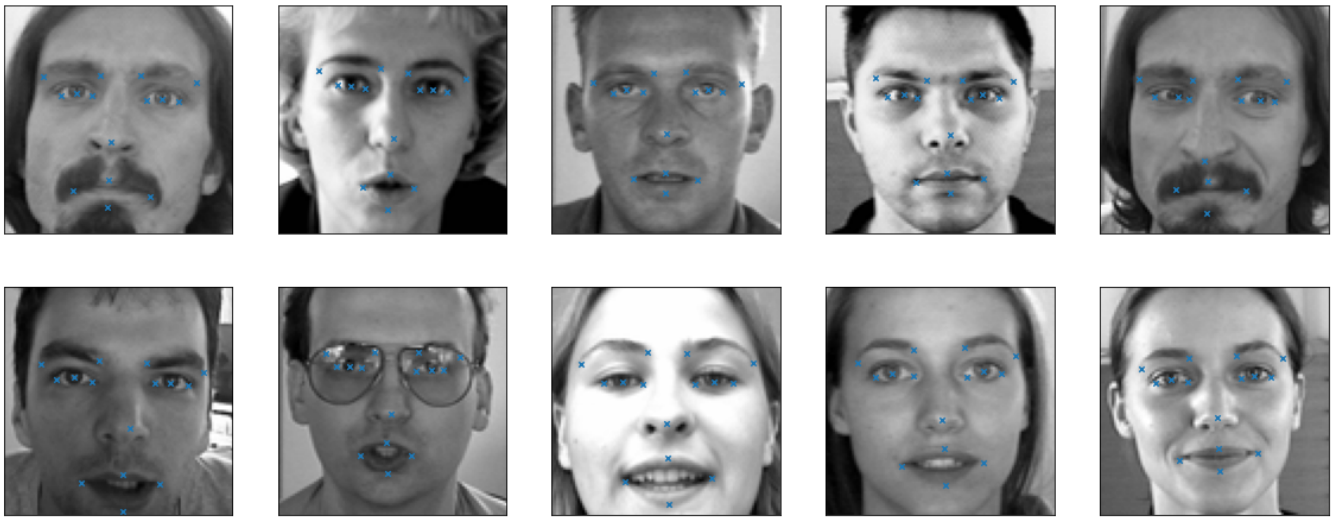
```
            plot_sample(unaltered_train_images[i], unaltered_train_keypoints[i], axis, "")
        plt.show()

    if random_noise_augmentation:
        print("Random Noise Augmentation: ")
        fig = plt.figure(figsize=(20,8))
        for i in range(10):
            axis = fig.add_subplot(2, 5, i+1, xticks=[], yticks=[])
            plot_sample(noisy_train_images[i], clean_train_keypoints[i], axis, "")
        plt.show()
```

```
Shape of final train_images: (26239, 96, 96, 1)
Shape of final train_keypoints: (26239, 30)
```

Clean Train Data:

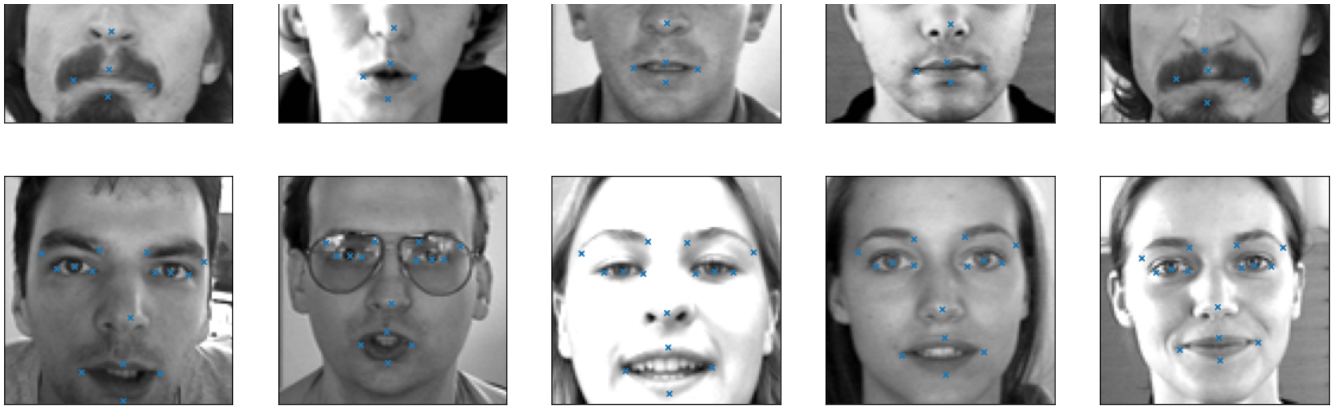


Unclean Train Data:

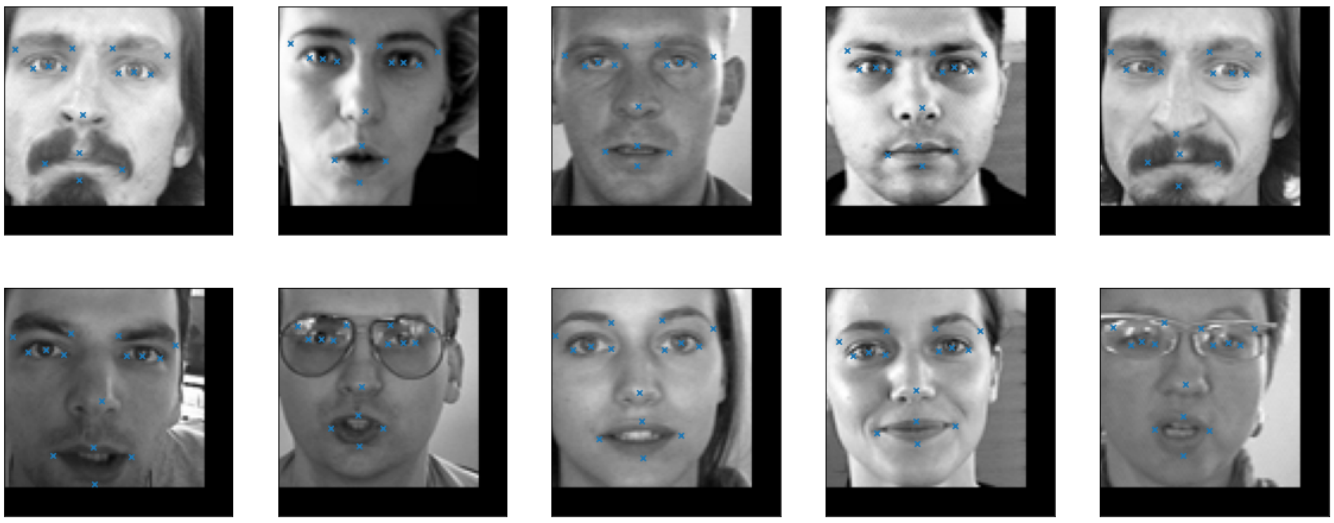


Rotation Augmentation:

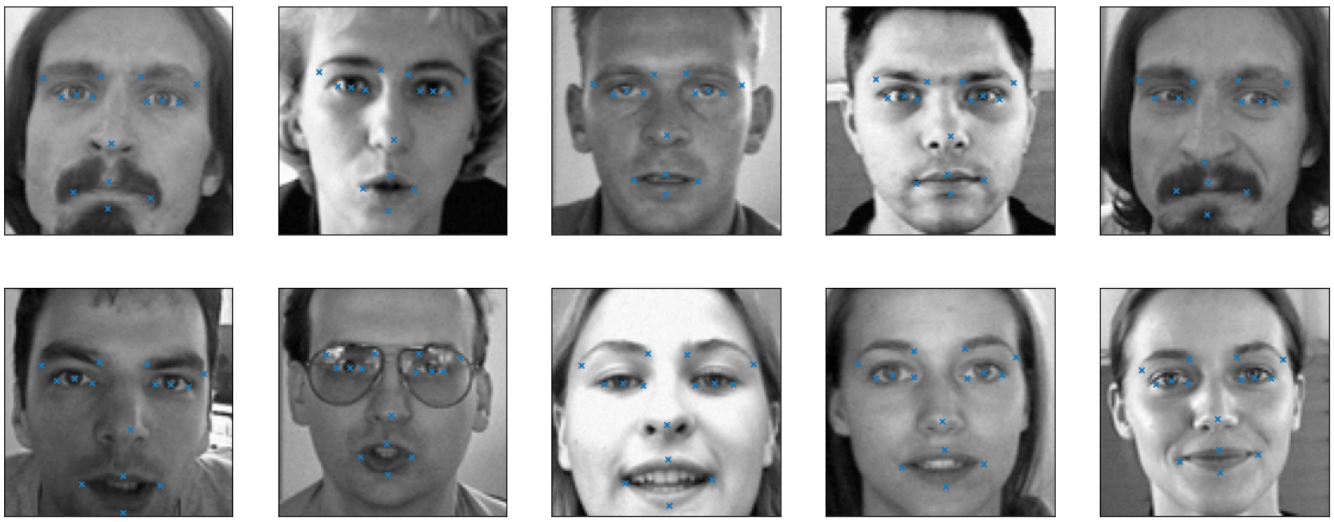
Brightness Augmentation:



Shift Augmentation:



Random Noise Augmentation:



Model

```
model = Sequential()

# Input dimensions: (None, 96, 96, 1)
model.add(Convolution2D(32, (3,3), padding='same', use_bias=False, input_shape=(96,96,1)))
model.add(LeakyReLU(alpha = 0.1))
model.add(BatchNormalization())
# Input dimensions: (None, 96, 96, 32)
model.add(Convolution2D(32, (3,3), padding='same', use_bias=False))
model.add(LeakyReLU(alpha = 0.1))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))

# Input dimensions: (None, 48, 48, 32)
model.add(Convolution2D(64, (3,3), padding='same', use_bias=False))
model.add(LeakyReLU(alpha = 0.1))
model.add(BatchNormalization())
# Input dimensions: (None, 48, 48, 64)
model.add(Convolution2D(64, (3,3), padding='same', use_bias=False))
model.add(LeakyReLU(alpha = 0.1))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))

# Input dimensions: (None, 24, 24, 64)
model.add(Convolution2D(96, (3,3), padding='same', use_bias=False))
model.add(LeakyReLU(alpha = 0.1))
model.add(BatchNormalization())
# Input dimensions: (None, 24, 24, 96)
model.add(Convolution2D(96, (3,3), padding='same', use_bias=False))
model.add(LeakyReLU(alpha = 0.1))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))

# Input dimensions: (None, 12, 12, 96)
model.add(Convolution2D(128, (3,3),padding='same', use_bias=False))
model.add(LeakyReLU(alpha = 0.1))
model.add(BatchNormalization())
# Input dimensions: (None, 12, 12, 128)
model.add(Convolution2D(128, (3,3),padding='same', use_bias=False))
model.add(LeakyReLU(alpha = 0.1))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))
```

```
# Input dimensions: (None, 6, 6, 128)
model.add(Convolution2D(256, (3,3),padding='same',use_bias=False))
model.add(LeakyReLU(alpha = 0.1))
model.add(BatchNormalization())
# Input dimensions: (None, 6, 6, 256)
model.add(Convolution2D(256, (3,3),padding='same',use_bias=False))
model.add(LeakyReLU(alpha = 0.1))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))

# Input dimensions: (None, 3, 3, 256)
model.add(Convolution2D(512, (3,3), padding='same', use_bias=False))
model.add(LeakyReLU(alpha = 0.1))
model.add(BatchNormalization())
# Input dimensions: (None, 3, 3, 512)
model.add(Convolution2D(512, (3,3), padding='same', use_bias=False))
model.add(LeakyReLU(alpha = 0.1))
model.add(BatchNormalization())

# Input dimensions: (None, 3, 3, 512)
model.add(Flatten())
model.add(Dense(512,activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(30))
model.summary()
```

```
conv2d_5 (Conv2D)            (None, 24, 24, 96)        82944
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 24, 24, 96)        0
_________________________________________________________________
batch_normalization_5 (Batch (None, 24, 24, 96)        384
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 12, 96)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 12, 12, 128)       110592
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 12, 12, 128)       0
_________________________________________________________________
batch_normalization_6 (Batch (None, 12, 12, 128)       512
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 12, 12, 128)       147456
_________________________________________________________________
leaky_re_lu_7 (LeakyReLU)    (None, 12, 12, 128)       0
_________________________________________________________________
batch_normalization_7 (Batch (None, 12, 12, 128)       512
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 128)         0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 6, 6, 256)         294912
_________________________________________________________________
leaky_re_lu_8 (LeakyReLU)    (None, 6, 6, 256)         0
_________________________________________________________________
batch_normalization_8 (Batch (None, 6, 6, 256)         1024
_________________________________________________________________
```

```
conv2d_9 (Conv2D)            (None, 6, 6, 256)         589824
_________________________________________________________________
leaky_re_lu_9 (LeakyReLU)    (None, 6, 6, 256)         0
_________________________________________________________________
batch_normalization_9 (Batch (None, 6, 6, 256)         1024
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 3, 3, 256)         0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 3, 3, 512)         1179648
_________________________________________________________________
leaky_re_lu_10 (LeakyReLU)   (None, 3, 3, 512)         0
_________________________________________________________________
batch_normalization_10 (Batc (None, 3, 3, 512)         2048
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 3, 3, 512)         2359296
_________________________________________________________________
leaky_re_lu_11 (LeakyReLU)   (None, 3, 3, 512)         0
_________________________________________________________________
batch_normalization_11 (Batc (None, 3, 3, 512)         2048
_________________________________________________________________
flatten (Flatten)            (None, 4608)              0
_________________________________________________________________
dense (Dense)                (None, 512)               2359808
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 30)                15390
=================================================================
Total params: 7,268,670
Trainable params: 7,264,318
Non-trainable params: 4,352
```

```
# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error', metrics=['mae', 'acc'])

# Train the model
history = model.fit(train_images, train_keypoints, epochs=NUM_EPOCHS, batch_size=BATCH_SIZE,
```

```
Epoch 1/80
 11/390 [.............................] - ETA: 31:30 - loss: 1316.6342 - mae: 29.2299 -
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
```

```
# summarize history for mean_absolute_error
try:
    plt.plot(history.history['mae'])
    plt.plot(history.history['val_mae'])
    plt.title('Mean Absolute Error vs Epoch')
    plt.ylabel('Mean Absolute Error')
    plt.xlabel('Epochs')
    plt.legend(['train', 'validation'], loc='upper right')
    plt.show()
    # summarize history for accuracy
    plt.plot(history.history['acc'])
    plt.plot(history.history['val_acc'])
    plt.title('Accuracy vs Epoch')
    plt.ylabel('Accuracy')
    plt.xlabel('Epochs')
    plt.legend(['train', 'validation'], loc='upper left')
    plt.show()
    # summarize history for loss
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('Loss vs Epoch')
    plt.ylabel('Loss')
    plt.xlabel('Epochs')
    plt.legend(['train', 'validation'], loc='upper left')
    plt.show()
except:
    print("One of the metrics used for plotting graphs is missing! See 'model.compile()'s `me
```

```
One of the metrics used for plotting graphs is missing! See 'model.compile()'s `metrics`
```

Fit the model on full dataset

```
model.fit(train_images, train_keypoints, epochs=NUM_EPOCHS, batch_size=BATCH_SIZE)
```

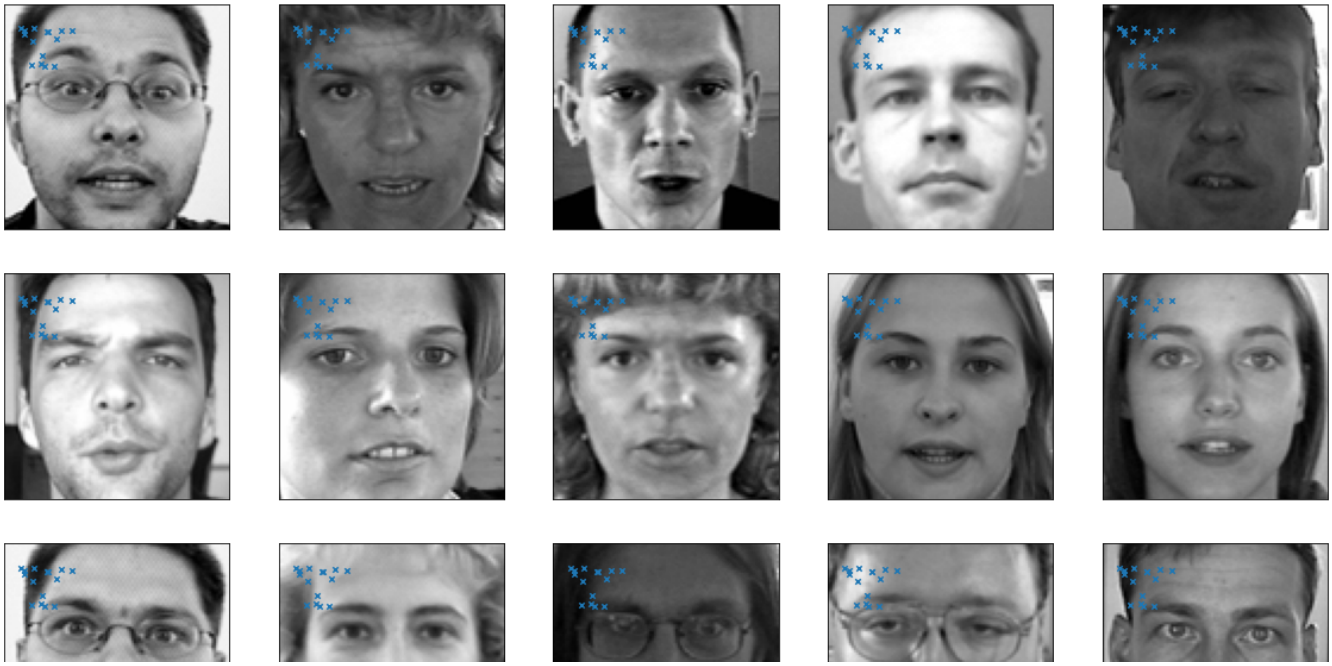
```
Epoch 1/80
--------------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-25-b696fd2240bb> in <module>()
----> 1 model.fit(train_images, train_keypoints, epochs=NUM_EPOCHS,
batch_size=BATCH_SIZE)
```

Predicting on Test Set

```
    quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
test_preds = model.predict(test_images)
    ---> 60                                             inputs, attrs, num_outputs)
```

Visualizing Test Predictions

```
fig = plt.figure(figsize=(20,16))
for i in range(20):
    axis = fig.add_subplot(4, 5, i+1, xticks=[], yticks=[])
    plot_sample(test_images[i], test_preds[i], axis, "")
plt.show()
```



Generating Submission File

```
feature_names = list(idlookup_data['FeatureName'])
image_ids = list(idlookup_data['ImageId']-1)
row_ids = list(idlookup_data['RowId'])

feature_list = []
for feature in feature_names:
    feature_list.append(feature_names.index(feature))

predictions = []
for x,y in zip(image_ids, feature_list):
    predictions.append(test_preds[x][y])

row_ids = pd.Series(row_ids, name = 'RowId')
locations = pd.Series(predictions, name = 'Location')
locations = locations.clip(0.0,96.0)
submission_result = pd.concat([row_ids,locations],axis = 1)
submission_result.to_csv('submission.csv',index = False)
```

✓ 4s completed at 3:04 AM