**Augmentation**

The Data augmentation is a technique of artificially increasing the training set by creating modified copies of a dataset using existing data. it includes making minor change to the dataset or using deep learning to generate new data points.

When should use data Augmentation ?

* 1] To prevent models from overfitting.
* 2] The initial training set is too small.
* 3] To improve the model accuracy.
* 4] To reduce the operational cost of labeling and cleaning the raw dataset.

And also we can use Augmentation for

* 1. Text data augmentation
  2. Image Augmentation
  3. Audio Data Augmentation

A} Text Data Augmentation :-

- Word or sentence shuffling: randomly changing the position of a word of sentence.

- word replacement : replace words with synonyms.

- syntax-tree manipulation : paraphrase the sentence using the same word.

- Random word insertion : inserts words at random

- Random word deletion : deletes words at random

B} Image Augmentation :-

- Geometric transformations : randomly flip, crop rotate, stretch and zoom images. you need to be careful bout applying multiple transformations on the same images a this can reduce model performance.

- Color space transformation : randomly change RGB color channels, contrast, and brightness.

- Kernel filters : randomly change the sharpness or blurring of the image.

- random erasing : delete some part of the initial image.

C} Audio Data Augmentation :-

- Noise injection: add gaussian or random noise to the audio dataset to improve the model performance.

- Shifting: shift audio left (fast forward) or right with random seconds.

- Changing the speed: stretches times series by a fixed rate.

- Changing the pitch: randomly change the pitch of the audio.

:-> Some important usefull libraries we can use in Augmentation

1] Augmentor

2] Albumentations

3] Imgaug

4] AutoAugment (DeepAugment)

1] Augmentor:- pip install Augmentor

Moving on to the libraries, Augmentor is a Python package that aims to be both a data augmentation tool and a library of basic image pre-processing functions.

In general, Augmentor consists of a number of classes for standard image transformation functions, such as Crop, Rotate, Flip, and many more.

The main features of Augmentor package are:

Perspective skewing – look at an image from a different angle

Elastic distortions – add distortions to an image

Rotating – simply, rotate an image

Shearing – tilt an image along with one of its sides

Cropping – crop an image

Mirroring – apply different types of flips

2] Albumentations :- pip install albumentation

Albumentations is a computer vision tool designed to perform fast and flexible image augmentations. It appears to have the largest set of transformation functions of all image augmentation libraries.

Albumentations is a fast and flexible Python tool for image augmentation. It is widely used in machine learning competitions, industry, and research to improve the performance of deep convolutional neural networks.

3] Imgaug

Imgaug is an open-source tool for image augmentation. It supports a wide variety of augmentation techniques, such as Gaussian noise, contrast, sharpness, crop, affine, and flip. It has a simple yet powerful stochastic interface, and it comes with keypoints, bounding boxes, heatmaps, and segmentation maps

Images :- We will be using TensorFlow and Keras for data augmentation and matplotlib for displaying the images.

%%capture

import matplotlib.pyplot as plt

import numpy as np

import tensorflow as tf

import tensorflow\_datasets as tfds

from keras import layers

import keras

**Text augmentation** :- Text augmentation is a technique used to increase the amount of training data for natural language processing models by generating new text data from existing text. This can be helpful in tasks like text classification, sentiment analysis, machine translation, and more, where having more diverse training data can lead to better model performance.

One common text augmentation technique is to apply various transformations to the text, such as synonym replacement, back translation, paraphrasing, or even introducing noise, to create new examples that are still semantically similar to the original text.

example of text augmentation using Python's nlpaug library for synonym replacement:

import nlpaug.augmenter.word as naw

# Example sentence

text = "Text augmentation is a useful technique for NLP tasks."

# Initialize the augmenter for word-level replacement

aug = naw.SynonymAug(aug\_src='wordnet')

# Augment the text

augmented\_text = aug.augment(text)

print("Original Text:", text)

print("Augmented Text:", augmented\_text)