CLOUD APPLICATION DEVELOPMENT

PROJECT: Image Recognition with IBM Cloud Visual Recognition

Phase 3 Submission document

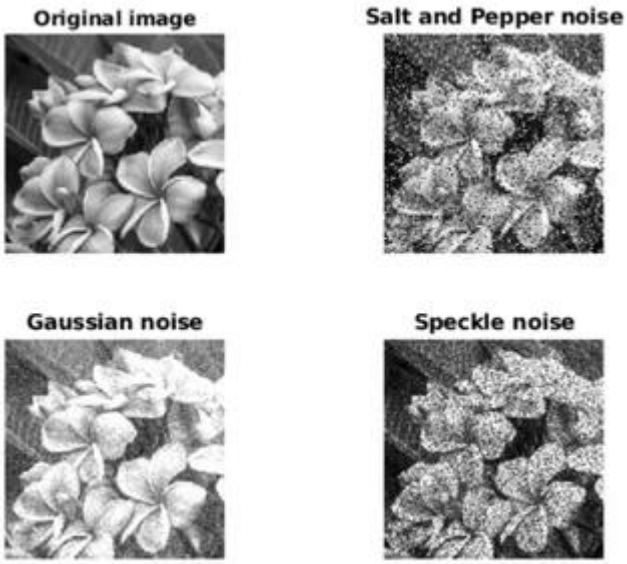
INTRODUCTION

Image recognition is a branch of computer vision that uses various algorithms to manipulate and analyze digital images. It involves the use of mathematical or statistical operations to modify images for many applications, including and not limited to medical and satellite imagery and digital photography. This article explores the fundamentals of image processing and some of the techniques used in this field.

Fundamentals of Image Processing

Digital images are broadly made up of pixels, which are tiny boxes representing the color and brightness values at that point in the image. Image processing involves handling these pixels in a desired manner to achieve what is required for the image. Most of the common operations performed on a digital image include filtering, enhancement, restoration, etc.

Filtering is a process of eliminating unwanted noise from an image. It is done by applying a filter that adjusts the image's pixel values. Based on the type of filter, they can be used for a wide range of applications. They can be designed to remove specific types of noise, such as Gaussian noise, salt-and-pepper noise, or speckle noise. The filters that help in removing the above-mentioned noises include the median filter, the mean filter, and the Gaussian filter.



Enhancement is one process that can improve the quality of an image. It is done by modifying the brightness or contrast of the image. These techniques may be simple, like adjusting the brightness and contrast using a histogram, or more complex, like using algorithms to enhance the edges and textures in an image.

Restoration is the process of recovering an image that some noise or other artifacts may degrade. The techniques involve using mathematical methods to estimate the original image from the corrupted version. It is done using techniques such as deconvolution, which is used to get the original image

from a blurred version, or denoising, which is used to remove noise from an image.

Image preprocessing is quite useful to improve the quality of images and thus boost them for analysis and further processing. Some powerful image preprocessing techniques include noise reduction, contrast enhancement, image resizing, color correction, segmentation, feature extraction, etc. It is an essential step in image analysis that helps enhance the data in images and reduce clutter. As technology continues to advance, image processing will likely become even more important in our daily lives.

Applications of Image Processing

Image preprocessing is a vital step when working with image data. The best results can be obtained when preprocessing of images is done according to the application involved. It is used in various domains, as listed below:

- Medical Imaging to improve the quality of medical images, making it easier to detect diseases or abnormalities

- Object Recognition in images, such as recognizing faces or license plates in surveillance videos
- Object Detection, i.e., primarily used in self-driving cars to navigate the roads better and avoid accidents
- Satellite imagery uses the same for enhancing the image quality for weather forecasting, mapping, etc

Techniques for Image Preprocessing

The choice of techniques depends on the nature of the image and the application. Here are a few techniques to improve image quality and suitability:

- **Noise Reduction**: Noise in an image can be caused by various factors such as low light, sensor noise, and compression artifacts. Noise reduction techniques aim to remove noise from the image while preserving its essential features. Some common noise reduction techniques include Gaussian smoothing, median filtering, and wavelet denoising.
- **Contrast Enhancement**: Contrast enhancement techniques aim to increase the contrast of an image, making it easier to distinguish between different image features. These techniques can be helpful in applications such as medical imaging and surveillance. Some standard contrast enhancement techniques include histogram equalization, adaptive histogram equalization, and contrast stretching.
- **Image Resizing**: Image resizing techniques are used to adjust the size of an image. Resizing can be done to make an image smaller or larger or to change its aspect ratio. Some typical image resizing techniques

include nearest neighbor interpolation, bilinear interpolation, and bicubic interpolation.

- **Color Correction**: Color correction techniques are used to adjust the color balance of an image. Color correction is important in applications such as photography, where the color accuracy of an image is critical. Some common color correction techniques include gray world assumption, white balance, and color transfer.
- **Segmentation**: Segmentation techniques are used to divide an image into regions based on its content. Segmentation can be helpful in applications such as medical imaging, where specific structures or organs must be isolated from the image. Some standard segmentation techniques include thresholding, edge detection, and region growing.
- **Feature Extraction**: Feature extraction techniques are used to identify and extract relevant features from an image. These features can be used in object recognition and image classification applications. Some standard feature extraction techniques include edge detection, corner detection, and texture analysis.

1. Install Dependencies and Setup

```
!pip install tensorflow tensorflow-gpu opencv-python matplotlib
!pip list
import tensorflow as tf
import os
# Avoid OOM errors by setting GPU Memory Consumption Growth
gpus = tf.config.experimental.list_physical_devices('GPU')
for gpu in gpus:
    tf.config.experimental.set_memory_growth(gpu, True)
```

```
tf.config.list_physical_devices('GPU')
```

```
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

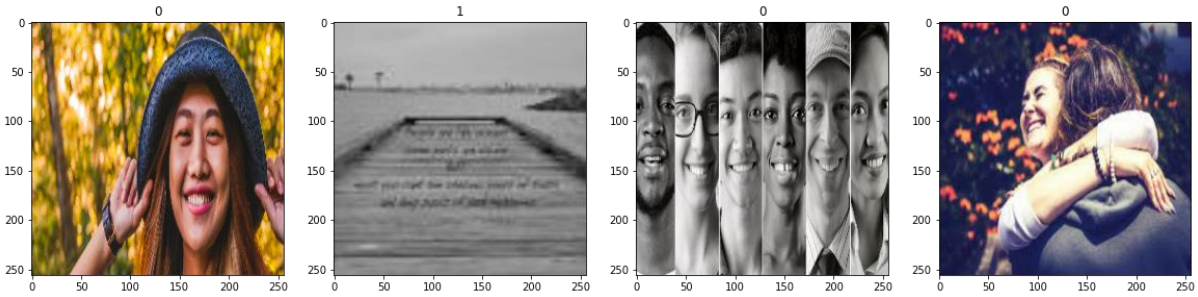
2. Remove dodgy images

```
import cv2
import imghdr
data_dir = 'data'
image_exts = ['jpeg','jpg', 'bmp', 'png']
for image_class in os.listdir(data_dir):
    for image in os.listdir(os.path.join(data_dir, image_class)):
        image_path = os.path.join(data_dir, image_class, image)
        try:
            img = cv2.imread(image_path)
            tip = imghdr.what(image_path)
            if tip not in image_exts:
                print('Image not in ext list {}'.format(image_path))
                os.remove(image_path)
        except Exception as e:
            print('Issue with image {}'.format(image_path))
            # os.remove(image_path)
```

3. Load Data

```
import numpy as np
from matplotlib import pyplot as plt
data = tf.keras.utils.image_dataset_from_directory('data')
data_iterator = data.as_numpy_iterator()
batch = data_iterator.next()
```

```
fig, ax = plt.subplots(ncols=4, figsize=(20,20))
for idx, img in enumerate(batch[0][:4]):
    ax[idx].imshow(img.astype(int))
    ax[idx].title.set_text(batch[1][idx])
```



4. Scale Data

```
data = data.map(lambda x,y: (x/255, y))
data.as_numpy_iterator().next()
```

5. Split Data

```
train_size = int(len(data)*.7)
val_size = int(len(data)*.2)
test_size = int(len(data)*.1)
train_size
train = data.take(train_size)
val = data.skip(train_size).take(val_size)
test = data.skip(train_size+val_size).take(test_size)
```

6. Build Deep Learning Model

train

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout
model = Sequential()
```

```
model.add(Conv2D(16, (3,3), 1, activation='relu', input_shape=(256,256,3)))
model.add(MaxPooling2D())
model.add(Conv2D(32, (3,3), 1, activation='relu'))
model.add(MaxPooling2D())
model.add(Conv2D(16, (3,3), 1, activation='relu'))
model.add(MaxPooling2D())
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile('adam', loss=tf.losses.BinaryCrossentropy(), metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 16)      448

 max_pooling2d (MaxPooling2D  (None, 127, 127, 16)     0
 )

 conv2d_1 (Conv2D)           (None, 125, 125, 32)      4640

 max_pooling2d_1 (MaxPooling  (None, 62, 62, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 60, 60, 16)        4624

 max_pooling2d_2 (MaxPooling  (None, 30, 30, 16)       0
 2D)
```

flatten (Flatten) (None, 14400) 0

dense (Dense) (None, 256) 3686656

dense_1 (Dense) (None, 1) 257

===

Total params: 3,696,625

Trainable params: 3,696,625

Non-trainable params: 0

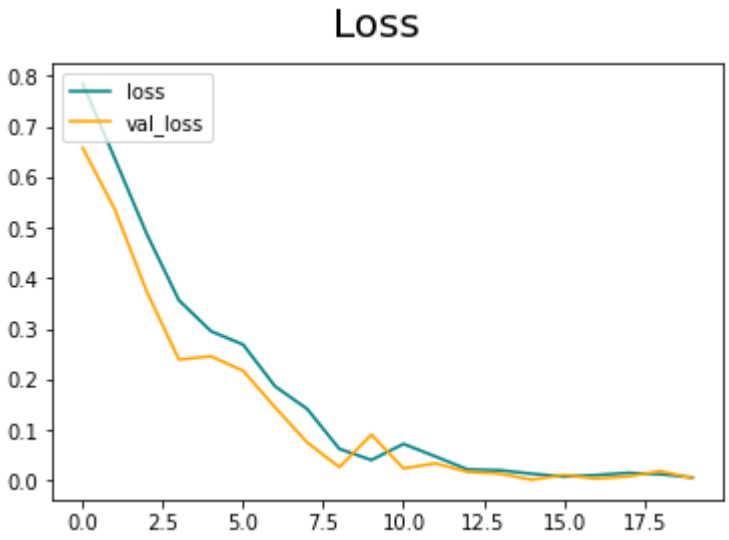
____

7. Train

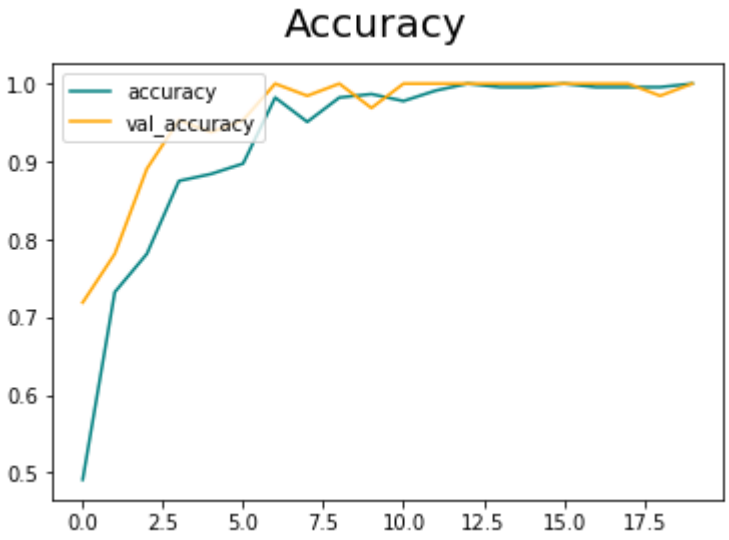
```
logdir='logs'
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=logdir)
hist = model.fit(train, epochs=20, validation_data=val, callbacks=[tensorboard_callback])
```

8. Plot Performance

```
fig = plt.figure()
plt.plot(hist.history['loss'], color='teal', label='loss')
plt.plot(hist.history['val_loss'], color='orange', label='val_loss')
fig.suptitle('Loss', fontsize=20)
plt.legend(loc="upper left")
plt.show()
```




```
fig = plt.figure()
plt.plot(hist.history['accuracy'], color='teal', label='accuracy')
plt.plot(hist.history['val_accuracy'], color='orange', label='val_accuracy')
fig.suptitle('Accuracy', fontsize=20)
plt.legend(loc="upper left")
plt.show()
```



9. Evaluate

```
from tensorflow.keras.metrics import Precision, Recall, BinaryAccuracy

pre = Precision()

re = Recall()

acc = BinaryAccuracy()

for batch in test.as_numpy_iterator():
    X, y = batch
    yhat = model.predict(X)
    pre.update_state(y, yhat)
    re.update_state(y, yhat)
    acc.update_state(y, yhat)

print(pre.result(), re.result(), acc.result())
```

```
tf.Tensor(1.0, shape=(), dtype=float32) tf.Tensor(1.0, shape=(), dtype=float32) tf.Tensor(1.0, shape=(), dtype=float32)
```

10. Test

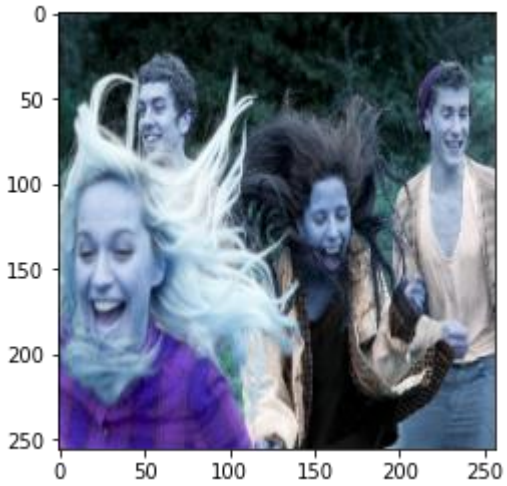
```
import cv2

img = cv2.imread('154006829.jpg')

plt.imshow(img)

plt.show()
```

```
resize = tf.image.resize(img, (256,256))
plt.imshow(resize.numpy().astype(int))
plt.show()
```




```
yhat = model.predict(np.expand_dims(resize/255, 0))
yhat
```

```
array([[0.01972741]], dtype=float32)
```

```
if yhat > 0.5:
    print(f'Predicted class is Sad')
else:
    print(f'Predicted class is Happy')
```

```
Predicted class is Happy
```

11. Save the Model

```
from tensorflow.keras.models import load_model
model.save(os.path.join('models','imageclassifier.h5'))
```

```
new_model = load_model('imageclassifier.h5')
new_model.predict(np.expand_dims(resize/255, 0))
```

```
array([[0.01972741]], dtype=float32)
```

Conclusion

This article guides you toward the first few steps of image processing. It summarizes some applications that are used in image processing. It is intended to make you familiar with some techniques used in the field and their applications. A few takeaways from the article include:

- Image processing is an essential step in upgrading the quality of the image.
- The wide spectrum of applications includes medical, satellite, object detection, and recognition.
- Filters can help remove noise from the image
- The gradient of an image helps detect the edges in the image