animalclassification

September 5, 2024

[40]:
```
import numpy as py
import tensorflow as tf
from tensorflow.keras.models import Sequential #object to create step-by-step␣
 ↪model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten
from tensorflow import keras

import random
import matplotlib.pyplot as plt

#Reference: Jay 'Coding Lane' Patel (https://www.youtube.com/@CodingLane), see␣
 ↪series on CNN and development of codel
```

[79]:
```
xtrain = py.loadtxt('input.csv', delimiter = ',')
ytrain = py.loadtxt('labels.csv', delimiter = ',')
xtest = py.loadtxt('input_test.csv', delimiter = ',')
ytest = py.loadtxt('labels_test.csv', delimiter = ',')
```

[81]:
```
xtrain.shape,ytrain.shape,xtest.shape,ytest.shape #total,␣
 ↪100(vertical)*100(horizontal)*3(rgb)
```

[81]:
```
((2000, 30000), (2000,), (400, 30000), (400,))
```

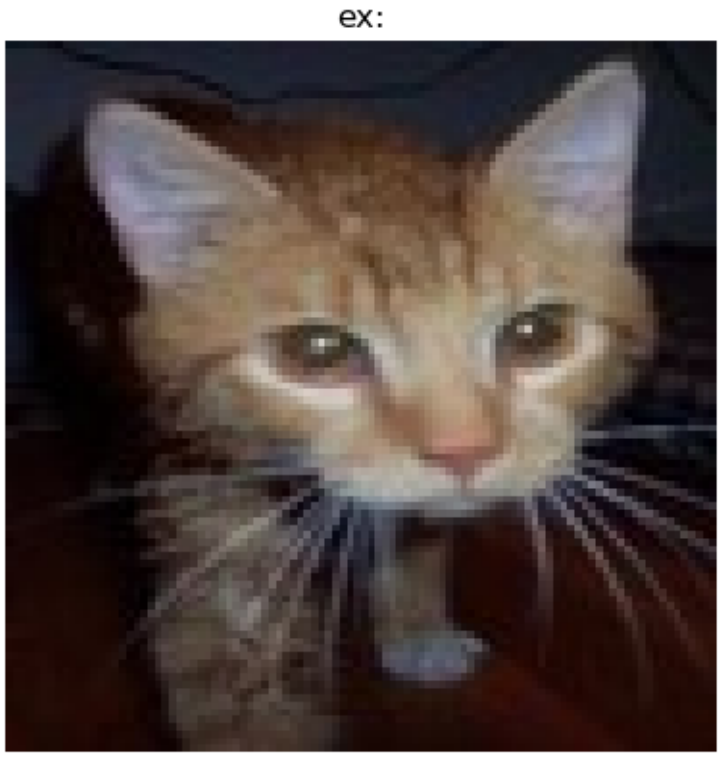
[83]:
```
#reshape images to be a 100*100 image w/ 3 rgb values per pixel
xtrain = xtrain.reshape(len(xtrain), 100, 100, 3)
ytrain = ytrain.reshape(len(ytrain), 1)
xtest = xtest.reshape(len(xtest), 100, 100, 3)
ytest = ytest.reshape(len(ytest), 1)

xtrain = xtrain/255.0
xtest = xtest/255.0 #normalize pixels w/ numpy
```

[85]:
```
xtrain.shape,ytrain.shape,xtest.shape,ytest.shape
```

[85]:
```
((2000, 100, 100, 3), (2000, 1), (400, 100, 100, 3), (400, 1))
```

[87]:
```
idx = random.randint(0, len(xtrain))
plt.imshow(xtrain[idx, :])
```

```
plt.axis('off')
plt.title('ex:')
plt.show()
```



[89]:
```
#num of filters 32, size (3,3)
model = Sequential([
    #convolution
    Conv2D(32, (3,3), activation = 'relu', input_shape = (100,100,3)),
    MaxPooling2D((2,2)), #default value 2
    Conv2D(32, (3,3), activation = 'relu'),
    MaxPooling2D((2,2)),

    #neural net
    Flatten(), #turn into inputs for NN
    Dense(64, activation = 'relu'), #fully connected layer of nodes
    Dense(1, activation = 'sigmoid') #binary classification
])
```

[91]:
```
opt = keras.optimizers.SGD(learning_rate=0.001) #specify SGD learning rate,␣
 ↪hyperparam
model.compile(loss = 'binary_crossentropy', optimizer = opt, metrics =␣
 ↪['accuracy'])
```

[93]:
```
model.fit(xtrain, ytrain, epochs = 25, batch_size = 64)
```

```
Epoch 1/25
32/32             2s 42ms/step -
accuracy: 0.5155 - loss: 0.6925
Epoch 2/25
32/32             1s 42ms/step -
accuracy: 0.5296 - loss: 0.6910
Epoch 3/25
32/32             1s 42ms/step -
accuracy: 0.5603 - loss: 0.6890
Epoch 4/25
32/32             1s 43ms/step -
accuracy: 0.5599 - loss: 0.6872
Epoch 5/25
32/32             1s 42ms/step -
accuracy: 0.5491 - loss: 0.6868
Epoch 6/25
32/32             1s 42ms/step -
accuracy: 0.5444 - loss: 0.6881
Epoch 7/25
32/32             1s 42ms/step -
accuracy: 0.5325 - loss: 0.6877
Epoch 8/25
32/32             1s 43ms/step -
accuracy: 0.5670 - loss: 0.6856
Epoch 9/25
32/32             1s 42ms/step -
accuracy: 0.5722 - loss: 0.6853
Epoch 10/25
32/32             1s 41ms/step -
accuracy: 0.5856 - loss: 0.6841
Epoch 11/25
32/32             1s 41ms/step -
accuracy: 0.5464 - loss: 0.6852
Epoch 12/25
32/32             1s 41ms/step -
accuracy: 0.5954 - loss: 0.6818
Epoch 13/25
32/32             1s 41ms/step -
accuracy: 0.5693 - loss: 0.6824
Epoch 14/25
32/32             1s 41ms/step -
accuracy: 0.5717 - loss: 0.6820
Epoch 15/25
32/32             1s 43ms/step -
accuracy: 0.5901 - loss: 0.6801
Epoch 16/25
```

```
32/32             1s 41ms/step -
accuracy: 0.5716 - loss: 0.6810
Epoch 17/25
32/32             1s 41ms/step -
accuracy: 0.5728 - loss: 0.6811
Epoch 18/25
32/32             1s 43ms/step -
accuracy: 0.5961 - loss: 0.6786
Epoch 19/25
32/32             1s 43ms/step -
accuracy: 0.5848 - loss: 0.6795
Epoch 20/25
32/32             1s 43ms/step -
accuracy: 0.5996 - loss: 0.6774
Epoch 21/25
32/32             1s 41ms/step -
accuracy: 0.5975 - loss: 0.6796
Epoch 22/25
32/32             1s 41ms/step -
accuracy: 0.5911 - loss: 0.6789
Epoch 23/25
32/32             1s 41ms/step -
accuracy: 0.5801 - loss: 0.6805
Epoch 24/25
32/32             1s 41ms/step -
accuracy: 0.5706 - loss: 0.6822
Epoch 25/25
32/32             1s 42ms/step -
accuracy: 0.5981 - loss: 0.6751
```

[93]:
```
<keras.src.callbacks.history.History at 0x24231f5f2c0>
```

[49]:
```
model.evaluate(xtest, ytest)
```

```
13/13             0s 11ms/step -
accuracy: 0.7811 - loss: 0.5923
```

[49]:
```
[0.6783967018127441, 0.5649999976158142]
```

[50]:
```
fig, axs = plt.subplots(2, 5, figsize=(15, 6))

fig.suptitle('Epochs = 25', fontsize=16)

for i in range(2):
    for j in range(5):
        idx = random.randint(0, len(ytest))

        ypred = model.predict(xtest[idx, :].reshape(1, 100, 100, 3))
```

```
        if ypred > 0.5:
            pred = 'cat'
        else:
            pred = 'dog'

        axs[i, j].imshow(xtest[idx])
        axs[i, j].set_title('Prediction: ' + pred)
        axs[i, j].axis('off')

plt.tight_layout()
plt.show()
```

```
1/1             0s 55ms/step
1/1             0s 19ms/step
1/1             0s 18ms/step
1/1             0s 17ms/step
1/1             0s 15ms/step
1/1             0s 17ms/step
1/1             0s 17ms/step
1/1             0s 18ms/step
1/1             0s 17ms/step
1/1             0s 17ms/step
```

[ ]: