1. TABLAS

TABLA I

Se adjunta el código de programación del último experimento con mejores resultados (Experimento 4).

```
Created on Sat Oct 10 21:54:22 2020


@author: pilar
"""
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import shutil
import os
print(os.listdir("../cell_images/"))
base_dir='../cell_images/'
work_dir=  "work/"
base_dir_A ='../cell_images/Parasitized/'
base_dir_B ='../cell_images/Uninfected/'
work_dir_A = "work/A/"
work_dir_B = "work/B/"
train_dir = os.path.join(work_dir, 'train')
validation_dir = os.path.join(work_dir, 'validation')
test_dir = os.path.join(work_dir, 'test')


train_pos_dir = os.path.join(train_dir, 'pos')
train_neg_dir = os.path.join(train_dir, 'neg')
validation_pos_dir = os.path.join(validation_dir, 'pos')
validation_neg_dir = os.path.join(validation_dir, 'neg')
test_pos_dir = os.path.join(test_dir, 'pos')
```

```
test_neg_dir = os.path.join(test_dir, 'neg')
fnames = ['pos{}.jpg'.format(i) for i in range(3000)]
for fname in fnames:
    src = os.path.join(work_dir_A, fname)
    dst = os.path.join(train_pos_dir, fname)
fnames = ['pos{}.jpg'.format(i) for i in range(3000, 4000)]
for fname in fnames:
    src = os.path.join(work_dir_A, fname)
    dst = os.path.join(validation_pos_dir, fname)
fnames = ['pos{}.jpg'.format(i) for i in range(4000, 4009)]
for fname in fnames:
    src = os.path.join(work_dir_A, fname)
    dst = os.path.join(test_pos_dir, fname)
fnames = ['neg{}.jpg'.format(i) for i in range(3000)]
for fname in fnames:
    src = os.path.join(work_dir_B, fname)
    dst = os.path.join(train_neg_dir, fname)
fnames = ['neg{}.jpg'.format(i) for i in range(3000, 4000)]
for fname in fnames:
    src = os.path.join(work_dir_B, fname)
    dst = os.path.join(validation_neg_dir, fname)
fnames = ['neg{}.jpg'.format(i) for i in range(4000, 4009)]
for fname in fnames:
    src = os.path.join(work_dir_B, fname)
    dst = os.path.join(test_neg_dir, fname)
print("Train, validation, and test datasets split and ready for use")
print('total training pos images:', len(os.listdir(train_pos_dir)))
print('total training neg images:', len(os.listdir(train_neg_dir)))
print('total validation pos images:', len(os.listdir(validation_pos_dir)))
```

```
print('total validation neg images:', len(os.listdir(validation_neg_dir)))
print('total test pos images:', len(os.listdir(test_pos_dir)))
print('total test meg images:', len(os.listdir(test_neg_dir)))

from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(150, 150),
        batch_size=20,
        class_mode='binary')
validation_generator = test_datagen.flow_from_directory(
        validation_dir,target_size=(150, 150),
        batch_size=20,
        class_mode='binary')
print("Image preprocessing complete")
from keras import layers
from keras import models
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',
                        input_shape=(150, 150, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
```

```
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
model.summary()
from keras import optimizers
model.compile(loss='binary_crossentropy',
optimizer=optimizers.RMSprop(lr=1e-2),
metrics=['acc'])
print("Model created")
history = model.fit_generator(
      train_generator,
      steps_per_epoch=50,
      epochs=20,
      validation_data=validation_generator,
      validation_steps=70)
model.save('basic_malaria_pos_neg_v1.h5')
import matplotlib.pyplot as plt
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
```

```
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()


eval_datagen = ImageDataGenerator(rescale=1./255)
eval_generator = eval_datagen.flow_from_directory(
        test_dir,target_size=(150, 150),
        batch_size=20,
        class_mode='binary')
eval_generator.reset()
pred = model.predict_generator(eval_generator,1000,verbose=1)
print("Predictions finished")
import matplotlib.image as mpimg
for index, probability in enumerate(pred):
    image_path = test_dir + "/" +eval_generator.filenames[index]
    img = mpimg.imread(image_path)
    plt.imshow(img)
    print(eval_generator.filenames[index])
    if probability > 0.5:
        plt.title("%.2f" % (probability[0]*100) + "% B")
    else:
        plt.title("%.2f" % ((1-probability[0])*100) + "% A")
    plt.show()
```

TABLA II

Se adjuntan las tablas, donde se recogen los valores obtenidos de pérdidas y precisión de los diferentes sets de entrenamiento y validación en cada época de los experimentos 2, 3 y 4

Tabla 7: Experimento 2.

Epoch	Train loss	Train accuracy	Val loss	Val accuracy
1	0,6921	0,5280	0,6907	0,5130
2	0,6876	0,5510	0,6859	0,5000
3	0,6859	0,5510	0,6668	0,6065
4	0,6795	0,6095	0,6723	0,5960
5	0,6767	0,6015	0,6531	0,6070
6	0,6686	0,6330	0,6618	0,6135
7	0,6633	0,6205	0,6164	0,5960
8	0,6602	0,612	0,5919	0,6250
9	0,6453	0,6535	0,6561	0,6245
10	0,6433	0,6395	0,6568	0,6210
11	0,6307	0,6560	0,5976	0,6260
12	0,6347	0,6435	0,6412	0,6325
13	0,6182	0,6765	0,6759	0,6295
14	0,6188	0,6685	0,6186	0,6445
15	0,6239	0,6600	0,6195	0,6395
16	0,6105	0,6785	0,6691	0,6565
17	0,6071	0,6865	0,5008	0,6510
18	0,6099	0,6750	0,7237	0,6450
19	0,5935	0,6980	0,5323	0,6470
20	0,5991	0,6830	0,6428	0,6535
21	0,5998	0,6945	0,5668	0,6620
22	0,6019	0,6890	0,6759	0,6475
23	0,5834	0,7025	0,6974	0,6590
24	0,5942	0,6870	0,6834	0,6630
25	0,5814	0,7025	0,5852	0,6395
26	0,5907	0,7070	0,5628	0,6660
27	0,5854	0,7090	0,5665	0,6670
28	0,5780	0,7070	0,5326	0,6740
29	0,5747	0,7205	0,5113	0,6750
30	0,5855	0,7075	0,5692	0,6740

Tabla 8: Experimento 3.

Epoch	Train loss	Train accuracy	Val loss	Val accuracy
1	0,7287	0,5380	0,6341	0,4990
2	0,6454	0,6400	0,6723	0,8035
3	0,3052	0,8905	0,2660	0,9310

4	0,2365	0,9265	0,2344	0,9385
5	0,2096	0,9300	0,5844	0,9495
6	0,2446	0,9340	0,2507	0,9520
7	0,1858	0,9490	0,0386	0,9230
8	0,1996	0,9420	0,0178	0,9520
9	0,1997	0,9370	0,0983	0,9540
10	0,1508	0,9490	0,0406	0,9565
11	0,1925	0,9465	0,2090	0,9075
12	0,1853	0,9440	0,3140	0,9575
13	0,1726	0,9440	0,1268	0,9505
14	0,1863	0,9385	0,1150	0,9485
15	0,1603	0,9560	0,0159	0,9370
16	0,1774	0,9420	0,1992	0,9570
17	0,1566	0,9480	0,0639	0,9530
18	0,17	0,9515	0,2507	0,9600
19	0,1583	0,9480	0,0268	0,9520
20	0,1891	0,9490	0,0281	0,9545
21	0,1585	0,9500	0,0480	0,9570
22	0,1631	0,9520	0,0182	0,9555
23	0,1732	0,9485	0,1021	0,9595
24	0,17	0,9405	0,0612	0,9595
25	0,164	0,9530	0,1918	0,9560
26	0,1633	0,9465	0,0718	0,9580
27	0,1493	0,9555	0,2553	0,9580
28	0,1528	0,9555	0,0287	0,9575
29	0,1794	0,9495	0,0454	0,9570
30	0,1762	0,9450	0,0789	0,9595
31	0,1745	0,9440	0,1555	0,9525
32	0,1799	0,9495	0,0243	0,9540
33	0,1579	0,9545	0,0996	0,9575
34	0,1535	0,9545	0,7407	0,8670
35	0,1533	0,9535	0,0773	0,9555
36	0,1832	0,9485	0,0394	0,9545
37	0,1581	0,9530	0,1243	0,9595
38	0,1578	0,9500	0,1077	0,9560
39	0,163	0,9505	0,2179	0,9565
40	0,1616	0,9540	0,1286	0,9570
41	0,1415	0,9565	0,1147	0,9570
42	0,2003	0,9455	0,1736	0,9555

43	0,1588	0,9515	0,0123	0,9605
44	0,1605	0,9550	0,1845	0,9515
45	0,1609	0,9490	0,5787	0,9565
46	0,1719	0,9565	0,0313	0,9610
47	0,1428	0,9560	0,3397	0,9530
48	0,1734	0,9445	0,0214	0,9540
49	0,1586	0,9485	0,0499	0,9600
50	0,1556	0,9575	0,1928	0,9475
51	0,1663	0,9540	0,1478	0,9585
52	0,1737	0,9470	0,0292	0,963
53	0,1390	0,9590	0,1136	0,9535
54	0,1585	0,9505	0,2279	0,9485
55	0,1746	0,9480	0,4592	0,9505
56	0,1409	0,9590	1,7304	0,9570
57	0,1669	0,9530	0,5319	0,9560
58	0,1508	0,9545	0,0229	0,9485
59	0,1863	0,9505	0,2502	0,9525
60	0,1762	0,9530	0,0362	0,961
61	0,1537	0,9505	0,0159	0,9615
62	0,1598	0,9510	0,1054	0,9610
63	0,1577	0,9530	0,3504	0,9565
64	0,1716	0,9530	0,1145	0,9570
65	0,1477	0,9525	0,2757	0,9600
66	0,1757	0,9535	0,1636	0,9550
67	0,1804	0,9510	0,2390	0,9590
68	0,1471	0,9540	0,0810	0,9590
69	0,1613	0,9540	0,0228	0,9605
70	0,1574	0,9530	0,2012	0,9565

Tabla 9: Experimento 4

Epoch	Train loss	Train accuracy	Val loss	Val accuracy
1	0,6081	0,6635	0,1161	0,922
2	0,2336	0,9193	0,5504	0,9010
3	0,2344	0,9300	0,1125	0,9555
4	0,1839	0,9398	0,1391	0,9490
5	0,1871	0,9438	0,0473	0,9545
6	0,1831	0,9460	0,4826	0,9535
7	0,1849	0,9415	0,0999	0,9570
8	0,1575	0,9550	0,0119	0,9525

9	0,1825	0,9460	0,1534	0,9435
10	0,1671	0,9457	0,4200	0,9585
11	0,1575	0,9490	0,1270	0,9335
12	0,1540	0,9538	0,0835	0,9565
13	0,1777	0,954	0,0450	0,9510
14	0,1690	0,9492	0,1418	0,9505
15	0,1571	0,9525	0,035p	0,9535
16	0,1587	0,9525	0,0094	0,955
17	0,1470	0,9548	0,2249	0,9395
18	0,1609	0,9498	0,3484	0,9515
19	0,1624	0,9532	0,0504	0,9595
20	0,1416	0,9575	0,0157	0,9565
21	0,1584	0,9542	0,4170	0,9475
22	0,1536	0,9530	0,1220	0,9585
23	0,1413	0,9538	0,0088	0,9545
24	0,1533	0,9553	0,0295	0,9565
25	0,1409	0,9588	0,0211	0,9415
26	0,1619	0,9540	0,4245	0,9580
27	0,1372	0,9563	0,1799	0,9455
28	0,1389	0,9557	0,3206	0,9485
29	0,1447	0,9553	0,0618	0,9525
30	0,155	0,9565	0,1640	0,9565
31	0,1366	0,9572	0,1508	0,9585
32	0,1513	0,9555	0,1742	0,9625
33	0,1326	0,9615	0,0873	0,9600
34	0,1323	0,9588	0,1537	0,9615
35	0,1374	0,9628	0,0122	0,9335
36	0,1300	0,9597	0,3283	0,9585
37	0,1227	0,9625	0,0097	0,9595
38	0,1271	0,9613	0,008	0,9535
39	0,1404	0,9607	0,1454	0,9535
40	0,1384	0,9570	0,9657	0,954
41	0,1581	0,958	0,4797	0,9575
42	0,1216	0,9628	0,1865	0,9400
43	0,1266	0,9600	0,0474	0,9545
44	0,1327	0,9603	0,1273	0,9480
45	0,1179	0,9640	0,5137	0,9600
46	0,1274	0,9660	0,2586	0,9185
47	0,1586	0,9607	0,0093	0,9510

48	0,1280	0,9628	0,0656	0,9500
49	0,1239	0,9638	0,0175	0,9620
50	0,1292	0,9597	0,1770	0,958

TABLA III

Se adjuntan las tablas donde se recogen los valores obtenidos de pérdidas y precisión de los sets de entrenamiento y validación en cada época del experimento de COVID-19.

Tabla 10: Experimento COVID-19.

Epoch	Train loss	Train accuracy	Val loss	Val accuracy
1	0,540400	0,800500	0,636800	0,589700
2	0,206200	0,934500	0,004000	0,986800
3	0,222700	0,959000	0,007200	0,973700
4	0,237700	0,972500	0,007600	0,979500
5	0,036800	0,989000	0,000000	0,986800
6	0,074300	0,982000	0,003200	0,986500
7	0,034200	0,988500	0,000602	0,996700
8	0,096800	0,987000	0,000530	0,980000
9	0,065100	0,989500	0,493400	0,923300
10	0,034700	0,991500	0,892100	0,966300
11	0,054600	0,991000	0,000077	0,986500
12	0,069200	0,993000	0,000680	0,993500
13	0,023600	0,994000	0,000000	0,956700
14	0,089900	0,992500	0,817900	0,990200
15	0,035800	0,996000	0,115800	0,990000
16	0,010300	0,998500	0,000002	0,990000
17	0,016600	0,997500	0,000003	0,996500
18	0,033300	0,995500	0,000045	0,993300
19	0,029600	0,996000	0,000010	0,993300
20	0,016600	0,998000	0,000000	0,976800
21	0,040000	0,994000	0,000016	0,979300
22	0,038400	0,997000	0,000018	0,982700
23	0,020300	0,996500	0,000000	0,990200
24	0,023900	0,997000	0,006900	0,983300
25	0,055500	0,996500	0,933700	0,960000
26	0,008200	0,998500	0,000013	0,993300
27	0,000002	1,000000	0,607400	0,987000
28	0,017300	0,997000	0,000000	0,989500
29	0,027800	0,995500	0,000000	0,993700
30	0,000036	1,000000	0,000000	0,993300