A.I Computer vision Task

Concrete defect analysis

crack or rebar or spall.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Importing Libraries

```
[74] import os
     import cv2
     import numpy as np
     from tqdm import tqdm
     from PIL import Image
     import matplotlib.pyplot as plt
     from sklearn.utils import shuffle
     import seaborn as sns

     import tensorflow as tf
     from tensorflow import keras
     from tensorflow.keras import layers
     from tensorflow.keras.models import Sequential, Model

     from keras.layers import Input, Lambda, Dense, Flatten, Activation, Dropout
     from keras.preprocessing.image import ImageDataGenerator
     from keras.optimizers import RMSprop
     from keras.callbacks import EarlyStopping, ReduceLROnPlateau
     from keras import applications
```

Data Visualization

```
[75] train = list(os.walk('drive/MyDrive/train'))
```

crack or rebar or spall.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text

```
[75] train = list(os.walk('drive/MyDrive/train'))

[78] label_names = train[0][1]
     dict_labels = dict(zip(label_names, list(range(len(label_names)))))
     print(dict_labels)

     {'rebar': 0, 'spalling': 1, 'crack': 2}

     def dataset(path):
         images = []
         labels = []
         for folder in tqdm(os.listdir(path)):
             value_of_label = dict_labels[folder]

             for file in (os.listdir(os.path.join(path, folder))):
                 path_of_file = os.path.join(os.path.join(path, folder), file)

                 image = cv2.imread(path_of_file)
                 image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
                 image = cv2.resize(image, (150, 150))
                 images.append(image)
                 labels.append(value_of_label)

         images = np.array(images, dtype = 'float32')/255.0
         labels = np.array(labels)

         return images, labels
     images, labels = dataset('drive/MyDrive/train')
     images, labels = shuffle(images, labels)
```

crack or rebar or spall.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text

```
100%|██████████| 3/3 [00:00<00:00,  3.31it/s]
```

```
[80] images.shape
```

```
(59, 150, 150, 3)
```

Total No. of Defect

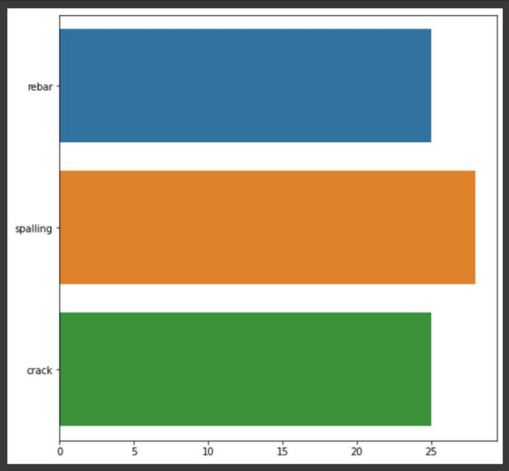
```
species = train[0][1]
number = []
for _ in range(1, 4):
    number.append(len(train[_][0]))
plt.figure(figsize = (8, 8))
sns.barplot(y = species, x = number);
```

crack or rebar or spall.ipynb

File Edit View Insert Runtime Tools Help Saving...

+ Code + Text

```
[81] plt.figure(figsize = (8, 8))
     sns.barplot(y = species, x = number);
```



Sample images

crack or rebar or spall.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

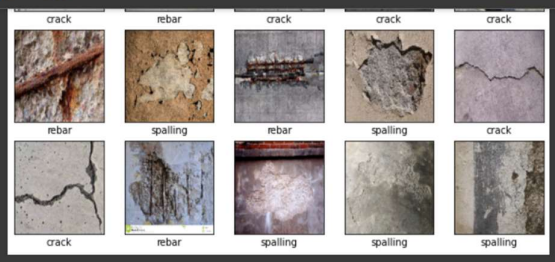
Sample images

```
plt.figure(figsize = (10,10))
for _ in range(25):
    plt.subplot(5, 5, _+1)
    plt.yticks([])
    plt.xticks([])
    plt.grid(False)
    data = images[_]
    plt.xlabel(label_names[labels[_]])
    plt.imshow(data);
```

crack or rebar or spall.ipynb

File Edit View Insert Runtime Tools Help Saving...

+ Code + Text



Load images using ImageDatagenerator

```
[83] image_size = (224, 224)
     batch_size = 3
     train_datagen = ImageDataGenerator(rescale = 1./255,
                                        shear_range = 0.4,
                                        zoom_range = 0.4,
                                        horizontal_flip = True,
                                        vertical_flip = True,
                                        validation_split = 0.2)
```

```
[84] train_ds = train_datagen.flow_from_directory('drive/MyDrive/train',
                                                  target_size = image_size,
                                                  batch_size = batch_size,
                                                  class_mode = 'categorical',
```

crack or rebar or spall.ipynb

File Edit View Insert Runtime Tools Help Saving...

+ Code + Text

```
                                        target_size = image_size,
                                        batch_size = batch_size,
                                        class_mode = 'categorical',
                                        subset = 'training',
                                        color_mode="rgb",)

val_ds = train_datagen.flow_from_directory('drive/MyDrive/train',
                                        target_size = image_size,
                                        batch_size = batch_size,
                                        class_mode = 'categorical',
                                        subset = 'validation',
                                        color_mode="rgb")
```

```
Found 48 images belonging to 3 classes.
Found 11 images belonging to 3 classes.
```

```
[85] train_ds.class_indices
```

```
{'crack': 0, 'rebar': 1, 'spalling': 2}
```

Some augmented images

```
[86] fig, ax = plt.subplots(nrows=1, ncols=5, figsize=(15,15))

     for i in range(5):
         image = next(train_ds)[0][0]
         image = np.squeeze(image)
         ax[i].imshow(image)
         ax[i].axis(False)
```

crack or rebar or spall.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

+ Code + Text

Transfer learning

VGG16

```
[87] vgg_base = applications.VGG16(weights = 'imagenet', include_top = False, input_shape = (224, 224, 3))
     vgg_base.trainable = False
```

```
[88] inputs = Input(shape=(224, 224, 3))

     x = vgg_base(inputs, training=False)
     x = layers.GlobalAveragePooling2D()(x)
     x = layers.Dense(1024, activation = 'relu')(x)
     x = layers.Dropout(0.5)(x)
     outputs = layers.Dense(3, activation = 'sigmoid')(x)
     vgg_model = Model(inputs, outputs)
     vgg_model.summary
```

crack or rebar or spall.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
<bound method Model.summary of <keras.engine.functional.Functional object at 0x7f7cd805e6a0>>
```

[89]
```
vgg_model.compile(
    optimizer=keras.optimizers.Adam(),
    loss= keras.losses.CategoricalCrossentropy(from_logits = True),
    metrics= [keras.metrics.CategoricalAccuracy()],
)
```

```
from tensorflow.keras.layers import BatchNormalization
epochs = 25
vgg_model.fit(train_ds, epochs=epochs, validation_data=val_ds)
```

```
Epoch 1/25
/usr/local/lib/python3.9/dist-packages/keras/backend.py:5534: UserWarning: "`categorical_crossentropy` received `from_logits=True`, but the `output` argumen
  output, from_logits = _get_logits(
16/16 [==============================] - 27s 2s/step - loss: 1.1471 - categorical_accuracy: 0.4375 - val_loss: 1.1319 - val_categorical_accuracy: 0.2727
Epoch 2/25
16/16 [==============================] - 26s 2s/step - loss: 1.1160 - categorical_accuracy: 0.4375 - val_loss: 0.7838 - val_categorical_accuracy: 0.8182
Epoch 3/25
16/16 [==============================] - 26s 2s/step - loss: 0.8168 - categorical_accuracy: 0.6250 - val_loss: 0.7465 - val_categorical_accuracy: 0.8182
Epoch 4/25
16/16 [==============================] - 26s 2s/step - loss: 0.7509 - categorical_accuracy: 0.6458 - val_loss: 0.6342 - val_categorical_accuracy: 0.8182
Epoch 5/25
16/16 [==============================] - 26s 2s/step - loss: 0.9096 - categorical_accuracy: 0.6250 - val_loss: 0.5499 - val_categorical_accuracy: 0.8182
Epoch 6/25
16/16 [==============================] - 26s 2s/step - loss: 0.6009 - categorical_accuracy: 0.7917 - val_loss: 0.6217 - val_categorical_accuracy: 0.8182
Epoch 7/25
16/16 [==============================] - 26s 2s/step - loss: 0.5744 - categorical_accuracy: 0.7083 - val_loss: 0.4998 - val_categorical_accuracy: 0.9091
Epoch 8/25
16/16 [==============================] - 31s 2s/step - loss: 0.7053 - categorical_accuracy: 0.6458 - val_loss: 0.6377 - val_categorical_accuracy: 0.6364
Epoch 9/25
16/16 [==============================] - 27s 2s/step - loss: 0.6582 - categorical_accuracy: 0.6667 - val_loss: 0.4486 - val_categorical_accuracy: 0.7273
Epoch 10/25
```

crack or rebar or spall.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
Epoch 10/25
16/16 [==============================] - 26s 2s/step - loss: 0.6223 - categorical_accuracy: 0.7708 - val_loss: 0.6012 - val_categorical_accuracy: 0.6364
Epoch 11/25
16/16 [==============================] - 26s 2s/step - loss: 0.4619 - categorical_accuracy: 0.8333 - val_loss: 0.6475 - val_categorical_accuracy: 0.7273
Epoch 12/25
16/16 [==============================] - 26s 2s/step - loss: 0.6281 - categorical_accuracy: 0.7500 - val_loss: 0.4949 - val_categorical_accuracy: 0.7273
Epoch 13/25
16/16 [==============================] - 26s 2s/step - loss: 0.4901 - categorical_accuracy: 0.7708 - val_loss: 0.4636 - val_categorical_accuracy: 0.8182
Epoch 14/25
16/16 [==============================] - 26s 2s/step - loss: 0.6733 - categorical_accuracy: 0.7708 - val_loss: 0.3451 - val_categorical_accuracy: 0.8182
Epoch 15/25
16/16 [==============================] - 26s 2s/step - loss: 0.4950 - categorical_accuracy: 0.7917 - val_loss: 0.4817 - val_categorical_accuracy: 0.8182
Epoch 16/25
16/16 [==============================] - 31s 2s/step - loss: 0.6233 - categorical_accuracy: 0.7292 - val_loss: 0.4366 - val_categorical_accuracy: 0.9091
Epoch 17/25
16/16 [==============================] - 26s 2s/step - loss: 0.4288 - categorical_accuracy: 0.8125 - val_loss: 0.4660 - val_categorical_accuracy: 0.8182
Epoch 18/25
16/16 [==============================] - 26s 2s/step - loss: 0.4727 - categorical_accuracy: 0.8125 - val_loss: 0.3918 - val_categorical_accuracy: 0.8182
Epoch 19/25
16/16 [==============================] - 26s 2s/step - loss: 0.4021 - categorical_accuracy: 0.8750 - val_loss: 0.5110 - val_categorical_accuracy: 0.7273
Epoch 20/25
16/16 [==============================] - 26s 2s/step - loss: 0.3531 - categorical_accuracy: 0.8750 - val_loss: 0.4389 - val_categorical_accuracy: 0.9091
Epoch 21/25
16/16 [==============================] - 26s 2s/step - loss: 0.5283 - categorical_accuracy: 0.8125 - val_loss: 0.3528 - val_categorical_accuracy: 0.8182
Epoch 22/25
16/16 [==============================] - 26s 2s/step - loss: 0.4294 - categorical_accuracy: 0.8125 - val_loss: 0.3391 - val_categorical_accuracy: 0.8182
Epoch 23/25
16/16 [==============================] - 26s 2s/step - loss: 0.3770 - categorical_accuracy: 0.8125 - val_loss: 0.2699 - val_categorical_accuracy: 1.0000
Epoch 24/25
16/16 [==============================] - 26s 2s/step - loss: 0.4518 - categorical_accuracy: 0.8125 - val_loss: 0.3508 - val_categorical_accuracy: 0.9091
Epoch 25/25
16/16 [==============================] - 26s 2s/step - loss: 0.4777 - categorical_accuracy: 0.7917 - val_loss: 0.3969 - val_categorical_accuracy: 0.8182
<keras.callbacks.History at 0x7f7cd826ee80>
```

crack or rebar or spall.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[91] vgg_model.save('vgg.hdf5')
```

Testing Model on Own Image

```
def predictor(img, model):
    image = cv2.imread(img)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image = cv2.resize(image, (224, 224))
    image = np.array(image, dtype = 'float32')/255.0
    plt.imshow(image)
    image = image.reshape(1, 224,224,3)

    label_names = train_ds.class_indices
    dict_class = dict(zip(list(range(len(label_names))), label_names))
    clas = model.predict(image).argmax()
    name = dict_class[clas]
    print('The given image is of \nclass: {0} \nDefect Type: {1}'.format(clas, name))
```

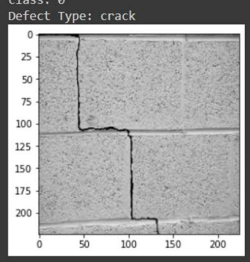
```
predictor('drive/MyDrive/validation/testCrack4.jpg', vgg_model)
```

```
1/1 [==============================] - 0s 400ms/step
The given image is of
class: 0
Defect Type: crack
```

+ Code + Text

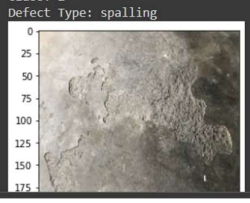
```
predictor('drive/MyDrive/validation/testCrack4.jpg', vgg_model)
```

```
1/1 [==============================] - 0s 400ms/step
The given image is of
class: 0
Defect Type: crack
```



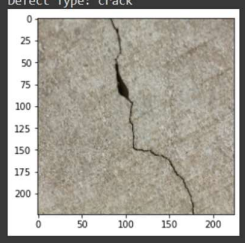
```
[96] predictor('drive/MyDrive/validation/testSpall5.jpg', vgg_model)
```

```
1/1 [==============================] - 0s 403ms/step
The given image is of
class: 2
Defect Type: spalling
```



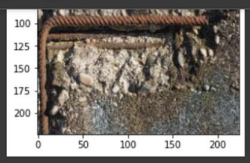
```
+ Code  + Text
[96]
```

```
[97] predictor('drive/MyDrive/validation/testCrack2.jpg', vgg_model)

1/1 [==============================] - 0s 404ms/step
The given image is of
class: 0
Defect Type: crack
```

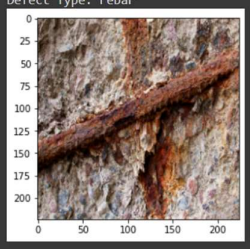


```
predictor('drive/MyDrive/validation/testRebar5.jpg', vgg_model)

1/1 [==============================] - 0s 407ms/step
The given image is of
class: 1
Defect Type: rebar
```

```
+ Code  + Text
[98]
```



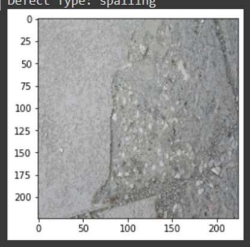
```
predictor('drive/MyDrive/validation/testRebar3.jpg', vgg_model)

1/1 [==============================] - 0s 401ms/step
The given image is of
class: 1
Defect Type: rebar
```



```
[100] predictor('drive/MyDrive/validation/testSpall4.jpg', vgg_model)

1/1 [==============================] - 1s 703ms/step
The given image is of
class: 2
Defect Type: spalling
```

+ Code + Text

[100]
```
Defect Type: spalling
```




```
predictor('drive/MyDrive/validation/testCrack3.jpg', vgg_model)
```

```
1/1 [==============================] - 0s 403ms/step
The given image is of
class: 0
Defect Type: crack
```