Data Preprocessing and Data Augmentation (Synthetic Oversampling)

Data augmentation is a technique used in machine learning and computer vision to artificially increase the diversity and quantity of your training data by applying various transformations to the original data. Data augmentation is particularly useful in deep learning, where having a large and diverse dataset can lead to more robust models and better generalization.

For image classification tasks, data augmentation techniques are often applied to image data to create new training examples with slight variations. Common augmentations include:

- **Rotation**: Rotating images by a certain degree (e.g., 90 degrees, 180 degrees) to simulate different angles of view.
- **Horizontal**: and Vertical Flips: Mirroring images horizontally or vertically to simulate different orientations or perspectives.
- **Scaling**: Resizing images to different scales to simulate objects at varying distances or sizes.
- **Translation**: Shifting the position of the image content within the frame to simulate different positions.
- **Brightness and Contrast Adjustments**: Altering the brightness and contrast levels of images to simulate varying lighting conditions.
- **Color Jittering**: Slightly changing the color and tone of images to simulate color variations.
- **Noise Addition**: Adding random noise to the images to simulate real-world noise.
- **Cropping**: Extracting random crops from images to focus on different parts of the image.

Data augmentation helps to:

- **Reduce overfitting**: By presenting the model with a more diverse set of examples, it learns to be more robust and generalizes better to unseen data.
- **Improve model performance**: Augmented data can enhance the model's ability to recognize variations in the data, making it more accurate in real-world scenarios.

```
import matplotlib.pyplot as plt
from tensorflow import keras
from keras.preprocessing.image import  ImageDataGenerator # Package for Data Augmentation
from keras.utils import img_to_array, array_to_img, load_img
```

```
image_file ='/content/car.jpg'
my_image = load_img( image_file)
my_image
```

```
my_img_array =img_to_array(my_image)
my_img_array
```

```
array([[[ 96., 104., 123.],
        [ 96., 104., 123.],
        [ 96., 104., 123.],
        ...,
        [151., 159., 170.],
        [151., 159., 170.],
        [152., 160., 171.]],

       [[ 96., 104., 123.],
        [ 96., 104., 123.],
        [ 97., 105., 124.],
        ...,
        [151., 159., 170.],
        [151., 159., 170.],
        [151., 159., 170.]],
```

```
           [[ 95., 103., 122.],
            [ 96., 104., 123.],
            [ 97., 105., 124.],
            ...,
            [151., 159., 170.],
            [151., 159., 170.],
            [151., 159., 170.]],

           ...,

           [[100.,  88.,  72.],
            [ 97.,  88.,  71.],
            [104.,  92.,  76.],
            ...,
            [108.,  97.,  79.],
            [109.,  98.,  80.],
            [107.,  95.,  79.]],

           [[100.,  88.,  72.],
            [ 97.,  88.,  71.],
            [ 97.,  85.,  69.],
            ...,
            [107.,  96.,  78.],
            [107.,  96.,  78.],
            [107.,  96.,  78.]],

           [[101.,  89.,  73.],
            [ 94.,  85.,  68.],
            [ 98.,  86.,  70.],
            ...,
            [107.,  96.,  78.],
            [107.,  96.,  78.],
            [107.,  96.,  78.]]], dtype=float32)
```

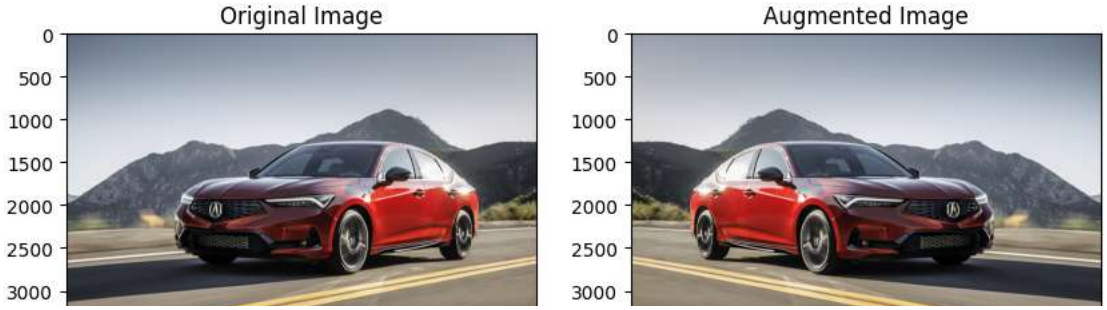
```
my_img_array.shape
```

```
(3668, 5500, 3)
```

```
def plot_image (original_img, augmented_img):
  plt.figure(figsize =(10,4))
  # Original
  plt.subplot(1,2,1)
  plt.title('Original Image')
  plt.imshow(original_img)
  # Augmented
  plt.subplot(1,2,2)
  plt.title('Augmented Image')
  plt.imshow(augmented_img)
plt.show()
```

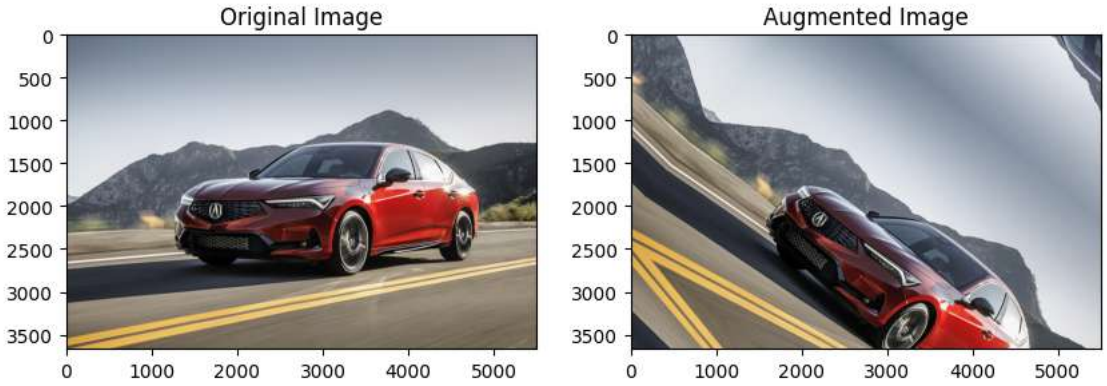
Horizontal Flip

```
augmented_data = keras.preprocessing.image.ImageDataGenerator(horizontal_flip= True)
augmented_array = augmented_data.random_transform(my_img_array)
augmented_img = array_to_img(augmented_array)
plot_image(my_image,augmented_img)
```

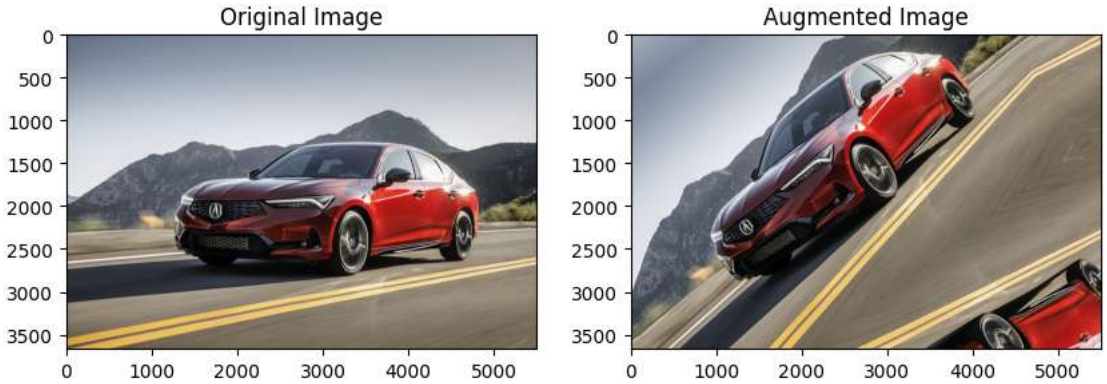


Rotation

```
updated_data = keras.preprocessing.image.ImageDataGenerator(rotation_range=45, fill_mode='reflect')
updated_array = updated_data.random_transform(my_img_array)
updated_img =array_to_img(updated_array)
plot_image(my_image, updated_img)
```

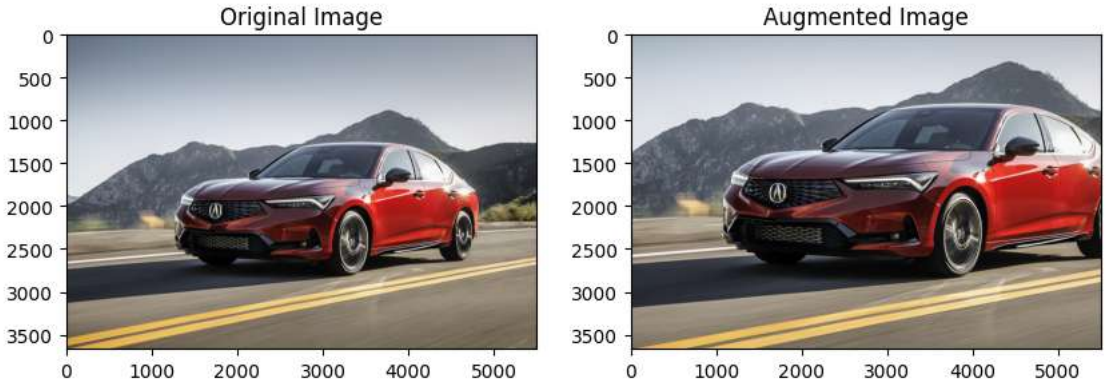



```
updated_data = keras.preprocessing.image.ImageDataGenerator(rotation_range=150, fill_mode='reflect')
updated_array = updated_data.random_transform(my_img_array)
updated_img =array_to_img(updated_array)
plot_image(my_image, updated_img)
```



Enlarge

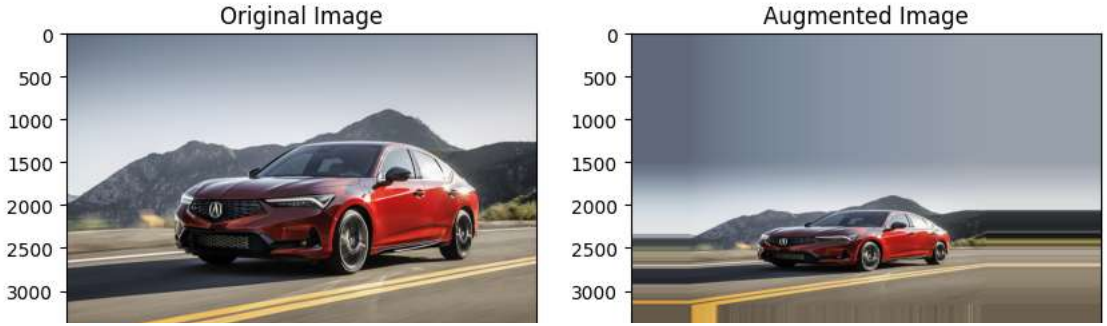
```
updated_data = keras.preprocessing.image.ImageDataGenerator(zoom_range=[0.8,0.7])
updated_array = updated_data.random_transform(my_img_array)
updated_img = array_to_img(updated_array)
plot_image(my_image,updated_img)
```



Zoom-out

```
updated_data = keras.preprocessing.image.ImageDataGenerator(zoom_range=[0.8,1.4])
updated_array = updated_data.random_transform(my_img_array)
updated_img = array_to_img(updated_array)
plot_image(my_image,updated_img)
```


```
updated_data = keras.preprocessing.image.ImageDataGenerator(zoom_range=[0.9,2.5], fill_mode= 'nearest')
updated_array = updated_data.random_transform(my_img_array)
updated_img = array_to_img(updated_array)
plot_image(my_image,updated_img)
```



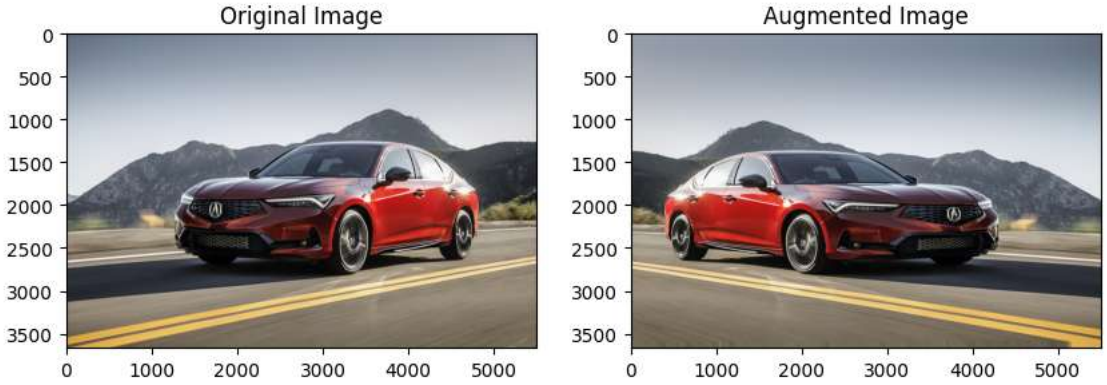
Enlarge and flip

```
updated_data = keras.preprocessing.image.ImageDataGenerator(horizontal_flip = True, zoom_range=[0.8,0.6])
updated_array = updated_data.random_transform(my_img_array)
updated_img = array_to_img(updated_array)
plot_image(my_image,updated_img)
```

Reduce and Flip

```
updated_data = keras.preprocessing.image.ImageDataGenerator(horizontal_flip = True,zoom_range=[0.9,3],fill_mode='nearest')
updated_array = updated_data.random_transform(my_img_array)
updated_img = array_to_img(updated_array)
plot_image(my_image,updated_img)
```


```
updated_data = keras.preprocessing.image.ImageDataGenerator(horizontal_flip = True,zoom_range=[0.9,2],fill_mode='nearest')
updated_array = updated_data.random_transform(my_img_array)
updated_img = array_to_img(updated_array)
plot_image(my_image,updated_img)
```




```
datagen = ImageDataGenerator(
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
x = img_to_array(my_image)
x = x.reshape((1,) +x.shape)
i=0
for batch in datagen.flow(x, batch_size=1, save_to_dir ='/content/img',
                          save_prefix ='car', save_format='jpeg'):
  i += 1
  if i >20:
    break
```

```
from PIL import Image
import os

image_folder_path = '/content/img'
os.chdir(image_folder_path)
image_files = os.listdir()
image_files = [file for file in image_files if file.endswith(('.jpg', '.jpeg', '.png'))]
images = [Image.open(image) for image in image_files]
```

```
len(images)
```

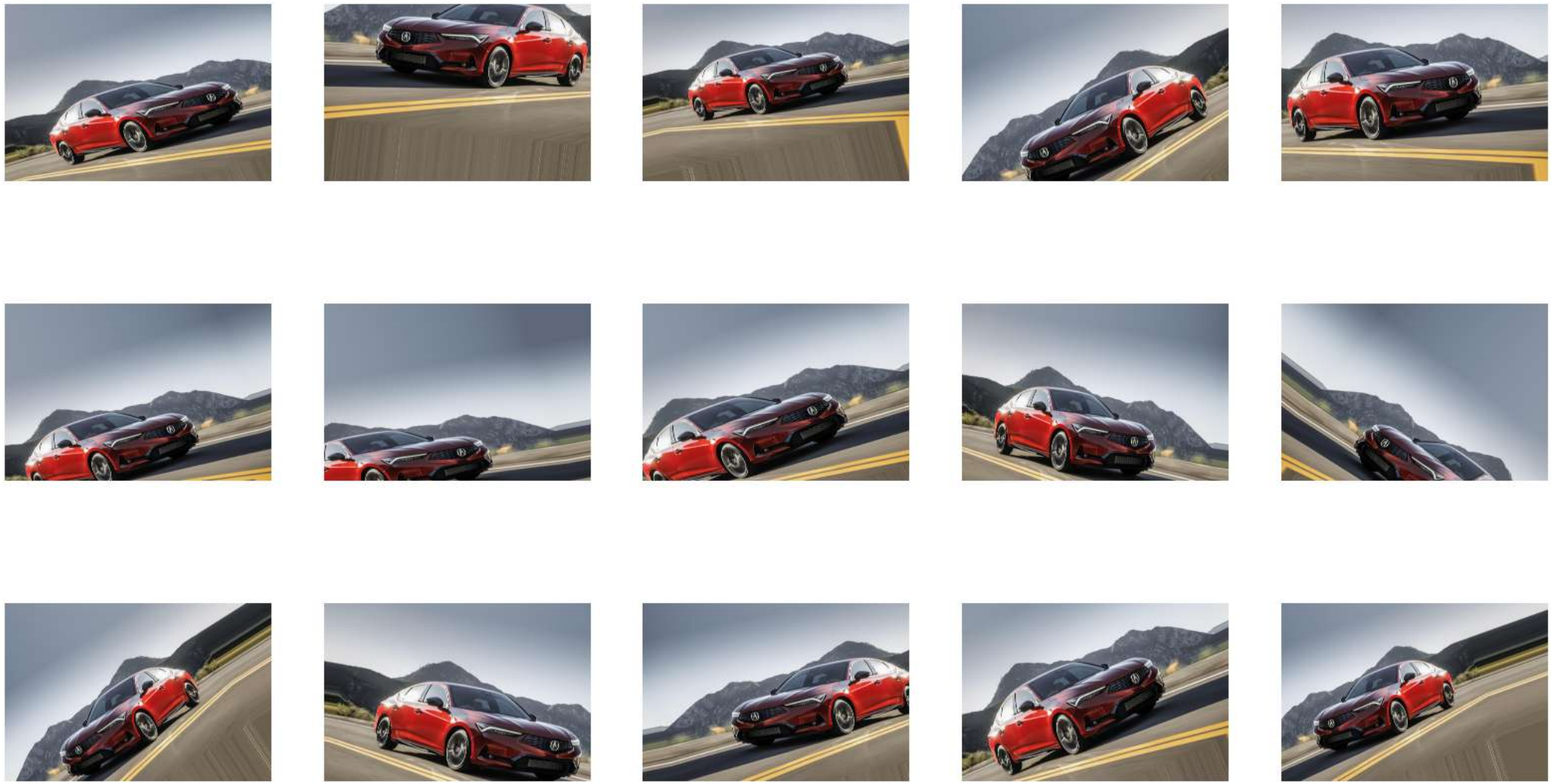
```
21
```

```
fig, axes = plt.subplots(4, 5, figsize=(20, 15))
```

```
for idx in range(1, len(images)):
    if idx <= len(images):
        plt.subplot(4, 5, idx)
        plt.imshow(images[idx - 1])
        plt.axis('off')
    else:

        plt.subplot(4, 5, idx)
        plt.axis('off')

plt.show()
```



Deep Learning requires abilty to auto learn features from data, which is only possible when lots of training data is avaialable

```
image_file ='/content/cat.jpg'
img1 = load_img( image_file)
```

```
img1
```

```
img1_array =img_to_array(img1)
img1_array.shape
```

```
(435, 580, 3)
```

```
datagen = ImageDataGenerator(
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

x = img_to_array(img1)
x = x.reshape((1,) +x.shape)
i=0
for batch in datagen.flow(x, batch_size=1, save_to_dir ='/content/cat',
                          save_prefix ='cat', save_format='jpeg'):
  i += 1
  if i >20:
    break
```

```
from PIL import Image
import os

image_folder_path = '/content/cat'
os.chdir(image_folder_path)
image_files = os.listdir()
image_files = [file for file in image_files if file.endswith(('.jpg', '.jpeg', '.png'))]
cat_images = [Image.open(image) for image in image_files]
```

```
fig, axes = plt.subplots(4, 5, figsize=(20, 15))

for idx in range(1, len(cat_images)):
    if idx <= len(cat_images):
        plt.subplot(4, 5, idx)
        plt.imshow(cat_images[idx - 1])
        plt.axis('off')
    else:

        plt.subplot(4, 5, idx)
        plt.axis('off')

plt.show()
```