

Image Data Augmentation

Lecture 02

Prepared by

Dr. Hager Saleh

Image data augmentation

- Image data augmentation is used to expand the training dataset in order to improve the performance and ability of the model to generalize.
- Image data augmentation is supported in the Keras deep learning library via the ImageDataGenerator class.

What are different types of data augmentation methods?

- Horizontal and Vertical Shift Augmentation: This augmentation shifts the image along the x-axis (horizontal) and y-axis (vertical).
- Horizontal and Vertical Flip Augmentation: Flips the image randomly along the horizontal or vertical axis.
- Random Rotation Augmentation: Rotates the image by a random angle within the specified range.
- Random Brightness Augmentation: Adjusts the brightness of the image randomly.
- Random Zoom Augmentation: Zooms into or out of the image randomly.

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import load_img,
```

```

img_to_array

def display_augmented_images(img_path, datagen, num_augmented=1):

    # Load and prepare the image
    img = load_img(img_path)
    img_array = img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)

    # Generate augmented images
    augmented_images = datagen.flow(img_array, batch_size=1)

    # Display original and augmented images
    plt.figure(figsize=(15, 6))

    # Show original image
    plt.subplot(1, num_augmented + 1, 1)
    plt.title('Original')
    plt.imshow(img)
    plt.axis('off')

    # Generate and display augmented images
    for i in range(num_augmented):
        batch = next(augmented_images)
        plt.subplot(1, num_augmented + 1, i + 2)
        plt.title(f'Aug {i+1}')
        plt.imshow(batch[0].astype('uint8'))
        plt.axis('off')

    plt.tight_layout()
    plt.show()

```

Horizontal and Vertical Shift Augmentation

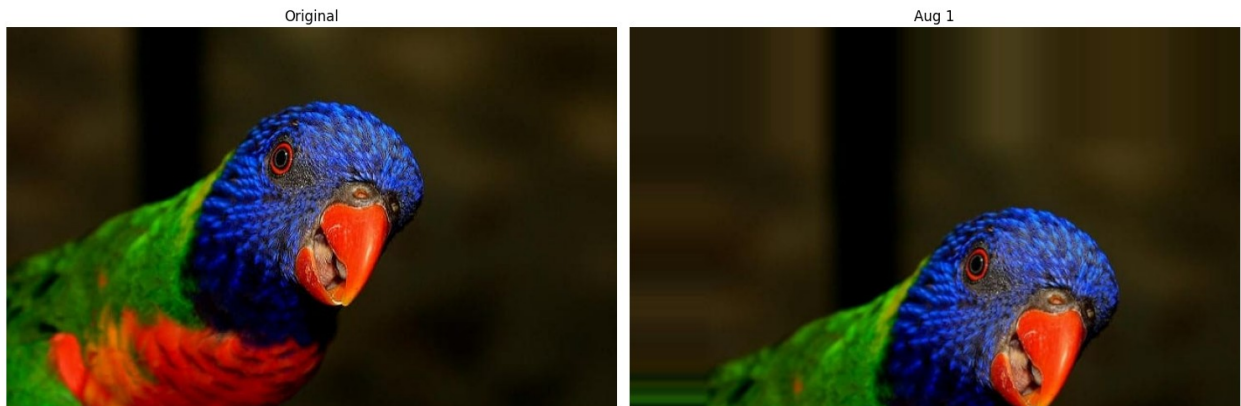
A shift to an image means moving all pixels of the image in one direction, such as horizontally or vertically, while keeping the image dimensions the same.

```

# Define the augmentation pipeline
datagen = ImageDataGenerator(
    width_shift_range=0.4,    # Horizontal shift
    height_shift_range=0.2,   # Vertical shift
)

# Call the function
display_augmented_images('bird.jpg', datagen)

```

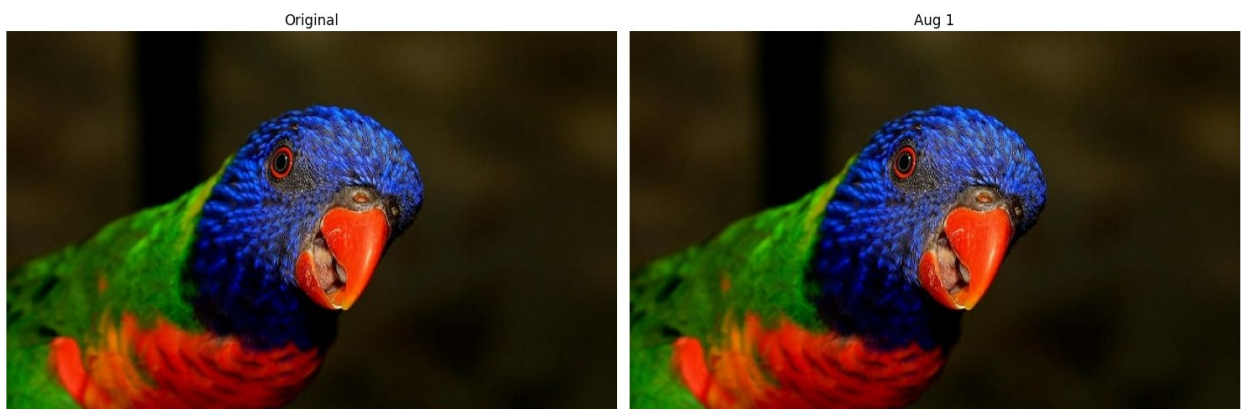


Horizontal and Vertical Flip Augmentation

An image flip means reversing the rows or columns of pixels in the case of a vertical or horizontal flip respectively.

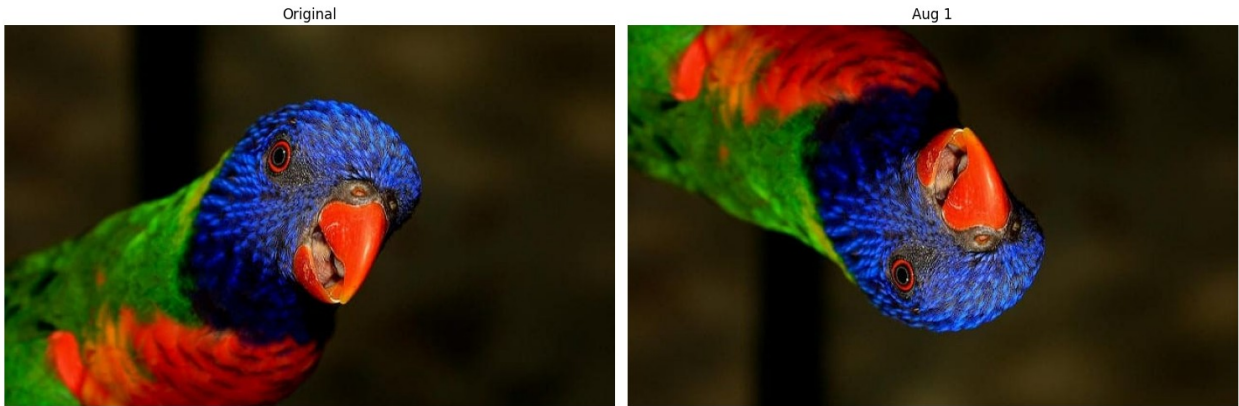
horizontal_flip

```
datagen = ImageDataGenerator(  
    horizontal_flip=True,    # Horizontal flip  
)  
display_augmented_images('bird.jpg', datagen)
```



vertical_flip

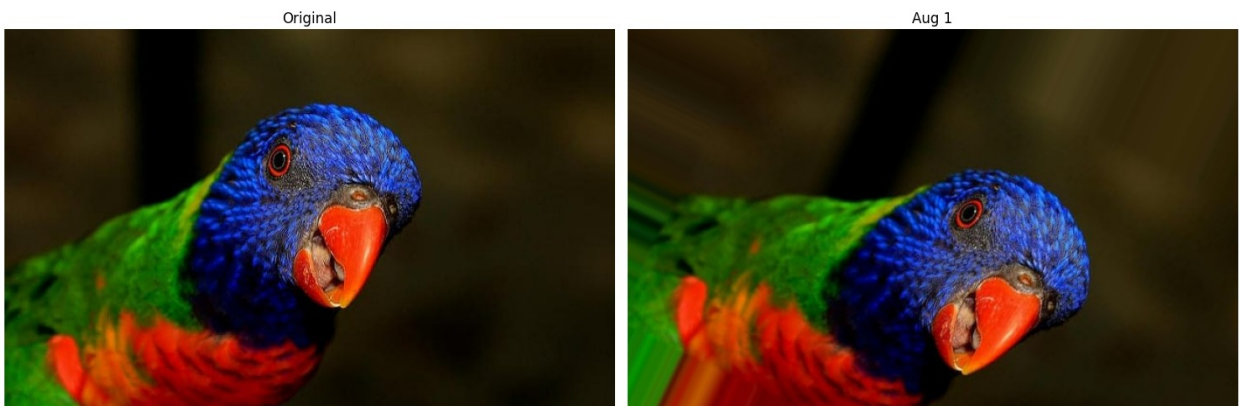
```
datagen = ImageDataGenerator(  
    vertical_flip=True,    # Horizontal flip  
)  
  
display_augmented_images('bird.jpg', datagen)
```



rotation augmentation randomly

A rotation augmentation randomly rotates the image clockwise by a given number of degrees from 0 to 360.

```
datagen = ImageDataGenerator(  
    rotation_range=40,      # Random rotation  
)  
display_augmented_images('bird.jpg', datagen)
```



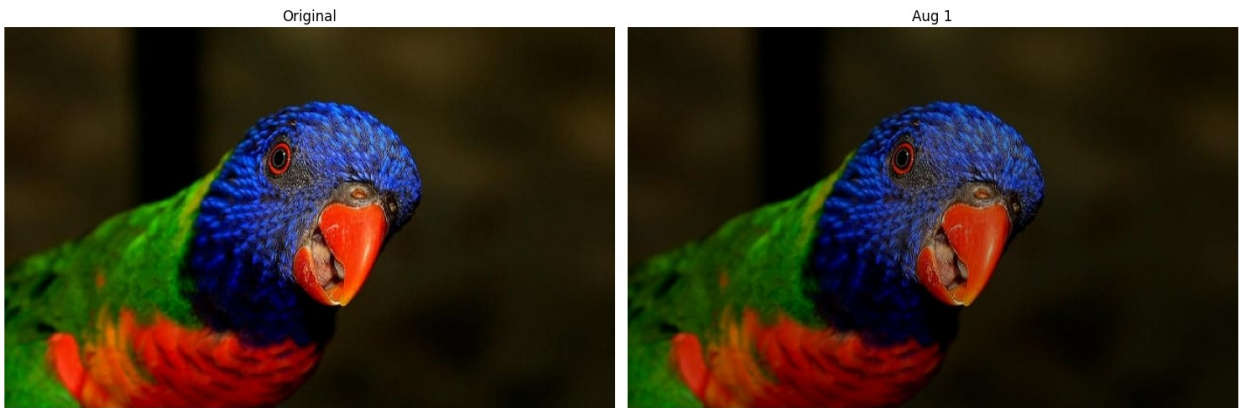
Random Brightness Augmentation

The brightness of the image can be augmented by either randomly darkening images, brightening images, or both.

The intent is to allow a model to generalize across images trained on different lighting levels.

```
datagen = ImageDataGenerator(  
    brightness_range=[0.5, 1.5], # Random brightness
```

```
)  
display_augmented_images('bird.jpg', datagen)
```



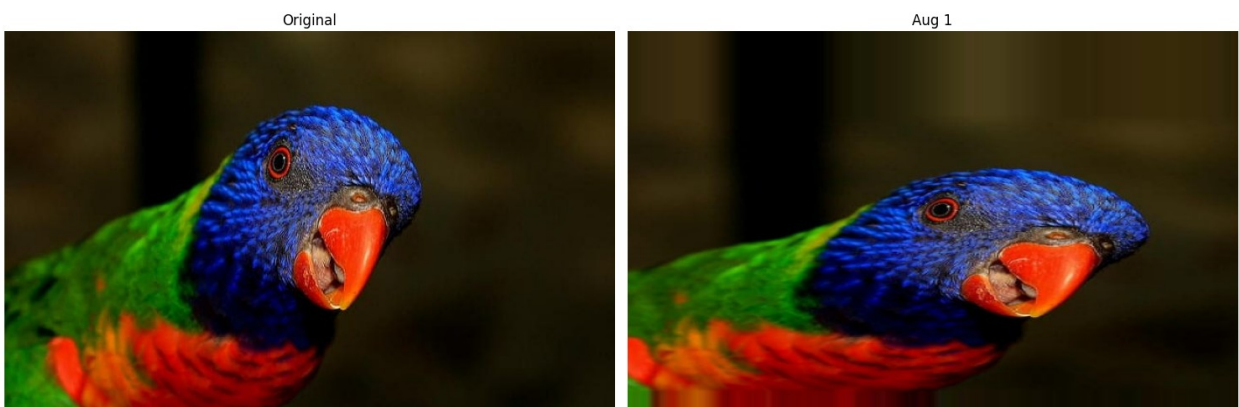
Random Zoom Augmentation

A zoom augmentation randomly zooms the image in and either adds new pixel values around the image or interpolates pixel values respectively.

Image zooming can be configured by the `zoom_range` argument to the `ImageDataGenerator` constructor. You can specify the percentage of the zoom as a single float or a range as an array or tuple.

If a float is specified, then the range for the zoom will be $[1-\text{value}, 1+\text{value}]$. For example, if you specify 0.3, then the range will be $[0.7, 1.3]$, or between 70% (zoom in) and 130% (zoom out).

```
datagen = ImageDataGenerator(  
    zoom_range=0.6,           # Random zoom  
)  
display_augmented_images('bird.jpg', datagen)
```



How Download data from Kaggle

Create a Kaggle Account (if you don't have one): Go to <https://www.kaggle.com> and sign up/log in.

Go this link: <https://www.kaggle.com/datasets/adityamahimkar/iqothnccd-lung-cancer-dataset/versions/1> and download the dataset

This dataset is about lung cancer and consists of three classes: Benign, Malignant, and Normal.

Splitting dataset

We split the dataset into different folders: training set, validation set, and testing set. Each folder contains subfolders named after the respective classes.

```
import os
import shutil
import numpy as np
from sklearn.model_selection import train_test_split

import shutil

original_dataset_dir = 'The IQ-0THNCCD lung cancer dataset' # Path to
the dataset with subfolders
base_dir = 'dataset' # Where you want to save new directories

train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'validation')
test_dir = os.path.join(base_dir, 'test')

for dir in [train_dir, validation_dir, test_dir]:
    os.makedirs(dir, exist_ok=True)

# Split ratio
train_ratio = 0.7
validation_ratio = 0.1
test_ratio = 0.2

# Function to split and copy files
def split_and_copy_files(class_dir, dest_base_dir):
    all_images = [os.path.join(class_dir, f) for f in
os.listdir(class_dir) if os.path.isfile(os.path.join(class_dir, f))]
    train_images, temp_images = train_test_split(all_images,
test_size=(1 - train_ratio), random_state=42)
    validation_images, test_images = train_test_split(temp_images,
test_size=test_ratio/(validation_ratio + test_ratio), random_state=42)

    for images, folder_name in zip([train_images, validation_images,
test_images], ['train', 'validation', 'test']):
        dest_dir = os.path.join(dest_base_dir, folder_name,
```

```

os.path.basename(class_dir))
    os.makedirs(dest_dir, exist_ok=True)
    for file in images:
        shutil.copy(file, dest_dir)

# Iterate over each class directory and split the files
for class_dir in os.listdir(original_dataset_dir):
    full_class_dir = os.path.join(original_dataset_dir, class_dir)
    if os.path.isdir(full_class_dir):
        split_and_copy_files(full_class_dir, base_dir)

def create_data_generators(train_dir, val_dir, test_dir,
img_size=(240, 240), batch_size=32):

    # Data augmentation for training dataset
    train_datagen = ImageDataGenerator(

        rotation_range=40,          # Randomly rotate images by up
to 40 degrees
        width_shift_range=0.2,      # Random horizontal shift
        height_shift_range=0.2,    # Random vertical shift

        zoom_range=0.2,            # Zoom in/out
        horizontal_flip=True,      # Horizontal flip
        brightness_range=[0.5, 1.5], # Vary brightness

    )

    # Only rescale for validation and testing
    val_test_datagen = ImageDataGenerator(rescale=1./255)

    # Generate augmented training images
    train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=img_size,
        batch_size=batch_size,
        class_mode='categorical',  # Use 'binary' for binary
classification
        shuffle=True
    )

    # Generate validation images
    val_generator = val_test_datagen.flow_from_directory(
        val_dir,
        target_size=img_size,
        batch_size=batch_size,
        class_mode='categorical',
        shuffle=False

```

```

    )

    # Generate test images
    test_generator = val_test_datagen.flow_from_directory(
        test_dir,
        target_size=img_size,
        batch_size=batch_size,
        class_mode='categorical',
        shuffle=False
    )

    return train_generator, val_generator, test_generator

# Directories for train, validation, and test datasets
train_dir = 'dataset/train'
val_dir = 'dataset/validation'
test_dir = 'dataset/test'

# Create data generators
train_gen, val_gen, test_gen = create_data_generators(train_dir,
val_dir, test_dir)

Found 766 images belonging to 3 classes.
Found 164 images belonging to 3 classes.
Found 222 images belonging to 3 classes.

```