Data Science, Machine Learning Case Study_2

Name- KAUSHIK KAR

EMP ID- 2216027

COHORT CODE- CDB22DW022

Research on various data augmentation techniques and demonstrate the effectiveness of those techniques on a sample image and indicate where data augmentation is necessary and mandatory

Introduction:

What is Data Augmentation and why it is important?

Growing interest of deep learning made convolutional neural network(CNN) the most common tool to use for the image classification and image analysis. In machine learning, image classification is a process to analyse the extracted image features and organize them into categories by using neural networks. But despite wide perspective they must deal with several challenges. The most common problem is the lack of good quality data or uneven class balance within the datasets

Data augmentation is **a process of artificially increasing the amount of data by generating new data points from existing data**. This includes adding minor alterations to data or using machine learning models to generate new data points in the latent space of original data to amplify the dataset.

Data augmentation is a set of techniques that **enhance the size and quality of machine learning training datasets so that better deep learning models can be trained with them**. Data Augmentation artificially inflates datasets using label-preserving data transformations.

One of the ways to dealing with these problems is Data Augmentation. Most popular and proven as Effective current practice for data augmentation is to perform traditional affine and data transformation. Moreover, the current research about adversarial so called attcks on CNN's misclassification of images just by partial rotation and image translation.

Data augmentation techniques generate different versions of a real dataset artificially to increase its size. Computer vision and natural language processing (NLP) models use data augmentation strategy to handle with data scarcity and insufficient data diversity.

Data-centric AI/ML development practices such as data augmentation can increase accuracy of machine learning models. According to an experiment, a deep learning model after image augmentation performs better in training loss (i.e. penalty for a bad prediction) & accuracy and validation loss & accuracy than a deep learning model without augmentation for image classification task.

Data Augmentation techniques:

1. **Adding noise**-it is applied only to images in the training set for strategic variation.
2. **Rotation**-A source image is random rotated clockwise or counter clockwise by some number of degrees, changing the position of the object in frame
3. **Translation**-involves moving the image along the X or Y direction (or both)
4. **Contrast**-Given training images and their super pixels, we first train the neural network in the second stream alone to obtain its initial weights.
5. **Saturation**-depth or intensity of color in an image
6. **Colour augmentation**-randomly alters the color channels of an input image, causing a model to consider alternative color schemes for objects and scenes in input images
7. **Brightness**-can output images with a larger value range than the input
8. **Scaling**- an augmentation technique where we randomly pick the short size of a image within a dimension range
9. **Flipping**-rotating an image in a horizontal or vertical axis
10. **Cropping**-Automatically crop images to optimal regions with deep neural networks.
11. **Channel shift** is used to introduce the color augmentation in the dataset so as to make the model learn color based features irrespective of its saturation value.
12. The **shear** augmentation is a part of the suite of data augmentation options in the Roboflow platform

Importance of data augmentation in deep learning:

Data augmentation is a set of techniques that enhance the size and quality of machine learning training datasets so that better deep learning models can be trained with them. Data Augmentation artificially inflates datasets using label-preserving data transformations

Data Augmentation is the most optimal solution when we face difficulty in meeting the requirement of enough training dataset. Data Augmentation becomes necessary because of the following reasons:

1. Improving model prediction accuracy.
2. Reducing cost of collecting and labelling data.
    a. Adding more training data into the model.
    b. Preventing data scarcity for better models.
    c. Helping resolve class imbalance issues in classification.
    d. Increasing generalization ability of the models.

Taking this picture as an input Image and try out some of the Data Augmentation techniques on it.

Data augmentation techniques in computer vision

1. Importing the libraries and also check the version of the TensorFlow.

```
[2]  %matplotlib inline
     import os
     import numpy as np
     import tensorflow as tf

     from PIL import Image
     from matplotlib import pyplot as plt

     print('Using TensorFlow', tf.__version__)
     from keras.preprocessing.image import ImageDataGenerator
```

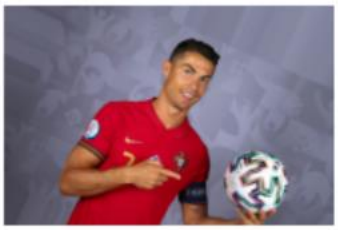
```
Using TensorFlow 2.9.2
```

2. The image is rotated by a degree between 0 and 360 degrees. Every rotated image will be unique in the model. By applying **Rotation,** we can see the picture tilt in left and right side also but there is no change in the image.

```
data_gen = tf.keras.preprocessing.image.ImageDataGenerator(
        rotation_range=40)

img = tf.keras.utils.load_img('CR7.jpg')
x = tf.keras.utils.img_to_array(img)
x = x.reshape((1,) + x.shape)
i = 0
fig, ax = plt.subplots(nrows=1, ncols=3, figsize=(10,8))
aug_iter=data_gen.flow(x, batch_size=1,save_to_dir="/content", save_prefix='CR7', save_format='jpeg')
# 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')
```


3. By applying **Width and height** we can see that the pictures may shifts their width and height but there is no change in the image.

```
data_gen = tf.keras.preprocessing.image.ImageDataGenerator(
    width_shift_range = [-100, -50, 0, 50, 100],
    height_shift_range = [-50, 0, 50]
)
img = tf.keras.utils.load_img('CR7.jpg')
x = tf.keras.utils.img_to_array(img)
x = x.reshape((1,) + x.shape)
i = 0
fig, ax = plt.subplots(nrows=1, ncols=3, figsize=(15,15))
aug_iter=data_gen.flow(x, batch_size=1,save_to_dir='/content', save_prefix='CR7', save_format='jpeg')
# 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')
```



4. By applying **zoom** we can see the same picture with large and small sizes.

```
data_gen = tf.keras.preprocessing.image.ImageDataGenerator(
    zoom_range = 0.5
)

img = tf.keras.utils.load_img('CR7.jpg')
x = tf.keras.utils.img_to_array(img)
x = x.reshape((1,) + x.shape)
i = 0
fig, ax = plt.subplots(nrows=1, ncols=3, figsize=(15,15))
aug_iter=data_gen.flow(x, batch_size=1,save_to_dir='/content', save_prefix='cat', save_format='jpeg')
# 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')
```

5. The **brightness** of the image is changed, and new image will be darker or lighter. This technique allows the model to recognize image in different lighting levels.

```
data_gen = tf.keras.preprocessing.image.ImageDataGenerator(
    brightness_range = (0.5, 2)
)
img = tf.keras.utils.load_img('CR7.jpg')
x = tf.keras.utils.img_to_array(img)
x = x.reshape((1,) + x.shape)
i = 0
fig, ax = plt.subplots(nrows=1, ncols=3, figsize=(15,15))
aug_iter=data_gen.flow(x, batch_size=1,save_to_dir='/content', save_prefix='cat', save_format='jpeg')
# 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')
```

6. The image is **flipped** horizontally and vertically. Flipping rearranges the pixels while protecting the features of the image. Vertical flipping is not meaningful for some photos, but it can be useful in cosmology or for microscopic photos.

```
data_gen = tf.keras.preprocessing.image.ImageDataGenerator(
    horizontal_flip = True,
    vertical_flip = True,
    rotation_range = 30
)
img = tf.keras.utils.load_img('CR7.jpg')
x = tf.keras.utils.img_to_array(img)
x = x.reshape((1,) + x.shape)
i = 0
fig, ax = plt.subplots(nrows=1, ncols=3, figsize=(15,15))
aug_iter=data_gen.flow(x, batch_size=1,save_to_dir='/content', save_prefix='cat', save_format='jpeg')
# 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')
```

7. **Channel shift** changes the color saturation level(eg. light Red/dark red) of pixels by changing the [R,G,B] channels of the input image.

```
datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    channel_shift_range = 100
)
img = tf.keras.utils.load_img('CR7.jpg')
x = tf.keras.utils.img_to_array(img)
x = x.reshape((1,) + x.shape)
i = 0
fig, ax = plt.subplots(nrows=1, ncols=3, figsize=(15,15))
aug_iter=datagen.flow(x, batch_size=1,save_to_dir='/content', save_prefix='cat', save_format='jpeg')
# 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')
```

8. **Shear tool** is used to shift one part of an image, a layer, a selection or a path to a direction and the other part to the opposite direction.

```
data_gen = tf.keras.preprocessing.image.ImageDataGenerator(
    # Your code here
    shear_range=40
)
img = tf.keras.utils.load_img('CR7.jpg')
x = tf.keras.utils.img_to_array(img)
x = x.reshape((1,) + x.shape)
i = 0
fig, ax = plt.subplots(nrows=1, ncols=3, figsize=(15,15))
aug_iter=data_gen.flow(x, batch_size=1,save_to_dir='/content', save_prefix='cat', save_format='jpeg')
# 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')
```



Conclusion:

Image augmentation is a technique that is used **to artificially expand the data-set**. This is helpful when we are given a data-set with very few data samples. In case of Deep Learning, this situation is bad as the model tends to over-fit when we train it on limited number of data samples.