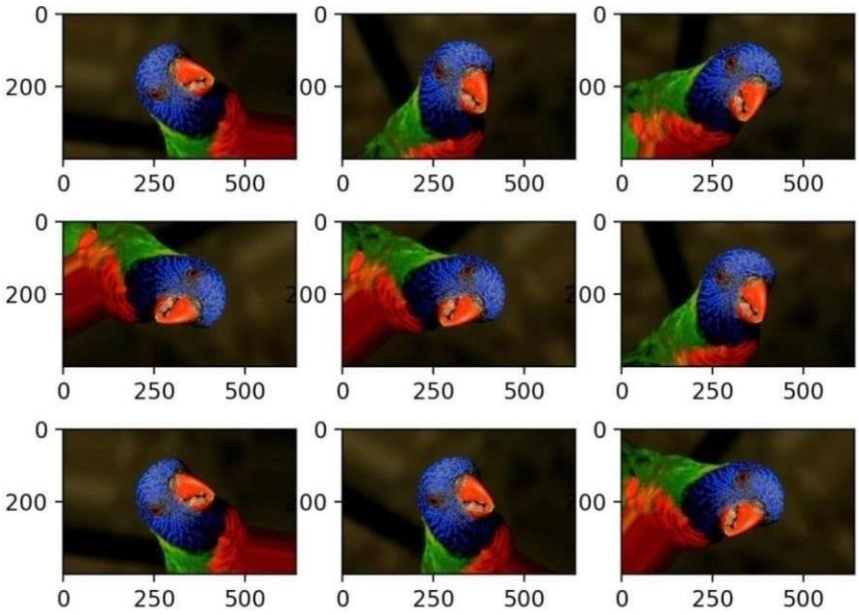
ARTIFICIAL INTELLIGENCE

Date	**14 November 2022**
Team ID	**PNT2022TMID46219**
Project name	**Natural Disasters Intensity Analysis & classification** **Using AI**

CONFIGURE IMAGE DATA GENERATOR CLASS:



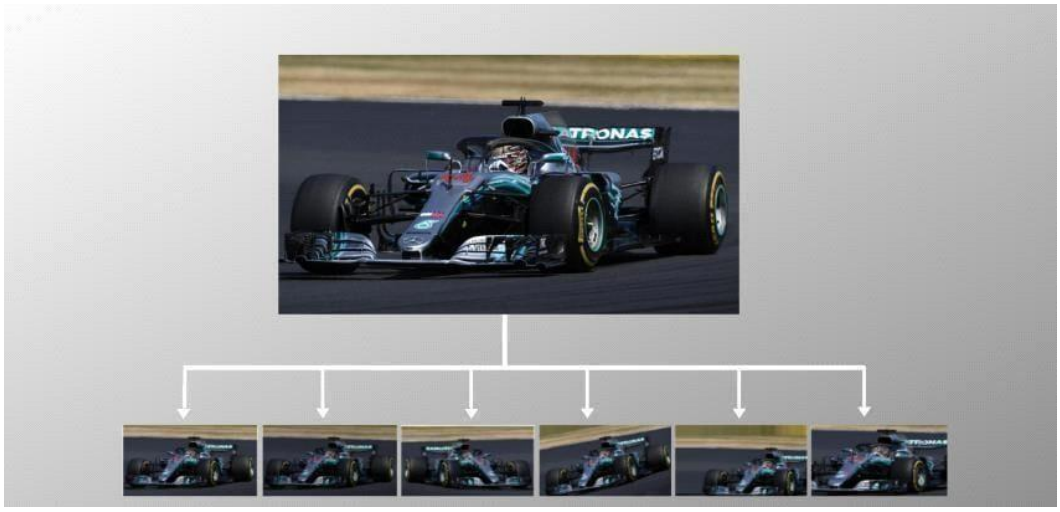
OVERVIEW:-

- **Understand image augmentation**
- **Learn Image Augmentation using**
- **Keras ImageDataGenerator**

INDRODUCTION:-

- **when working with deep learning models, I have often found myself in a peculiar situation when there is not much data to train my model. It was in times like these when I came across the concept of image augmentation.**

- **The image augmentation technique is a great way to expand the size of your dataset. You can come up with new transformed images from your original dataset. But many people use the conservative way of augmenting the images i.e. augmenting images and storing them in a numpy array or in a folder. I have got to admit, I used to do this until I stumbled upon the ImageDataGenerator class**



- **Keras ImageDataGenerator is a gem! It lets you augment your images in real-time while your model is still training!**

AUGMENTATION TECHNIQUES WITH KERAS IMAGE DATA GENERATOR CLASS:-

Random Rotations:

- Image rotation is one of the widely used augmentation techniques and allows the model to become invariant to the orientation of the object.

- ImageDataGenerator class allows you to randomly rotate images through any degree between 0 and 360 by providing an integer value in the rotation_range argument.

```
# ImageDataGenerator rotation datagen =
ImageDataGenerator(rotation_range=30,
fill_mode='nearest')

# iterator
aug_iter = datagen.flow(img, batch_size=1)

# generate samples and plot
fig, ax = plt.subplots(nrows=1, ncols=3, figsize=(15,15))

# generate batch of images for
i in range(3):

 # convert to unsigned integers
 image = next(aug_iter)[0].astype('uint8')

# plot image ax[i].imshow(image)
ax[i].axis('off')
```



Random Brightness:

*It randomly changes the brightness of the image. It is also avery useful augmentation technique because most of the time our object will not be under perfect lighting condition. So, it becomes imperative to train our model on images under different lighting conditions.

*Brightness can be controlled in the ImageDataGenrator class through the brightness_range argument. It accepts a list of two float values and picks a brightness shift value from that range. Values less than 1.0 darkens the image, whereas values above 1.0 brighten the image.

```
# ImageDataGenerator brightness
Datagen = ImageDataGenerator(zoom_range)=0.3

# iterator
aug_iter = datagen.flow(img, batch_size=1)

# generate samples and plot
fig, ax = plt.subplots(nrows=1, ncols=3, figsize=(15,15))

# generate batch of images for
i in range(3):

# convert to unsigned integers image =
next(aug_iter)[0].astype('uint8')
```

Random Zoom:

- **The zoom augmentation either randomly zooms in on the image or zooms out of the image.**

- **ImageDataGenerator class takes in a float value for zooming in the zoom_range argument. You could provide a list with two values specifying the lower and the upper limit. Else, if you specify a float value, then zoom will be done in the range [1-zoom_range,1+zoom_range].**

- **Any value smaller than 1 will zoom in on the image. Whereas any value greater than 1 will zoom out on the image.**

```
# ImageDataGenerator zoom
datagen = ImageDataGenerator(zoom_range=0.3

# iterator
aug_iter = datagen.flow(img, batch_size=1)

# generate samples and plot fig, ax =

plt.subplots(nrows=1, ncols=3, figsize=(15,15))


# generate batch of images for
i in range(3):

# convert to unsigned integers image =

next(aug_iter)[0].astype('uint8'
```