1. Install Dependencies and Setup

[1]:
```
pip install tensorflow
```

```
•••
```

[2]:
```
pip install opencv-python
```

```
•••
```

[3]:
```
pip install matplotlib
```

```
•••
```

[5]:
```
#!pip install tensorflow tensorflow-gpu opencv-python matplotlib
```

[4]:
```
!pip list
```

```
•••
```

[5]:
```
import tensorflow as tf
import os
#example use = os.path.join('data', 'vegetation_sample')
```

[6]:
```
# 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)
```

[7]:
```
tf.config.list_physical_devices('GPU')
```

6. Build Deep Learning Model

[45]:
```
train
```

[45]:
```
<_TakeDataset element_spec=(TensorSpec(shape=(None, 256, 256, 3), dtype=tf.float32, name=None), TensorSpec(shape=(None,), dtype=tf.int32, name=None))>
```

[21]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout
```

[47]:
```
model = Sequential()
```

[48]:
```
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'))
```

[49]:
```
model.compile('adam', loss=tf.losses.BinaryCrossentropy(), metrics=['accuracy'])
```

[50]:
```
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
_________________________________________________________________
```

4. Scale Data

[37]:
```
data = data.map(lambda x,y: (x/255, y))
```

[38]:
```
data.as_numpy_iterator().next()
```

...

5. Split Data

[39]:
```
len(data)
```

[39]:
```
25
```

[42]:
```
#70% testing 20% validation 10% testing
train_size = int(len(data)*.7)+1
val_size = int(len(data)*.2)
test_size = int(len(data)*.1)
```

[43]:
```
train_size+val_size+test_size
```

[43]:
```
25
```

[44]:
```
train = data.take(train_size)
val = data.skip(train_size).take(val_size)
test = data.skip(train_size+val_size).take(test_size)
```

2. Remove dodgy images

[8]:
```
import cv2
import imghdr
from matplotlib import pyplot as plt
```

[9]:
```
data_dir = 'data'
```

10]:
```
#show all images in happy = os.listdir(os.path.join(data_dir, 'vegetation_sample'))
```

11]:
```
image_exts = ['jpeg','jpg', 'bmp', 'png']
```

12]:
```
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)
```

•••

13]:
```
#cv2.imread(os.path.join('data','no_vegetation_sample', '1003.jpg'))
```

14]:
```
img.shape
```

7. Train

[51]:
```
logdir='logs'
```

[52]:
```
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=logdir)
```

[53]:
```
hist = model.fit(train, epochs=20, validation_data=val, callbacks=[tensorboard_callback])
```

```
Epoch 1/20
18/18 [==============================] - 16s 762ms/step - loss: 0.8314 - accuracy: 0.5538 - val_loss: 0.6810 - val_accuracy: 0.6812
Epoch 2/20
18/18 [==============================] - 14s 742ms/step - loss: 0.6669 - accuracy: 0.6128 - val_loss: 0.6409 - val_accuracy: 0.6938
Epoch 3/20
18/18 [==============================] - 14s 736ms/step - loss: 0.6319 - accuracy: 0.6649 - val_loss: 0.5930 - val_accuracy: 0.7437
Epoch 4/20
18/18 [==============================] - 14s 760ms/step - loss: 0.6256 - accuracy: 0.6684 - val_loss: 0.6210 - val_accuracy: 0.7000
Epoch 5/20
18/18 [==============================] - 14s 771ms/step - loss: 0.5833 - accuracy: 0.7066 - val_loss: 0.5361 - val_accuracy: 0.7688
Epoch 6/20
18/18 [==============================] - 15s 788ms/step - loss: 0.5676 - accuracy: 0.7170 - val_loss: 0.5466 - val_accuracy: 0.7375
Epoch 7/20
18/18 [==============================] - 14s 757ms/step - loss: 0.4964 - accuracy: 0.7760 - val_loss: 0.4930 - val_accuracy: 0.7688
Epoch 8/20
18/18 [==============================] - 14s 771ms/step - loss: 0.3746 - accuracy: 0.8403 - val_loss: 0.3225 - val_accuracy: 0.8813
Epoch 9/20
18/18 [==============================] - 14s 768ms/step - loss: 0.3284 - accuracy: 0.8767 - val_loss: 0.3479 - val_accuracy: 0.8500
Epoch 10/20
```

[15]:
```
plt.imshow(img)
```

[15]:
```
<matplotlib.image.AxesImage at 0x21425621030>
```

[16]:
```
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.show()
```

3. Load Data

[17]:
```
import numpy as np
from matplotlib import pyplot as plt
```

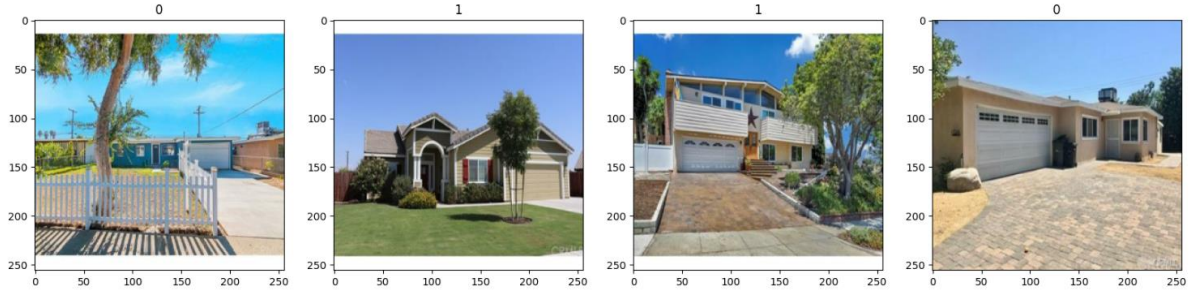
[18]:
```
data = tf.keras.utils.image_dataset_from_directory('data')
```

```
...
```

[19]:
```
data_iterator = data.as_numpy_iterator()
```

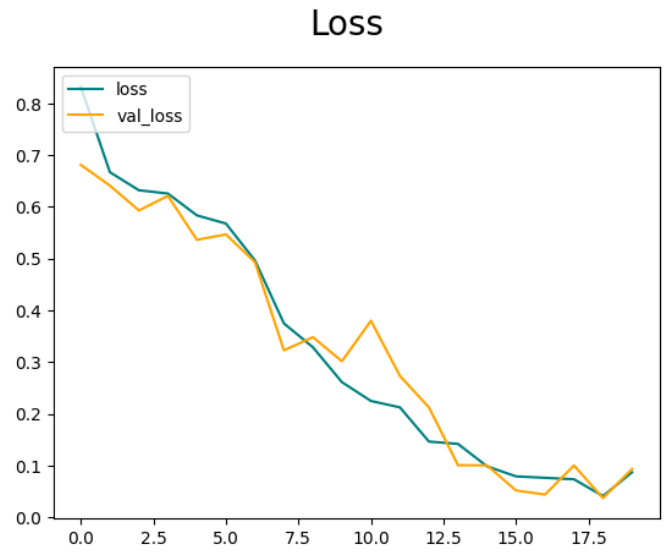
[20]:
```
batch = data_iterator.next()
```

[21]:
```
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])
#0 no veg 1 yes veg
```

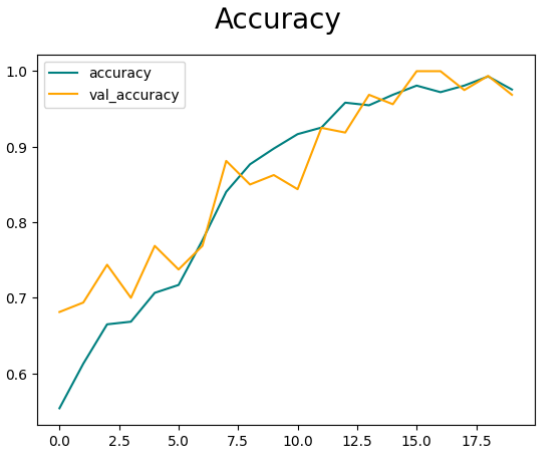


8. Plot Performance

```
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()
```

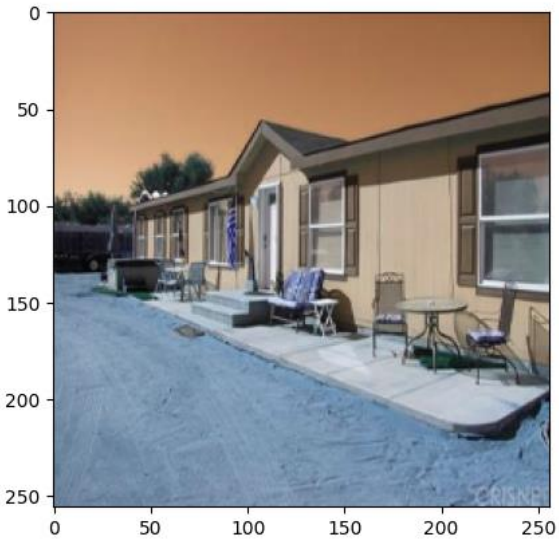


10. Test

```
import cv2
```

```
img = cv2.imread('1003.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))
```

```
1/1 [==============================] - 0s 58ms/step
```

```
yhat
```

```
array([[0.17673443]], dtype=float32)
```

```
if yhat > 0.5:
    print(f'Predicted class is vegetation')
else:
    print(f'Predicted class is non-vegetation')
```

```
Predicted class is non-vegetation
```

11. Save the Model

```
from tensorflow.keras.models import load_model
```

```
model.save(os.path.join('models','vegetationclassifier.h5'))
```

...

```
new_model = load_model(os.path.join('models', 'vegetationclassifier.h5'))
```

```
new_model.predict(np.expand_dims(resize/255, 0))
```

```
1/1 [==============================] - 0s 161ms/step
array([[0.17673443]], dtype=float32)
```