

# Fracture detection using Modified and pre-trained VGG19

## Note:

We originally developed the algorithm to detect fractures from image logs / seismic discontinuity (seismic contrasts) from 2D and 3D seismic volumes for the oil/gas and geothermal industry. However, given the issue of data confidentiality, we only present the application of the algorithm to images of fractures/cracks that are available in the public domain.

Here, we only share glimpses of the algorithm (to learn more please contact [yesser.nasser@icloud.com](mailto:yesser.nasser@icloud.com))

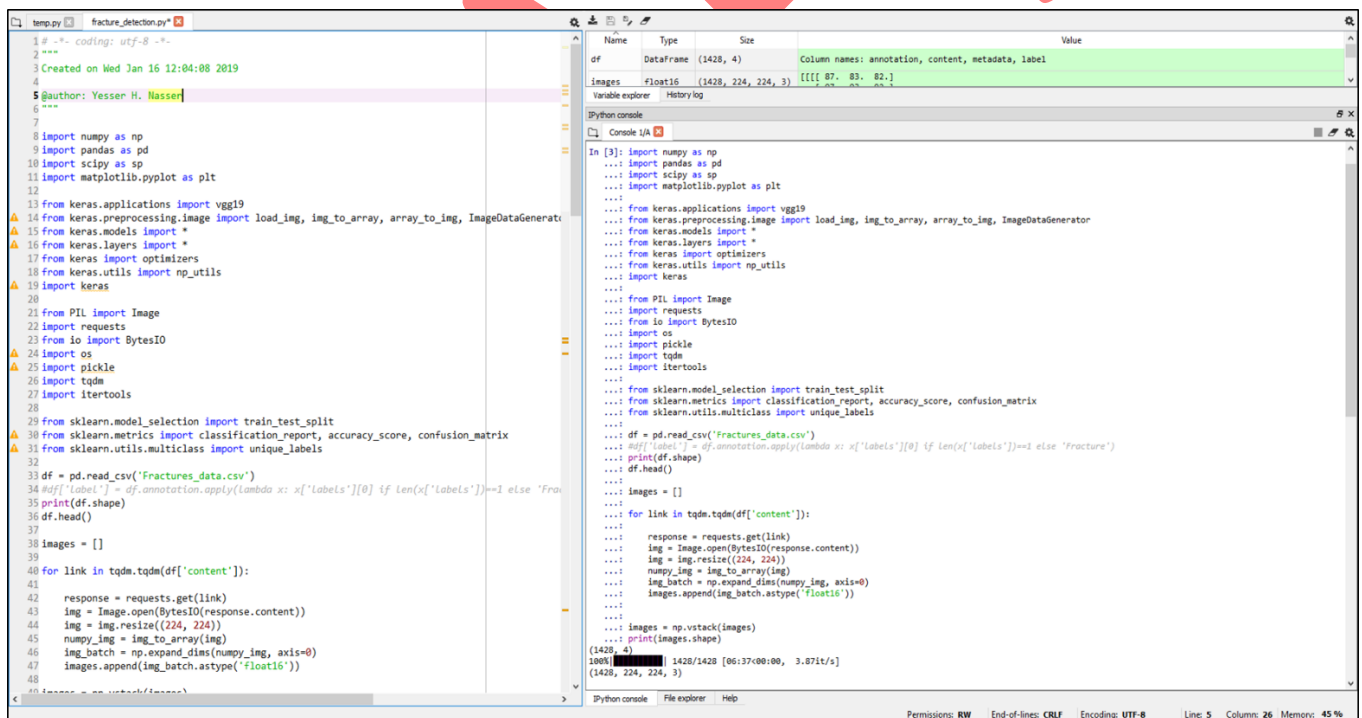
## Abstract

----- an updated version is coming soon -----

### • Input Data:

The input data used for this project consists of images of free-surface fractures/cracks and images of random objects. Additionally, images of fracture-like objects are added to the dataset during the training process to boost the DNN learning process and validate the robustness of the DNN in detecting true fractures.

The dataset consists of 1142 images for training and 286 for testing. Given the architecture, padding (p), strides (s), and number of filters used in this DNN the input data is resized to 224x224x3.



```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Wed Jan 16 12:04:08 2019
4
5 @author: Yesser H. Nasser
6 """
7
8 import numpy as np
9 import pandas as pd
10 import scipy as sp
11 import matplotlib.pyplot as plt
12
13 from keras.applications import vgg19
14 from keras.preprocessing.image import load_img, img_to_array, array_to_img, ImageDataGenerator
15 from keras.models import *
16 from keras.layers import *
17 from keras import optimizers
18 from keras.utils import np_utils
19 import keras
20
21 from PIL import Image
22 import requests
23 from io import BytesIO
24 import os
25 import pickle
26 import tqdm
27 import itertools
28
29 from sklearn.model_selection import train_test_split
30 from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
31 from sklearn.utils.multiclass import unique_labels
32
33 df = pd.read_csv('Fractures_data.csv')
34 df['label'] = df.annotation.apply(lambda x: x['labels'][0] if len(x['labels'])==1 else 'Fracture')
35 print(df.shape)
36 df.head()
37
38 images = []
39
40 for link in tqdm.tqdm(df['content']):
41     response = requests.get(link)
42     img = Image.open(BytesIO(response.content))
43     img = img.resize((224, 224))
44     numpy_img = img_to_array(img)
45     img_batch = np.expand_dims(numpy_img, axis=0)
46     images.append(img_batch.astype('float16'))
47
48 images = np.vstack(images)
49
50 print(images.shape)
51 print(images)
52
53 (1428, 4)
54 array([[[[ 0.0, 0.0, 0.0, 0.0],
55         [ 0.0, 0.0, 0.0, 0.0],
56         [ 0.0, 0.0, 0.0, 0.0],
57         [ 0.0, 0.0, 0.0, 0.0],
58         [ 0.0, 0.0, 0.0, 0.0],
59         [ 0.0, 0.0, 0.0, 0.0],
60         [ 0.0, 0.0, 0.0, 0.0],
61         [ 0.0, 0.0, 0.0, 0.0],
62         [ 0.0, 0.0, 0.0, 0.0],
63         [ 0.0, 0.0, 0.0, 0.0],
64         [ 0.0, 0.0, 0.0, 0.0],
65         [ 0.0, 0.0, 0.0, 0.0],
66         [ 0.0, 0.0, 0.0, 0.0],
67         [ 0.0, 0.0, 0.0, 0.0],
68         [ 0.0, 0.0, 0.0, 0.0],
69         [ 0.0, 0.0, 0.0, 0.0],
70         [ 0.0, 0.0, 0.0, 0.0],
71         [ 0.0, 0.0, 0.0, 0.0],
72         [ 0.0, 0.0, 0.0, 0.0],
73         [ 0.0, 0.0, 0.0, 0.0],
74         [ 0.0, 0.0, 0.0, 0.0],
75         [ 0.0, 0.0, 0.0, 0.0],
76         [ 0.0, 0.0, 0.0, 0.0],
77         [ 0.0, 0.0, 0.0, 0.0],
78         [ 0.0, 0.0, 0.0, 0.0],
79         [ 0.0, 0.0, 0.0, 0.0],
80         [ 0.0, 0.0, 0.0, 0.0],
81         [ 0.0, 0.0, 0.0, 0.0],
82         [ 0.0, 0.0, 0.0, 0.0],
83         [ 0.0, 0.0, 0.0, 0.0],
84         [ 0.0, 0.0, 0.0, 0.0],
85         [ 0.0, 0.0, 0.0, 0.0],
86         [ 0.0, 0.0, 0.0, 0.0],
87         [ 0.0, 0.0, 0.0, 0.0],
88         [ 0.0, 0.0, 0.0, 0.0],
89         [ 0.0, 0.0, 0.0, 0.0],
90         [ 0.0, 0.0, 0.0, 0.0],
91         [ 0.0, 0.0, 0.0, 0.0],
92         [ 0.0, 0.0, 0.0, 0.0],
93         [ 0.0, 0.0, 0.0, 0.0],
94         [ 0.0, 0.0, 0.0, 0.0],
95         [ 0.0, 0.0, 0.0, 0.0],
96         [ 0.0, 0.0, 0.0, 0.0],
97         [ 0.0, 0.0, 0.0, 0.0],
98         [ 0.0, 0.0, 0.0, 0.0],
99         [ 0.0, 0.0, 0.0, 0.0],
100        [ 0.0, 0.0, 0.0, 0.0],
101        [ 0.0, 0.0, 0.0, 0.0],
102        [ 0.0, 0.0, 0.0, 0.0],
103        [ 0.0, 0.0, 0.0, 0.0],
104        [ 0.0, 0.0, 0.0, 0.0],
105        [ 0.0, 0.0, 0.0, 0.0],
106        [ 0.0, 0.0, 0.0, 0.0],
107        [ 0.0, 0.0, 0.0, 0.0],
108        [ 0.0, 0.0, 0.0, 0.0],
109        [ 0.0, 0.0, 0.0, 0.0],
110        [ 0.0, 0.0, 0.0, 0.0],
111        [ 0.0, 0.0, 0.0, 0.0],
112        [ 0.0, 0.0, 0.0, 0.0],
113        [ 0.0, 0.0, 0.0, 0.0],
114        [ 0.0, 0.0, 0.0, 0.0],
115        [ 0.0, 0.0, 0.0, 0.0],
116        [ 0.0, 0.0, 0.0, 0.0],
117        [ 0.0, 0.0, 0.0, 0.0],
118        [ 0.0, 0.0, 0.0, 0.0],
119        [ 0.0, 0.0, 0.0, 0.0],
120        [ 0.0, 0.0, 0.0, 0.0],
121        [ 0.0, 0.0, 0.0, 0.0],
122        [ 0.0, 0.0, 0.0, 0.0],
123        [ 0.0, 0.0, 0.0, 0.0],
124        [ 0.0, 0.0, 0.0, 0.0],
125        [ 0.0, 0.0, 0.0, 0.0],
126        [ 0.0, 0.0, 0.0, 0.0],
127        [ 0.0, 0.0, 0.0, 0.0],
128        [ 0.0, 0.0, 0.0, 0.0],
129        [ 0.0, 0.0, 0.0, 0.0],
130        [ 0.0, 0.0, 0.0, 0.0],
131        [ 0.0, 0.0, 0.0, 0.0],
132        [ 0.0, 0.0, 0.0, 0.0],
133        [ 0.0, 0.0, 0.0, 0.0],
134        [ 0.0, 0.0, 0.0, 0.0],
135        [ 0.0, 0.0, 0.0, 0.0],
136        [ 0.0, 0.0, 0.0, 0.0],
137        [ 0.0, 0.0, 0.0, 0.0],
138        [ 0.0, 0.0, 0.0, 0.0],
139        [ 0.0, 0.0, 0.0, 0.0],
140        [ 0.0, 0.0, 0.0, 0.0],
141        [ 0.0, 0.0, 0.0, 0.0],
142        [ 0.0, 0.0, 0.0, 0.0],
143        [ 0.0, 0.0, 0.0, 0.0],
144        [ 0.0, 0.0, 0.0, 0.0],
145        [ 0.0, 0.0, 0.0, 0.0],
146        [ 0.0, 0.0, 0.0, 0.0],
147        [ 0.0, 0.0, 0.0, 0.0],
148        [ 0.0, 0.0, 0.0, 0.0],
149        [ 0.0, 0.0, 0.0, 0.0],
150        [ 0.0, 0.0, 0.0, 0.0],
151        [ 0.0, 0.0, 0.0, 0.0],
152        [ 0.0, 0.0, 0.0, 0.0],
153        [ 0.0, 0.0, 0.0, 0.0],
154        [ 0.0, 0.0, 0.0, 0.0],
155        [ 0.0, 0.0, 0.0, 0.0],
156        [ 0.0, 0.0, 0.0, 0.0],
157        [ 0.0, 0.0, 0.0, 0.0],
158        [ 0.0, 0.0, 0.0, 0.0],
159        [ 0.0, 0.0, 0.0, 0.0],
160        [ 0.0, 0.0, 0.0, 0.0],
161        [ 0.0, 0.0, 0.0, 0.0],
162        [ 0.0, 0.0, 0.0, 0.0],
163        [ 0.0, 0.0, 0.0, 0.0],
164        [ 0.0, 0.0, 0.0, 0.0],
165        [ 0.0, 0.0, 0.0, 0.0],
166        [ 0.0, 0.0, 0.0, 0.0],
167        [ 0.0, 0.0, 0.0, 0.0],
168        [ 0.0, 0.0, 0.0, 0.0],
169        [ 0.0, 0.0, 0.0, 0.0],
170        [ 0.0, 0.0, 0.0, 0.0],
171        [ 0.0, 0.0, 0.0, 0.0],
172        [ 0.0, 0.0, 0.0, 0.0],
173        [ 0.0, 0.0, 0.0, 0.0],
174        [ 0.0, 0.0, 0.0, 0.0],
175        [ 0.0, 0.0, 0.0, 0.0],
176        [ 0.0, 0.0, 0.0, 0.0],
177        [ 0.0, 0.0, 0.0, 0.0],
178        [ 0.0, 0.0, 0.0, 0.0],
179        [ 0.0, 0.0, 0.0, 0.0],
180        [ 0.0, 0.0, 0.0, 0.0],
181        [ 0.0, 0.0, 0.0, 0.0],
182        [ 0.0, 0.0, 0.0, 0.0],
183        [ 0.0, 0.0, 0.0, 0.0],
184        [ 0.0, 0.0, 0.0, 0.0],
185        [ 0.0, 0.0, 0.0, 0.0],
186        [ 0.0, 0.0, 0.0, 0.0],
187        [ 0.0, 0.0, 0.0, 0.0],
188        [ 0.0, 0.0, 0.0, 0.0],
189        [ 0.0, 0.0, 0.0, 0.0],
190        [ 0.0, 0.0, 0.0, 0.0],
191        [ 0.0, 0.0, 0.0, 0.0],
192        [ 0.0, 0.0, 0.0, 0.0],
193        [ 0.0, 0.0, 0.0, 0.0],
194        [ 0.0, 0.0, 0.0, 0.0],
195        [ 0.0, 0.0, 0.0, 0.0],
196        [ 0.0, 0.0, 0.0, 0.0],
197        [ 0.0, 0.0, 0.0, 0.0],
198        [ 0.0, 0.0, 0.0, 0.0],
199        [ 0.0, 0.0, 0.0, 0.0],
200        [ 0.0, 0.0, 0.0, 0.0],
201        [ 0.0, 0.0, 0.0, 0.0],
202        [ 0.0, 0.0, 0.0, 0.0],
203        [ 0.0, 0.0, 0.0, 0.0],
204        [ 0.0, 0.0, 0.0, 0.0],
205        [ 0.0, 0.0, 0.0, 0.0],
206        [ 0.0, 0.0, 0.0, 0.0],
207        [ 0.0, 0.0, 0.0, 0.0],
208        [ 0.0, 0.0, 0.0, 0.0],
209        [ 0.0, 0.0, 0.0, 0.0],
210        [ 0.0, 0.0, 0.0, 0.0],
211        [ 0.0, 0.0, 0.0, 0.0],
212        [ 0.0, 0.0, 0.0, 0.0],
213        [ 0.0, 0.0, 0.0, 0.0],
214        [ 0.0, 0.0, 0.0, 0.0],
215        [ 0.0, 0.0, 0.0, 0.0],
216        [ 0.0, 0.0, 0.0, 0.0],
217        [ 0.0, 0.0, 0.0, 0.0],
218        [ 0.0, 0.0, 0.0, 0.0],
219        [ 0.0, 0.0, 0.0, 0.0],
220        [ 0.0, 0.0, 0.0, 0.0],
221        [ 0.0, 0.0, 0.0, 0.0],
222        [ 0.0, 0.0, 0.0, 0.0],
223        [ 0.0, 0.0, 0.0, 0.0],
224        [ 0.0, 0.0, 0.0, 0.0],
225        [ 0.0, 0.0, 0.0, 0.0],
226        [ 0.0, 0.0, 0.0, 0.0],
227        [ 0.0, 0.0, 0.0, 0.0],
228        [ 0.0, 0.0, 0.0, 0.0],
229        [ 0.0, 0.0, 0.0, 0.0],
230        [ 0.0, 0.0, 0.0, 0.0],
231        [ 0.0, 0.0, 0.0, 0.0],
232        [ 0.0, 0.0, 0.0, 0.0],
233        [ 0.0, 0.0, 0.0, 0.0],
234        [ 0.0, 0.0, 0.0, 0.0],
235        [ 0.0, 0.0, 0.0, 0.0],
236        [ 0.0, 0.0, 0.0, 0.0],
237        [ 0.0, 0.0, 0.0, 0.0],
238        [ 0.0, 0.0, 0.0, 0.0],
239        [ 0.0, 0.0, 0.0, 0.0],
240        [ 0.0, 0.0, 0.0, 0.0],
241        [ 0.0, 0.0, 0.0, 0.0],
242        [ 0.0, 0.0, 0.0, 0.0],
243        [ 0.0, 0.0, 0.0, 0.0],
244        [ 0.0, 0.0, 0.0, 0.0],
245        [ 0.0, 0.0, 0.0, 0.0],
246        [ 0.0, 0.0, 0.0, 0.0],
247        [ 0.0, 0.0, 0.0, 0.0],
248        [ 0.0, 0.0, 0.0, 0.0],
249        [ 0.0, 0.0, 0.0, 0.0],
250        [ 0.0, 0.0, 0.0, 0.0],
251        [ 0.0, 0.0, 0.0, 0.0],
252        [ 0.0, 0.0, 0.0, 0.0],
253        [ 0.0, 0.0, 0.0, 0.0],
254        [ 0.0, 0.0, 0.0, 0.0],
255        [ 0.0, 0.0, 0.0, 0.0],
256        [ 0.0, 0.0, 0.0, 0.0],
257        [ 0.0, 0.0, 0.0, 0.0],
258        [ 0.0, 0.0, 0.0, 0.0],
259        [ 0.0, 0.0, 0.0, 0.0],
260        [ 0.0, 0.0, 0.0, 0.0],
261        [ 0.0, 0.0, 0.0, 0.0],
262        [ 0.0, 0.0, 0.0, 0.0],
263        [ 0.0, 0.0, 0.0, 0.0],
264        [ 0.0, 0.0, 0.0, 0.0],
265        [ 0.0, 0.0, 0.0, 0.0],
266        [ 0.0, 0.0, 0.0, 0.0],
267        [ 0.0, 0.0, 0.0, 0.0],
268        [ 0.0, 0.0, 0.0, 0.0],
269        [ 0.0, 0.0, 0.0, 0.0],
270        [ 0.0, 0.0, 0.0, 0.0],
271        [ 0.0, 0.0, 0.0, 0.0],
272        [ 0.0, 0.0, 0.0, 0.0],
273        [ 0.0, 0.0, 0.0, 0.0],
274        [ 0.0, 0.0, 0.0, 0.0],
275        [ 0.0, 0.0, 0.0, 0.0],
276        [ 0.0, 0.0, 0.0, 0.0],
277        [ 0.0, 0.0, 0.0, 0.0],
278        [ 0.0, 0.0, 0.0, 0.0],
279        [ 0.0, 0.0, 0.0, 0.0],
280        [ 0.0, 0.0, 0.0, 0.0],
281        [ 0.0, 0.0, 0.0, 0.0],
282        [ 0.0, 0.0, 0.0, 0.0],
283        [ 0.0, 0.0, 0.0, 0.0],
284        [ 0.0, 0.0, 0.0, 0.0],
285        [ 0.0, 0.0, 0.0, 0.0],
286        [ 0.0, 0.0, 0.0, 0.0],
287        [ 0.0, 0.0, 0.0, 0.0],
288        [ 0.0, 0.0, 0.0, 0.0],
289        [ 0.0, 0.0, 0.0, 0.0],
290        [ 0.0, 0.0, 0.0, 0.0],
291        [ 0.0, 0.0, 0.0, 0.0],
292        [ 0.0, 0.0, 0.0, 0.0],
293        [ 0.0, 0.0, 0.0, 0.0],
294        [ 0.0, 0.0, 0.0, 0.0],
295        [ 0.0, 0.0, 0.0, 0.0],
296        [ 0.0, 0.0, 0.0, 0.0],
297        [ 0.0, 0.0, 0.0, 0.0],
298        [ 0.0, 0.0, 0.0, 0.0],
299        [ 0.0, 0.0, 0.0, 0.0],
300        [ 0.0, 0.0, 0.0, 0.0],
301        [ 0.0, 0.0, 0.0, 0.0],
302        [ 0.0, 0.0, 0.0, 0.0],
303        [ 0.0, 0.0, 0.0, 0.0],
304        [ 0.0, 0.0, 0.0, 0.0],
305        [ 0.0, 0.0, 0.0, 0.0],
306        [ 0.0, 0.0, 0.0, 0.0],
307        [ 0.0, 0.0, 0.0, 0.0],
308        [ 0.0, 0.0, 0.0, 0.0],
309        [ 0.0, 0.0, 0.0, 0.0],
310        [ 0.0, 0.0, 0.0, 0.0],
311        [ 0.0, 0.0, 0.0, 0.0],
312        [ 0.0, 0.0, 0.0, 0.0],
313        [ 0.0, 0.0, 0.0, 0.0],
314        [ 0.0, 0.0, 0.0, 0.0],
315        [ 0.0, 0.0, 0.0, 0.0],
316        [ 0.0, 0.0, 0.0, 0.0],
317        [ 0.0, 0.0, 0.0, 0.0],
318        [ 0.0, 0.0, 0.0, 0.0],
319        [ 0.0, 0.0, 0.0, 0.0],
320        [ 0.0, 0.0, 0.0, 0.0],
321        [ 0.0, 0.0, 0.0, 0.0],
322        [ 0.0, 0.0, 0.0, 0.0],
323        [ 0.0, 0.0, 0.0, 0.0],
324        [ 0.0, 0.0, 0.0, 0.0],
325        [ 0.0, 0.0, 0.0, 0.0],
326        [ 0.0, 0.0, 0.0, 0.0],
327        [ 0.0, 0.0, 0.0, 0.0],
328        [ 0.0, 0.0, 0.0, 0.0],
329        [ 0.0, 0.0, 0.0, 0.0],
330        [ 0.0, 0.0, 0.0, 0.0],
331        [ 0.0, 0.0, 0.0, 0.0],
332        [ 0.0, 0.0, 0.0, 0.0],
333        [ 0.0, 0.0, 0.0, 0.0],
334        [ 0.0, 0.0, 0.0, 0.0],
335        [ 0.0, 0.0, 0.0, 0.0],
336        [ 0.0, 0.0, 0.0, 0.0],
337        [ 0.0, 0.0, 0.0, 0.0],
338        [ 0.0, 0.0, 0.0, 0.0],
339        [ 0.0, 0.0, 0.0, 0.0],
340        [ 0.0, 0.0, 0.0, 0.0],
341        [ 0.0, 0.0, 0.0, 0.0],
342        [ 0.0, 0.0, 0.0, 0.0],
343        [ 0.0, 0.0, 0.0, 0.0],
344        [ 0.0, 0.0, 0.0, 0.0],
345        [ 0.0, 0.0, 0.0, 0.0],
346        [ 0.0, 0.0, 0.0, 0.0],
347        [ 0.0, 0.0, 0.0, 0.0],
348        [ 0.0, 0.0, 0.0, 0.0],
349        [ 0.0, 0.0, 0.0, 0.0],
350        [ 0.0, 0.0, 0.0, 0.0],
351        [ 0.0, 0.0, 0.0, 0.0],
352        [ 0.0, 0.0, 0.0, 0.0],
353        [ 0.0, 0.0, 0.0, 0.0],
354        [ 0.0, 0.0, 0.0, 0.0],
355        [ 0.0, 0.0, 0.0, 0.0],
356        [ 0.0, 0.0, 0.0, 0.0],
357        [ 0.0, 0.0, 0.0, 0.0],
358        [ 0.0, 0.0, 0.0, 0.0],
359        [ 0.0, 0.0, 0.0, 0.0],
360        [ 0.0, 0.0, 0.0, 0.0],
361        [ 0.0, 0.0, 0.0, 0.0],
362        [ 0.0, 0.0, 0.0, 0.0],
363        [ 0.0, 0.0, 0.0, 0.0],
364        [ 0.0, 0.0, 0.0, 0.0],
365        [ 0.0, 0.0, 0.0, 0.0],
366        [ 0.0, 0.0, 0.0, 0.0],
367        [ 0.0, 0.0, 0.0, 0.0],
368        [ 0.0, 0.0, 0.0, 0.0],
369        [ 0.0, 0.0, 0.0, 0.0],
370        [ 0.0, 0.0, 0.0, 0.0],
371        [ 0.0, 0.0, 0.0, 0.0],
372        [ 0.0, 0.0, 0.0, 0.0],
373        [ 0.0, 0.0, 0.0, 0.0],
374        [ 0.0, 0.0, 0.0, 0.0],
375        [ 0.0, 0.0, 0.0, 0.0],
376        [ 0.0, 0.0, 0.0, 0.0],
377        [ 0.0, 0.0, 0.0, 0.0],
378        [ 0.0, 0.0, 0.0, 0.0],
379        [ 0.0, 0.0, 0.0, 0.0],
380        [ 0.0, 0.0, 0.0, 0.0],
381        [ 0.0, 0.0, 0.0, 0.0],
382        [ 0.0, 0.0, 0.0, 0.0],
383        [ 0.0, 0.0, 0.0, 0.0],
384        [ 0.0, 0.0, 0.0, 0.0],
385        [ 0.0, 0.0, 0.0, 0.0],
386        [ 0.0, 0.0, 0.0, 0.0],
387        [ 0.0, 0.0, 0.0, 0.0],
388        [ 0.0, 0.0, 0.0, 0.0],
389        [ 0.0, 0.0, 0.0, 0.0],
390        [ 0.0, 0.0, 0.0, 0.0],
391        [ 0.0, 0.0, 0.0, 0.0],
392        [ 0.0, 0.0, 0.0, 0.0],
393        [ 0.0, 0.0, 0.0, 0.0],
394        [ 0.0, 0.0, 0.0, 0.0],
395        [ 0.0, 0.0, 0.0, 0.0],
396        [ 0.0, 0.0, 0.0, 0.0],
397        [ 0.0, 0.0, 0.0, 0.0],
398        [ 0.0, 0.0, 0.0, 0.0],
399        [ 0.0, 0.0, 0.0, 0.0],
400        [ 0.0, 0.0, 0.0, 0.0],
401        [ 0.0, 0.0, 0.0, 0.0],
402        [ 0.0, 0.0, 0.0, 0.0],
403        [ 0.0, 0.0, 0.0, 0.0],
404        [ 0.0, 0.0, 0.0, 0.0],
405        [ 0.0, 0.0, 0.0, 0.0],
406        [ 0.0, 0.0, 0.0, 0.0],
407        [ 0.0, 0.0, 0.0, 0.0],
408        [ 0.0, 0.0, 0.0, 0.0],
409        [ 0.0, 0.0, 0.0, 0.0],
410        [ 0.0, 0.0, 0.0, 0.0],
411        [ 0.0, 0.0, 0.0, 0.0],
412        [ 0.0, 0.0, 0.0, 0.0],
413        [ 0.0, 0.0, 0.0, 0.0],
414        [ 0.0, 0.0, 0.0, 0.0],
415        [ 0.0, 0.0, 0.0, 0.0],
416        [ 0.0, 0.0, 0.0, 0.0],
417        [ 0.0, 0.0, 0.0, 0.0],
418        [ 0.0, 0.0, 0.0, 0.0],
419        [ 0.0, 0.0, 0.0, 0.0],
420        [ 0.0, 0.0, 0.0, 0.0],
421        [ 0.0, 0.0, 0.0, 0.0],
422        [ 0.0, 0.0, 0.0, 0.0],
423        [ 0.0, 0.0, 0.0, 0.0],
424        [ 0.0, 0.0, 0.0, 0.0],
425        [ 0.0, 0.0, 0.0, 0.0],
426        [ 0.0, 0.0, 0.0, 0.0],
427        [ 0.0, 0.0, 0.0, 0.0],
428        [ 0.0, 0.0, 0.0, 0.0],
429        [ 0.0, 0.0, 0.0, 0.0],
430        [ 0.0, 0.0, 0.0, 0.0],
431        [ 0.0, 0.0, 0.0, 0.0],
432        [ 0.0, 0.0, 0.0, 0.0],
433        [ 0.0, 0.0, 0.0, 0.0],
434        [ 0.0, 0.0, 0.0, 0.0],
435        [ 0.0, 0.0, 0.0, 0.0],
436        [ 0.0, 0.0, 0.0, 0.0],
437        [ 0.0, 0.0, 0.0, 0.0],
438        [ 0.0, 0.0, 0.0, 0.0],
439        [ 0.0, 0.0, 0.0, 0.0],
440        [ 0.0, 0.0, 0.0, 0.0],
441        [ 0.0, 0.0, 0.0, 0.0],
442        [ 0.0, 0.0, 0.0, 0.0],
443        [ 0.0, 0.0, 0.0, 0.0],
444        [ 0.0, 0.0, 0.0, 0.0],
445        [ 0.0, 0.0, 0.0, 0.0],
446        [ 0.0, 0.0, 0.0, 0.0],
447        [ 0.0, 0.0, 0.0, 0.0],
448        [ 0.0, 0.0, 0.0, 0.0],
449        [ 0.0, 0.0, 0.0, 0.0],
450        [ 0.0, 0.0, 0.0, 0.0],
451        [ 0.0, 0.0, 0.0, 0.0],
452        [ 0.0, 0.0, 0.0, 0.0],
453        [ 0.0, 0.0, 0.0, 0.0],
454        [ 0.0, 0.0, 0.0, 0.0],
455        [ 0.0, 0.0, 0.0, 0.0],
456        [ 0.0, 0.0, 0.0, 0.0],
457        [ 0.0, 0.0, 0.0, 0.0],
458        [ 0.0, 0.0, 0.0, 0.0],
459        [ 0.0, 0.0, 0.0, 0.0],
460        [ 0.0, 0.0, 0.0, 0.0],
461        [ 0.0, 0.0, 0.0, 0.0],
462        [ 0.0, 0.0, 0.0, 0.0],
463        [ 0.0, 0.0, 0.0, 0.0],
464        [ 0.0, 0.0, 0.0, 0.0],
465        [ 0.0, 0.0, 0.0, 0.0],
466        [ 0.0, 0.0, 0.0, 0.0],
467        [ 0.0, 0.0, 0.0, 0.0],
468        [ 0.0, 0.0, 0.0, 0.0],
469        [ 0.0, 0.0, 0.0, 0.0],
470        [ 0.0, 0.0, 0.0, 0.0],
471        [ 0.0, 0.0, 0.0, 0.0],
472        [ 0.0, 0.0, 0.0, 0.0],
473        [ 0.0, 0.0, 0.0, 0.0],
474        [ 0.0, 0.0, 0.0, 0.0],
475        [ 0.0, 0.0, 0.0, 0.0],
476        [ 0.0, 0.0, 0.0, 0.0],
477        [ 0.0, 0.0, 0.0, 0.0],
478        [ 0.0, 0.0, 0.0, 0.0],
479        [ 0.0, 0.0, 0.0, 0.0],
480        [ 0.0, 0.0, 0.0, 0.0],
481        [ 0.0, 0.0, 0.0
```

Name	Type	Size	Value
X_test	float16	(286, 224, 224, 3)	[[[143. 135. 132.] [140. 132. 129.]
X_train	float16	(1142, 224, 224, 3)	[[[ 23. 33. 42.] [ 27. 36. 43.]
axes	object	(4,)	ndarray object of numpy module
conf_matrix	int64	(2, 2)	[[122 40] [ 6 118]]
df	DataFrame	(1428, 4)	Column names: annotation, content, metadata, label
i	int	1	25
images	float16	(1428, 224, 224, 3)	[[[ 87. 83. 82.] [ 97. 93. 92.]
img	float16	(224, 224, 3)	[[187. 178. 173.] [184. 175. 170.]
img_batch	float32	(1, 224, 224, 3)	[[[217. 203. 177.] [218. 204. 178.]
link	str	1	
numpy_img	float32	(224, 224, 3)	[[[217. 203. 177.] [218. 204. 178.]
random_id	int32	(4,)	[1380 878 722 918]
title	str	1	Crack
y	float32	(1428, 2)	[[1. 0.] [0. 1.]
y_pred	float32	(286, 2)	[[1.1736659e-10 1.0000000e+00] [8.7819628e-02 9.1218042e-01]
y_test	float32	(286, 2)	[[0. 1.] [1. 0.]
y_train	float32	(1142, 2)	[[0. 1.] [1. 0.]

Figure 2: Data structure and dimensions



Figure 3: Random display of 25 images from the dataset. The training dataset consist of fractures and random objects.

Figure 2 summarizes the data dimension and structure. Figure 3 shows random display of 25 images of fractures and other random objects. We include 'fracture-like' objects to challenge the model and hence increase the accuracy of the DNN predictions. Given the size of the data we applied data augmentation techniques (horizontal flip, width and height shift, and rotation).

- **DNN - Structure**

In this work we use VGG 19 Neural Network structure with modification and prior training from larger dataset. Here, the prior training is introduced for the purpose of learning transfer between datasets and simplify features/anomaly detections (figure 4). This learning transfer consists of preserving the pre-trained weights of the first  $N - p$  layers using the large dataset and then train the last  $p$  layers using the new dataset. In this process a total of 14M new parameters (weights and hyperparameters) will be calibrated during the training process. To achieve better accuracy and reduce loss we modify the network to fit the net dataset and the number of classes.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
.....		
dense_1 (Dense)	(None, 2)	1026
.....		
Trainable params: 14,159,874		

Figure 4: Overview of the Network architecture, the full details are not shared in this document.

#### • Training

To evaluate the training process, we use 'categorical cross-entropy' as a loss function and 'Stochastic Gradient Descent' as an optimizer with a learning rate of 0.0001 and a momentum of 0.9. The metric for our model is 'accuracy'. We run 20 iterations, Figure 5 is a snapshots of loss and accuracy with iterations.

```
Epoch 1/20
36/35 [=====] - 372s 10s/step - loss: 0.8554 - acc: 0.5679
Epoch 2/20
36/35 [=====] - 368s 10s/step - loss: 0.6314 - acc: 0.6691TA: 2:32
- loss: 0.6388 - acc: 0.6577
Epoch 3/20
36/35 [=====] - 377s 10s/step - loss: 0.5841 - acc: 0.7247
.....
[=====] - ETA: 3:32 - loss: 0.4135 - acc: 0.816736/35
[=====] - 363s 10s/step - loss: 0.3913 - acc: 0.8287
Epoch 19/20
36/35 [=====] - 372s 10s/step - loss: 0.3176 - acc: 0.8722
Epoch 20/20
36/35 [=====] - 367s 10s/step - loss: 0.3081 - acc: 0.8756
```

Figure 5: Training Iterations (model achieves accuracy of 0.87 after 20 iterations)

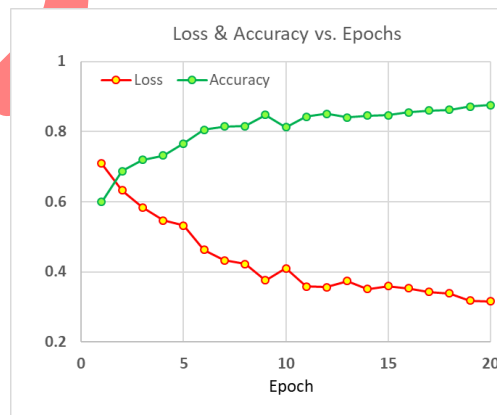


Figure 6: Loss and Accuracy during training.

- **Testing /Evolution: results (classification report)**

To evaluate the model, we run the prediction on the test dataset. In this section we present the high-level results of the test process.

Table 1 presents a summary of the DNN prediction. %95 of the predicted fractures are true fractures. and 77% of the prediction are True negative. Overall the F1-score is 85%.

	Precision	Recall	F1-score	Support
<b>0 : Fracture</b>	0.95	0.77	0.85	162
<b>1: No fracture</b>	0.76	0.95	0.84	124
<b>Micro Avg</b>	0.85	0.85	0.85	286
<b>Macro Avg</b>	0.86	0.86	0.85	286
<b>Weighted Avg</b>	0.87	0.85	0.85	286

Table 1: Classification report

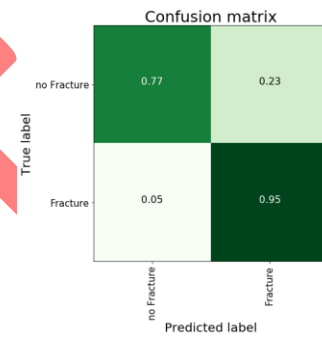


Figure 7: Confusion Matrix

- **Evaluation:**

To illustrate the findings of the DNN in identifying fractures from images. We import intermediate weights from the network to highlight the model attention in identifying fractures.

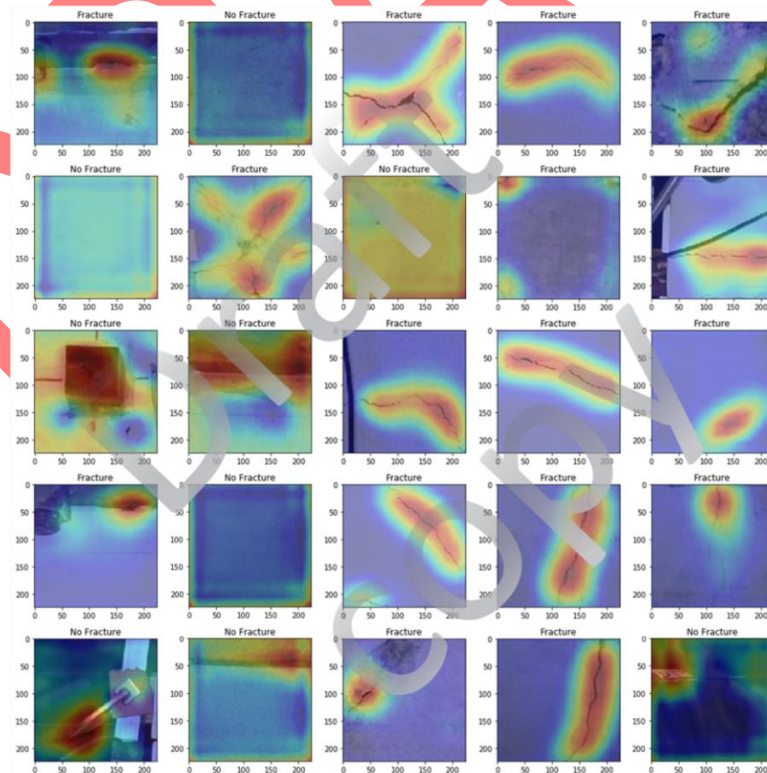
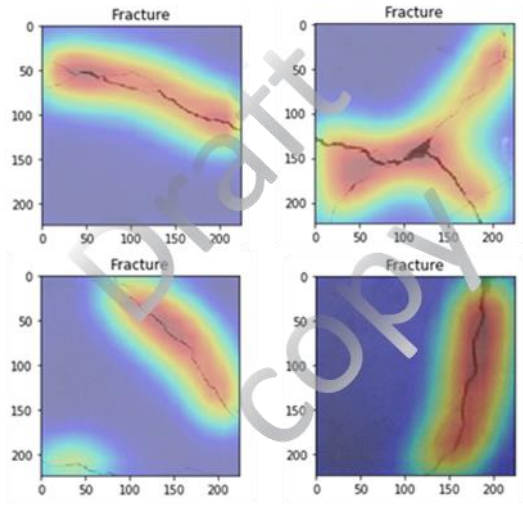
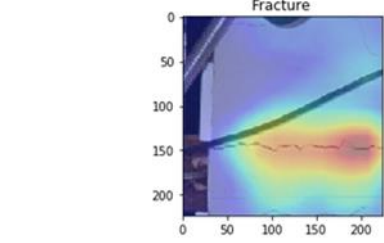
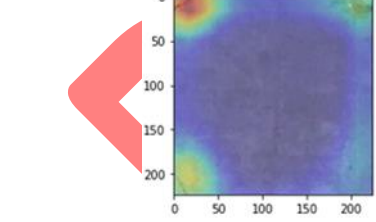
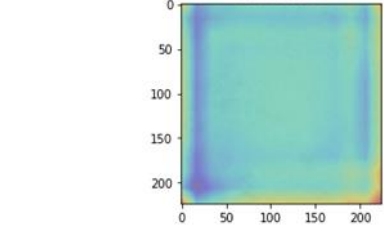


Figure 8: DNN prediction with intermediate trained weights overlapped on top of the images to highlight the model attention to fractures.

 <p>Figure 9: Model predictions – clear fractures</p>	<p>Figure 9 presents clear example of fractures identified by the model. In this application, the model was able to identify all the clear fractures introduced at the test stage. The calibrated weights of the hidden layers are overlapped on top of the images to highlight the fractures and the model attention. Clearly the model was able to identify the exact location as well as the orientation of the fractures. The intensity of the weights / “model-attention” is strongly correlated with the density/width of the fracture. This workflow provides a quantitative approach to identify and evaluation fractures.</p>
 <p>Figure 10: Model predictions: fracture with additional objects in the image</p>	<p>Figure 10 highlights a case where we add additional objects, a dark wire, next to the fracture. the model was successful in identifying the ‘actual fracture’ and disregarding the ‘fracture-lookalike’ object. The overlapped attention/weights clearly highlight the fracture location.</p>
 <p>Figure 11: Model prediction - fracture at the corner of the image</p>	<p>Figure 11 highlights fractures identified at the corner of the image. Despite the odd locations of the fractures, the model was able to successfully identify the fractures.</p>
 <p>Figure 12: Model prediction - no fracture</p>	<p>Figure 12 presents a case where there is no fracture. Clearly the model was able to recognize the absence of fractures.</p>



- **Fracture width from DNN weights:**

Here we illustrate the correlation between the fracture width and the VGG19 attention (calibrated weights).

In Figure 13 A) the neural network attentions are overlaid on top of the image to highlight the network attention to fracture. The fracture width is highlighted in Figure 13 B). A measurement of the normalized fracture width and the normalized VGG19 attention (calibrated weights) are shown in figure 13 C) and D) respectively. The correlation between the normalized neural network attention and the actual fracture width is clearly visible. The red and the blue dashed boxes highlight two areas where the normalized fracture width and the normalized neural network attention are in good agreement for the case of “No-fracture” and the case of “Maximum-fracture-width” respectively.

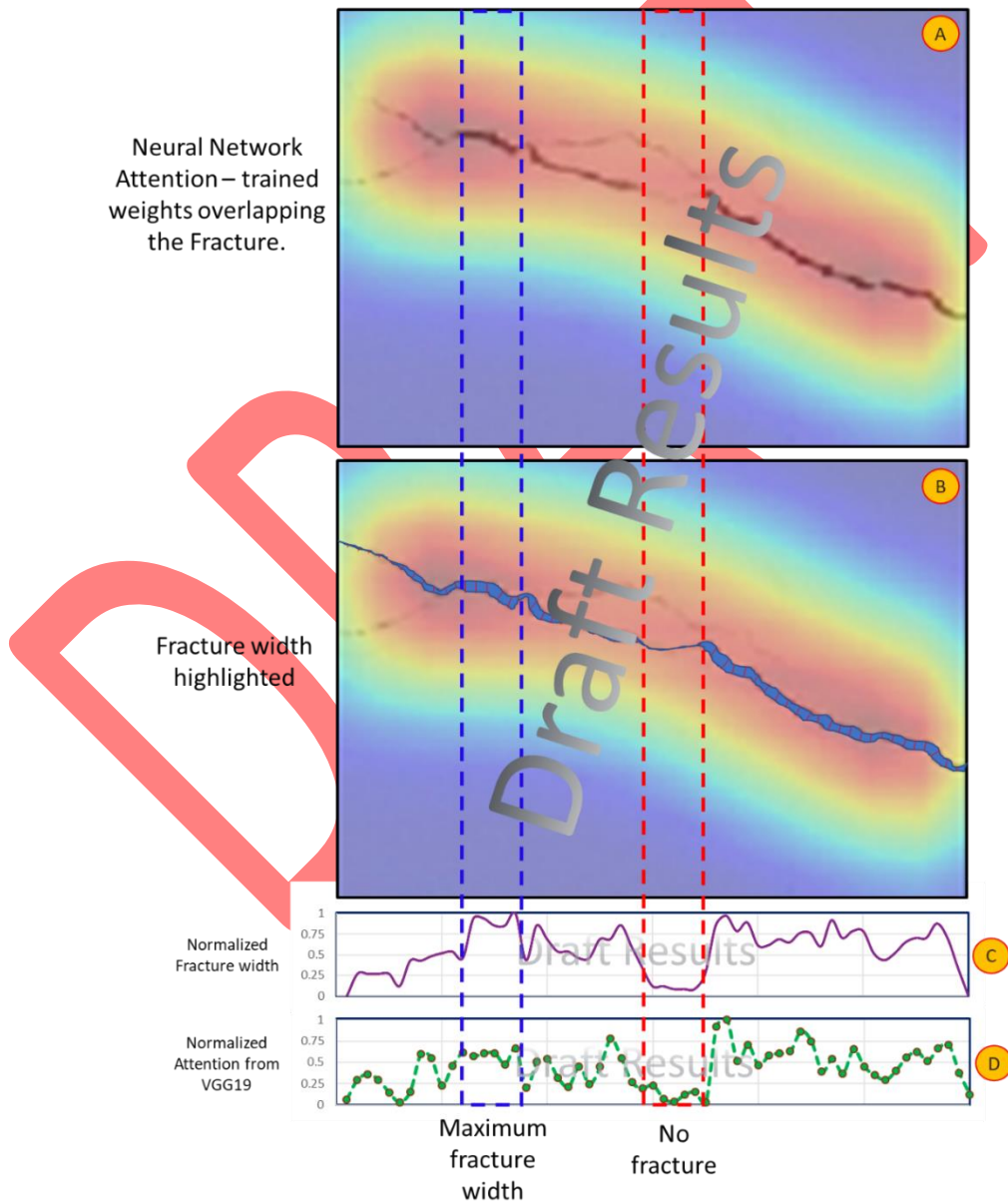


Figure 13: Quantitative Fracture width characterization using the calibrated weights from the Neural Network.  
C) Normalized fracture width, D) Normalized calibrated weights from VGG19.