Data preparation, exploration, visualization

Overall, from lines 35-43 was the EDA in order to split the training data from the test data in the set. Because this dataset had to do with images, rather than text data, I first plotted a test image in line 39 and in line 40, resized the data. I then created a label for “dogs” and “cats” in line 42 and found that there were an equal amount of dog labels to cat labels (2000 each). After doing this initial EDA process, I continued to go over the modeling methods for this assignment.

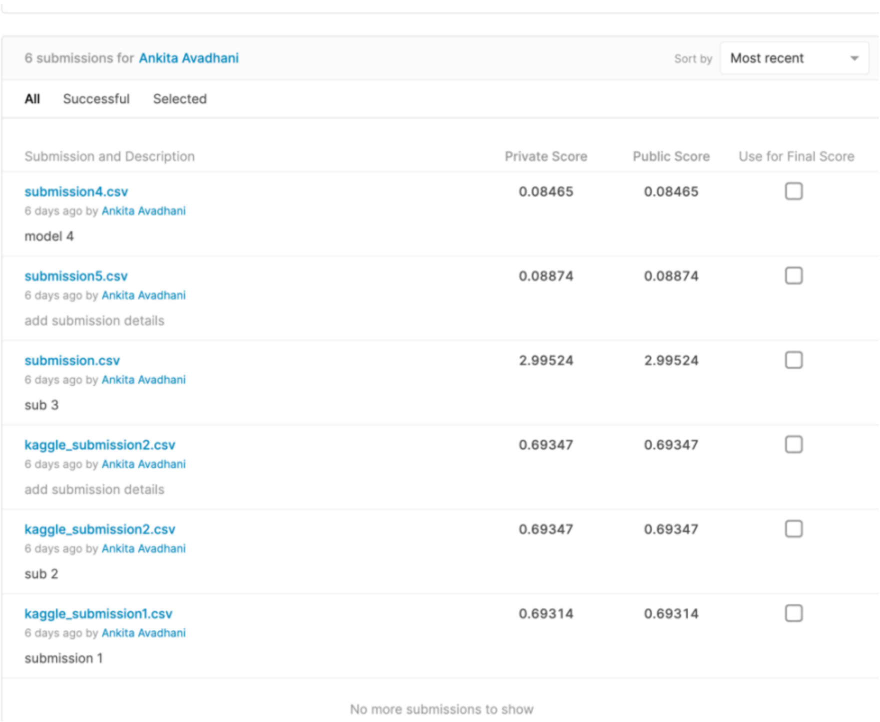
Review research design and modeling methods

In total, I used four different models in order forecast the training set for the test set. In model 1, I used 64 epochs and a CNN in line 54. In line 109, I plotted the accuracy and loss as well. For model 2, I used 6 layers with dropout regularization as well as normalization in line 35. I plotted the same accuracy and loss chart in line 138. Model 3 had 8 layers and 20 epochs as shown in line 159 and model 4 had 8 layers with dropout regularization and batch normalization. Model 3 results were plotted in line 164 and model 4 was plotted in line 193. Overall, the models were all variations of CNN’s with a combination of testing dropout regularizations. I found like unlike previous assignments, there was a large variation in terms of how accurate each type of model was. The number of epochs and layers truly made a difference when using CNN’s to predict the test set from the training set.

Review results, evaluate models

Overall model 1 yielded an accuracy of .807 with a loss of .445, model 2 yielded an accuracy of 0.8602 with a value loss of 0.41875, model 3 yielded an accuracy of 0.8569 and a loss of 0.4179 and lastly, model 4 0.86 with a value loss of 0.412. In the end, I would choose to use model 4 which would be 8 layers with dropout regularization and batch normalization.

Implementation and programming as evidenced by Kaggle.com submissions



6 submissions for Ankita Avadhani

Sort by Most recent

All Successful Selected

Submission and Description	Private Score	Public Score	Use for Final Score
submission4.csv 6 days ago by Ankita Avadhani model 4	0.08465	0.08465	□
submission5.csv 6 days ago by Ankita Avadhani add submission details	0.08874	0.08874	□
submission.csv 6 days ago by Ankita Avadhani sub 3	2.99524	2.99524	□
kaggle_submission2.csv 6 days ago by Ankita Avadhani add submission details	0.69347	0.69347	□
kaggle_submission2.csv 6 days ago by Ankita Avadhani sub 2	0.69347	0.69347	□
kaggle_submission1.csv 6 days ago by Ankita Avadhani submission 1	0.69314	0.69314	□

No more submissions to show

Since this competition already passed, these were my submission scores for the competition. They are also commented in the code document below. With the best score being 0.08465.

Exposition, problem description, and management recommendations

If I had to manage the problem of labeling images from end users for a website and choose the most accurate model, I would choose model 4, a CNN with 8 layers with dropout regularization and batch normalization. This is due to the high accuracy and low value loss. The dataset for this assignment was 2000 colored pictures of dogs and cats. By transforming these pictures into an array and then changing the layers (each layer representing a color/greyscale for an image), I was able to create a model that was accurate for prediction. Because model 4 was the most accurate with 8 layers, I believe that the best images to start the data set with are images 32x32 size with three color channels (red, blue and green). While it takes more memory to process this type of model, it yields the most accurate results which is what we want for this problem.

In [35]:

```
%matplotlib inline

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import seaborn as sns
from datetime import datetime
import os
import random
import gc
import cv2
from sklearn.model_selection import train_test_split

import tensorflow as tf
from tensorflow import keras
```

In [36]:

```
train_dogs = ['train/{}'.format(i) for i in os.listdir('train') if 'dog' in i]
train_cats = ['train/{}'.format(i) for i in os.listdir('train') if 'cat' in i]
```

In [37]:

```
df_test = ['test/{}'.format(i) for i in os.listdir('test')]
df_train = train_dogs[:2000] + train_cats[:2000]
```

In [38]:

```
del train_dogs
del train_cats

gc.collect()
```

Out[38]:

```
88
```

In [39]:

```
plt.imshow(mpimg.imread(df_train[0]))
```

Out[39]:

```
<matplotlib.image.AxesImage at 0x13255a5f8>
```

In [ ]:

```

```

In [40]:

```
size = 64
channels = 3
X = []
y = []

for pic in df_train:
        X.append(cv2.resize(cv2.imread(pic, cv2.IMREAD_COLOR), (size , size), interpolation=cv2.INTER_CUBIC))
        y.append(1) if 'dog' in pic else y.append(0)
```

In [41]:

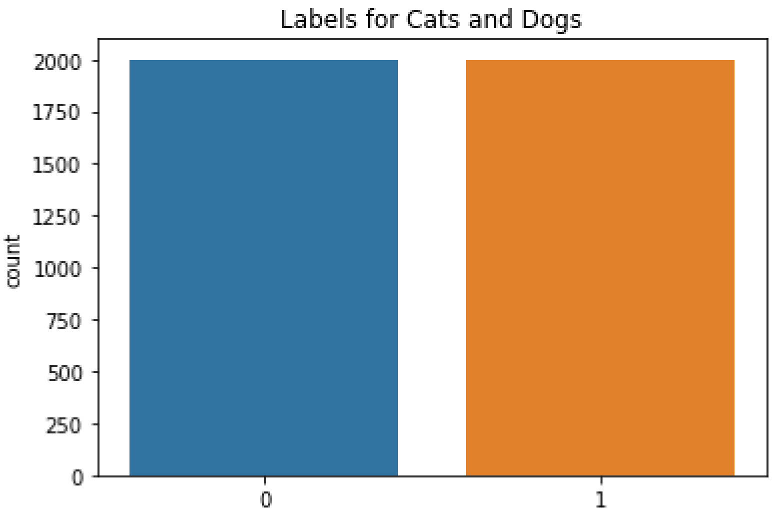
```
X = np.array(X)
y = np.array(y)
```

In [42]:

```
sns.countplot(y)
plt.title('Labels for Cats and Dogs')
```

Out[42]:

```
Text(0.5, 1.0, 'Labels for Cats and Dogs')
```



In [43]:

```
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.20, random_stat
e=42)
```

In [45]:

```
model1 = keras.models.Sequential()
model1.add(keras.layers.Conv2D(32, (3, 3), activation='relu',input_shape=(size, siz
e, channels)))
model1.add(keras.layers.MaxPooling2D((2, 2)))
model1.add(keras.layers.Conv2D(64, (3, 3), activation='relu'))
model1.add(keras.layers.MaxPooling2D((2, 2)))
model1.add(keras.layers.Conv2D(128, (3, 3), activation='relu'))
model1.add(keras.layers.MaxPooling2D((2, 2)))
model1.add(keras.layers.Flatten())
model1.add(keras.layers.Dropout(0.5))  #Dropout for regularization
model1.add(keras.layers.Dense(256, activation='relu'))
model1.add(keras.layers.Dense(1, activation='sigmoid'))
```

In [49]:

```
model1.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 62, 62, 32)        896       
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 31, 31, 32)        0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 29, 29, 64)        18496     
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 64)        0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 12, 12, 128)       73856     
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 128)         0         
_________________________________________________________________
flatten (Flatten)            (None, 4608)              0         
_________________________________________________________________
dropout (Dropout)            (None, 4608)              0         
_________________________________________________________________
dense (Dense)                (None, 256)               1179904   
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 257       
=================================================================
Total params: 1,273,409
Trainable params: 1,273,409
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:

```
#model 1 using 64 epochs and CNN
```

In [54]:

```
batch = 16
ntrain = len(X_train)
nval = len(X_val)

model1.compile(loss='binary_crossentropy', optimizer=keras.optimizers.RMSprop(lr=1e
-4), metrics=['acc'])

val_datagen = keras.preprocessing.image.ImageDataGenerator(rescale=1./255)

train_datagen = keras.preprocessing.image.ImageDataGenerator(rescale=1./255,
                                                             rotation_range=20,
                                                             zoom_range=0.1,
                                                             width_shift_range=0.1,
                                                             height_shift_range=0.1
,
                                                             shear_range=0.1,
                                                             horizontal_flip=True,
                                                             fill_mode="nearest")

train_generator = train_datagen.flow(X_train, y_train, batch_size=batch)

val_generator = val_datagen.flow(X_val, y_val, batch_size=batch)


m1_start = datetime.now()

history = model1.fit_generator(train_generator,
                              steps_per_epoch=ntrain // batch,
                              epochs=64,
                              validation_data=val_generator,
                              validation_steps=nval // batch)

m1_end = datetime.now()
```

```
Epoch 1/64
200/200 [==============================] - 13s 63ms/step - loss: 0.6935
- acc: 0.5337 - val_loss: 0.6795 - val_acc: 0.5325
Epoch 2/64
200/200 [==============================] - 12s 61ms/step - loss: 0.6862
- acc: 0.5541 - val_loss: 0.6607 - val_acc: 0.6338
Epoch 3/64
200/200 [==============================] - 12s 59ms/step - loss: 0.6704
- acc: 0.5931 - val_loss: 0.6371 - val_acc: 0.6825
Epoch 4/64
200/200 [==============================] - 14s 70ms/step - loss: 0.6520
- acc: 0.6187 - val_loss: 0.6114 - val_acc: 0.6850
Epoch 5/64
200/200 [==============================] - 14s 71ms/step - loss: 0.6408
- acc: 0.6316 - val_loss: 0.6058 - val_acc: 0.6888
Epoch 6/64
200/200 [==============================] - 13s 64ms/step - loss: 0.6321
- acc: 0.6459 - val_loss: 0.6030 - val_acc: 0.6850
Epoch 7/64
200/200 [==============================] - 16s 79ms/step - loss: 0.6142
- acc: 0.6656 - val_loss: 0.5782 - val_acc: 0.6950
Epoch 8/64
200/200 [==============================] - 14s 70ms/step - loss: 0.6138
- acc: 0.6684 - val_loss: 0.5818 - val_acc: 0.6938
Epoch 9/64
200/200 [==============================] - 12s 60ms/step - loss: 0.6090
- acc: 0.6725 - val_loss: 0.5885 - val_acc: 0.6963
Epoch 10/64
200/200 [==============================] - 13s 64ms/step - loss: 0.5960
- acc: 0.6828 - val_loss: 0.5591 - val_acc: 0.7163
Epoch 11/64
200/200 [==============================] - 10s 51ms/step - loss: 0.5858
- acc: 0.6934 - val_loss: 0.5591 - val_acc: 0.7138
Epoch 12/64
200/200 [==============================] - 11s 54ms/step - loss: 0.5792
- acc: 0.6919 - val_loss: 0.5458 - val_acc: 0.7237
Epoch 13/64
200/200 [==============================] - 12s 58ms/step - loss: 0.5753
- acc: 0.7041 - val_loss: 0.5554 - val_acc: 0.7150
Epoch 14/64
200/200 [==============================] - 12s 60ms/step - loss: 0.5684
- acc: 0.7069 - val_loss: 0.5271 - val_acc: 0.7387
Epoch 15/64
200/200 [==============================] - 14s 70ms/step - loss: 0.5614
- acc: 0.7150 - val_loss: 0.5324 - val_acc: 0.7450
Epoch 16/64
200/200 [==============================] - 10s 52ms/step - loss: 0.5562
- acc: 0.7237 - val_loss: 0.5462 - val_acc: 0.7200
Epoch 17/64
200/200 [==============================] - 10s 50ms/step - loss: 0.5545
- acc: 0.7194 - val_loss: 0.5365 - val_acc: 0.7188
Epoch 18/64
200/200 [==============================] - 10s 51ms/step - loss: 0.5524
- acc: 0.7200 - val_loss: 0.5654 - val_acc: 0.7050
Epoch 19/64
200/200 [==============================] - 10s 51ms/step - loss: 0.5395
- acc: 0.7375 - val_loss: 0.5299 - val_acc: 0.7287
```

```
Epoch 20/64
200/200 [==============================] - 10s 50ms/step - loss: 0.5375
- acc: 0.7353 - val_loss: 0.5126 - val_acc: 0.7425
Epoch 21/64
200/200 [==============================] - 10s 50ms/step - loss: 0.5267
- acc: 0.7375 - val_loss: 0.5348 - val_acc: 0.7163
Epoch 22/64
200/200 [==============================] - 10s 51ms/step - loss: 0.5366
- acc: 0.7344 - val_loss: 0.5335 - val_acc: 0.7212
Epoch 23/64
200/200 [==============================] - 10s 50ms/step - loss: 0.5238
- acc: 0.7416 - val_loss: 0.4891 - val_acc: 0.7638
Epoch 24/64
200/200 [==============================] - 10s 50ms/step - loss: 0.5211
- acc: 0.7472 - val_loss: 0.4794 - val_acc: 0.7650
Epoch 25/64
200/200 [==============================] - 11s 54ms/step - loss: 0.5222
- acc: 0.7431 - val_loss: 0.5039 - val_acc: 0.7525
Epoch 26/64
200/200 [==============================] - 10s 51ms/step - loss: 0.5139
- acc: 0.7481 - val_loss: 0.4819 - val_acc: 0.7650
Epoch 27/64
200/200 [==============================] - 10s 51ms/step - loss: 0.5171
- acc: 0.7472 - val_loss: 0.4799 - val_acc: 0.7725
Epoch 28/64
200/200 [==============================] - 10s 51ms/step - loss: 0.4988
- acc: 0.7575 - val_loss: 0.4897 - val_acc: 0.7688
Epoch 29/64
200/200 [==============================] - 10s 51ms/step - loss: 0.5071
- acc: 0.7584 - val_loss: 0.5099 - val_acc: 0.7513
Epoch 30/64
200/200 [==============================] - 10s 51ms/step - loss: 0.4985
- acc: 0.7619 - val_loss: 0.5050 - val_acc: 0.7487
Epoch 31/64
200/200 [==============================] - 13s 66ms/step - loss: 0.5042
- acc: 0.7506 - val_loss: 0.5190 - val_acc: 0.7450
Epoch 32/64
200/200 [==============================] - 14s 69ms/step - loss: 0.4921
- acc: 0.7622 - val_loss: 0.4823 - val_acc: 0.7650
Epoch 33/64
200/200 [==============================] - 13s 63ms/step - loss: 0.4923
- acc: 0.7616 - val_loss: 0.5016 - val_acc: 0.7613
Epoch 34/64
200/200 [==============================] - 11s 57ms/step - loss: 0.4859
- acc: 0.7663 - val_loss: 0.4752 - val_acc: 0.7638
Epoch 35/64
200/200 [==============================] - 13s 67ms/step - loss: 0.4835
- acc: 0.7741 - val_loss: 0.4627 - val_acc: 0.7825
Epoch 36/64
200/200 [==============================] - 14s 70ms/step - loss: 0.4786
- acc: 0.7722 - val_loss: 0.4665 - val_acc: 0.7837
Epoch 37/64
200/200 [==============================] - 13s 64ms/step - loss: 0.4715
- acc: 0.7784 - val_loss: 0.4804 - val_acc: 0.7750
Epoch 38/64
200/200 [==============================] - 13s 66ms/step - loss: 0.4745
- acc: 0.7734 - val_loss: 0.4678 - val_acc: 0.7750
```

```
Epoch 39/64
200/200 [==============================] - 12s 61ms/step - loss: 0.4594
- acc: 0.7862 - val_loss: 0.4526 - val_acc: 0.7825
Epoch 40/64
200/200 [==============================] - 11s 57ms/step - loss: 0.4630
- acc: 0.7794 - val_loss: 0.5168 - val_acc: 0.7563
Epoch 41/64
200/200 [==============================] - 12s 59ms/step - loss: 0.4550
- acc: 0.7937 - val_loss: 0.4660 - val_acc: 0.7725
Epoch 42/64
200/200 [==============================] - 14s 68ms/step - loss: 0.4616
- acc: 0.7859 - val_loss: 0.4555 - val_acc: 0.7775
Epoch 43/64
200/200 [==============================] - 12s 58ms/step - loss: 0.4609
- acc: 0.7806 - val_loss: 0.4898 - val_acc: 0.7638
Epoch 44/64
200/200 [==============================] - 11s 56ms/step - loss: 0.4583
- acc: 0.7884 - val_loss: 0.4931 - val_acc: 0.7550
Epoch 45/64
200/200 [==============================] - 12s 62ms/step - loss: 0.4518
- acc: 0.7866 - val_loss: 0.4769 - val_acc: 0.7700
Epoch 46/64
200/200 [==============================] - 12s 60ms/step - loss: 0.4577
- acc: 0.7831 - val_loss: 0.4523 - val_acc: 0.7925
Epoch 47/64
200/200 [==============================] - 24s 119ms/step - loss: 0.456
0 - acc: 0.7819 - val_loss: 0.4495 - val_acc: 0.7887
Epoch 48/64
200/200 [==============================] - 12s 61ms/step - loss: 0.4470
- acc: 0.7894 - val_loss: 0.4625 - val_acc: 0.7800
Epoch 49/64
200/200 [==============================] - 15s 74ms/step - loss: 0.4450
- acc: 0.7869 - val_loss: 0.4721 - val_acc: 0.7837
Epoch 50/64
200/200 [==============================] - 11s 57ms/step - loss: 0.4504
- acc: 0.7837 - val_loss: 0.5027 - val_acc: 0.7688
Epoch 51/64
200/200 [==============================] - 11s 56ms/step - loss: 0.4408
- acc: 0.7922 - val_loss: 0.4983 - val_acc: 0.7688
Epoch 52/64
200/200 [==============================] - 11s 56ms/step - loss: 0.4350
- acc: 0.7922 - val_loss: 0.4591 - val_acc: 0.7850
Epoch 53/64
200/200 [==============================] - 11s 56ms/step - loss: 0.4455
- acc: 0.7919 - val_loss: 0.4648 - val_acc: 0.7788
Epoch 54/64
200/200 [==============================] - 12s 61ms/step - loss: 0.4309
- acc: 0.8034 - val_loss: 0.4791 - val_acc: 0.7925
Epoch 55/64
200/200 [==============================] - 11s 53ms/step - loss: 0.4346
- acc: 0.8000 - val_loss: 0.4737 - val_acc: 0.7788
Epoch 56/64
200/200 [==============================] - 10s 48ms/step - loss: 0.4436
- acc: 0.7984 - val_loss: 0.4894 - val_acc: 0.7775
Epoch 57/64
200/200 [==============================] - 12s 58ms/step - loss: 0.4268
- acc: 0.8097 - val_loss: 0.4673 - val_acc: 0.7812
```

```
Epoch 58/64
200/200 [==============================] - 11s 57ms/step - loss: 0.4265
- acc: 0.8012 - val_loss: 0.4707 - val_acc: 0.7675
Epoch 59/64
200/200 [==============================] - 12s 58ms/step - loss: 0.4106
- acc: 0.8103 - val_loss: 0.4775 - val_acc: 0.7862
Epoch 60/64
200/200 [==============================] - 11s 54ms/step - loss: 0.4318
- acc: 0.7994 - val_loss: 0.4485 - val_acc: 0.7875
Epoch 61/64
200/200 [==============================] - 10s 50ms/step - loss: 0.4221
- acc: 0.8031 - val_loss: 0.4746 - val_acc: 0.7825
Epoch 62/64
200/200 [==============================] - 11s 57ms/step - loss: 0.4109
- acc: 0.8153 - val_loss: 0.4601 - val_acc: 0.7850
Epoch 63/64
200/200 [==============================] - 11s 56ms/step - loss: 0.4240
- acc: 0.8016 - val_loss: 0.4622 - val_acc: 0.7825
Epoch 64/64
200/200 [==============================] - 11s 55ms/step - loss: 0.4252
- acc: 0.8075 - val_loss: 0.4450 - val_acc: 0.7850
```

In [61]:

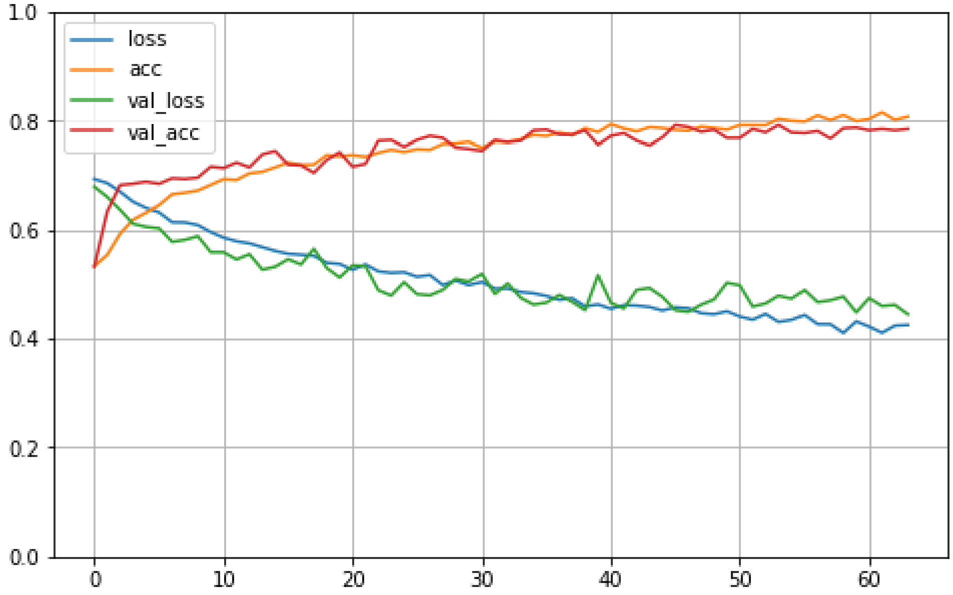
```
model1.save_weights('model1_weights.h5')
model1.save('model1_keras.h5')
```

In [71]:

```
X_test = []

for pic in df_test:
        X_test.append(cv2.resize(cv2.imread(pic, cv2.IMREAD_COLOR), (size , size),
interpolation=cv2.INTER_CUBIC))
x_test = np.array(X_test)
test_datagen = keras.preprocessing.image.ImageDataGenerator(rescale=1./255)

test_generator = test_datagen.flow(x_test, batch_size=1, shuffle=False)
test_datagen = keras.preprocessing.image.ImageDataGenerator(rescale=1./255)
predictions1 = model1.predict(test_generator)



columns = 5
```

In [109]:

```
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1) # set the vertical range to [0-1]
plt.show()
```



In [72]:

```
predictions1 = model1.predict(test_generator)
```

In [73]:

```
submission1 = pd.read_csv('sample_submission.csv')
kaggle_submission1 =submission1.to_csv(r'kaggle_submission1.csv', index=False)
```

In [74]:

```
#kaggle score 0.69314, Ankita Avadhani
```

In [ ]:

```
# Model 2: 6 layers with dropout regularization and batch normalization
# Model definition
```

In [135]:

```
model2 = keras.models.Sequential()
model2.add(keras.layers.Conv2D(32, (3, 3), activation='relu',input_shape=(size, siz
e, channels)))
model2.add(keras.layers.MaxPooling2D((2, 2)))
model2.add(keras.layers.Conv2D(10, (3, 3), activation='relu'))
model2.add(keras.layers.MaxPooling2D((2, 2)))
model2.add(keras.layers.Conv2D(128, (3, 3), activation='relu'))
model2.add(keras.layers.MaxPooling2D((2, 2)))
model2.add(keras.layers.Flatten())
model2.add(keras.layers.Dropout(0.5))  #Dropout for regularization
model2.add(keras.layers.Dense(256, activation='relu'))
model2.add(keras.layers.Dense(1, activation='sigmoid'))
```

In [136]:

```
batch = 16
ntrain = len(X_train)
nval = len(X_val)

model2.compile(loss='binary_crossentropy', optimizer=keras.optimizers.RMSprop(lr=1e
-4), metrics=['acc'])

val_datagen = keras.preprocessing.image.ImageDataGenerator(rescale=1./255)

train_datagen = keras.preprocessing.image.ImageDataGenerator(rescale=1./255,
                                                             rotation_range=20,
                                                             zoom_range=0.1,
                                                             width_shift_range=0.1,
                                                             height_shift_range=0.1
,
                                                             shear_range=0.1,
                                                             horizontal_flip=True,
                                                             fill_mode="nearest")

train_generator = train_datagen.flow(X_train, y_train, batch_size=batch)

val_generator = val_datagen.flow(X_val, y_val, batch_size=batch)


m2_start = datetime.now()

history = model1.fit_generator(train_generator,
                              steps_per_epoch=ntrain // batch,
                              epochs=64,
                              validation_data=val_generator,
                              validation_steps=nval // batch)

m2_end = datetime.now()
```

```
Epoch 1/64
200/200 [==============================] - 11s 54ms/step - loss: 0.4146
- acc: 0.8091 - val_loss: 0.4482 - val_acc: 0.7925
Epoch 2/64
200/200 [==============================] - 11s 55ms/step - loss: 0.4215
- acc: 0.8019 - val_loss: 0.4791 - val_acc: 0.7775
Epoch 3/64
200/200 [==============================] - 11s 55ms/step - loss: 0.4014
- acc: 0.8116 - val_loss: 0.4361 - val_acc: 0.7962
Epoch 4/64
200/200 [==============================] - 11s 56ms/step - loss: 0.3971
- acc: 0.8138 - val_loss: 0.4508 - val_acc: 0.7850
Epoch 5/64
200/200 [==============================] - 10s 51ms/step - loss: 0.4111
- acc: 0.8069 - val_loss: 0.4568 - val_acc: 0.7825
Epoch 6/64
200/200 [==============================] - 10s 51ms/step - loss: 0.4106
- acc: 0.8122 - val_loss: 0.4386 - val_acc: 0.8062
Epoch 7/64
200/200 [==============================] - 12s 60ms/step - loss: 0.3984
- acc: 0.8188 - val_loss: 0.4765 - val_acc: 0.7763
Epoch 8/64
200/200 [==============================] - 13s 63ms/step - loss: 0.4094
- acc: 0.8106 - val_loss: 0.4452 - val_acc: 0.7937
Epoch 9/64
200/200 [==============================] - 11s 55ms/step - loss: 0.3932
- acc: 0.8175 - val_loss: 0.4554 - val_acc: 0.7825
Epoch 10/64
200/200 [==============================] - 11s 55ms/step - loss: 0.3948
- acc: 0.8272 - val_loss: 0.4520 - val_acc: 0.7837
Epoch 11/64
200/200 [==============================] - 11s 56ms/step - loss: 0.3976
- acc: 0.8147 - val_loss: 0.4428 - val_acc: 0.7962
Epoch 12/64
200/200 [==============================] - 11s 57ms/step - loss: 0.4036
- acc: 0.8191 - val_loss: 0.4296 - val_acc: 0.7987
Epoch 13/64
200/200 [==============================] - 13s 66ms/step - loss: 0.3941
- acc: 0.8275 - val_loss: 0.4648 - val_acc: 0.7713
Epoch 14/64
200/200 [==============================] - 13s 65ms/step - loss: 0.3986
- acc: 0.8184 - val_loss: 0.4288 - val_acc: 0.8037
Epoch 15/64
200/200 [==============================] - 13s 64ms/step - loss: 0.3895
- acc: 0.8253 - val_loss: 0.4257 - val_acc: 0.8062
Epoch 16/64
200/200 [==============================] - 16s 82ms/step - loss: 0.3879
- acc: 0.8253 - val_loss: 0.4378 - val_acc: 0.8000
Epoch 17/64
200/200 [==============================] - 17s 83ms/step - loss: 0.3900
- acc: 0.8291 - val_loss: 0.4458 - val_acc: 0.7900
Epoch 18/64
200/200 [==============================] - 16s 80ms/step - loss: 0.3893
- acc: 0.8300 - val_loss: 0.4157 - val_acc: 0.8125
Epoch 19/64
200/200 [==============================] - 17s 84ms/step - loss: 0.3731
- acc: 0.8288 - val_loss: 0.4574 - val_acc: 0.7862
```

```
Epoch 20/64
200/200 [==============================] - 17s 84ms/step - loss: 0.3771
- acc: 0.8272 - val_loss: 0.4621 - val_acc: 0.7875
Epoch 21/64
200/200 [==============================] - 17s 84ms/step - loss: 0.3914
- acc: 0.8209 - val_loss: 0.4282 - val_acc: 0.8037
Epoch 22/64
200/200 [==============================] - 17s 85ms/step - loss: 0.3859
- acc: 0.8256 - val_loss: 0.4242 - val_acc: 0.7987
Epoch 23/64
200/200 [==============================] - 16s 80ms/step - loss: 0.3741
- acc: 0.8313 - val_loss: 0.4167 - val_acc: 0.8188
Epoch 24/64
200/200 [==============================] - 12s 58ms/step - loss: 0.3863
- acc: 0.8253 - val_loss: 0.4189 - val_acc: 0.8037
Epoch 25/64
200/200 [==============================] - 11s 56ms/step - loss: 0.3855
- acc: 0.8303 - val_loss: 0.4264 - val_acc: 0.7987
Epoch 26/64
200/200 [==============================] - 12s 60ms/step - loss: 0.3756
- acc: 0.8328 - val_loss: 0.4286 - val_acc: 0.7950
Epoch 27/64
200/200 [==============================] - 14s 68ms/step - loss: 0.3669
- acc: 0.8381 - val_loss: 0.4370 - val_acc: 0.8000
Epoch 28/64
200/200 [==============================] - 17s 83ms/step - loss: 0.3554
- acc: 0.8459 - val_loss: 0.4174 - val_acc: 0.8062
Epoch 29/64
200/200 [==============================] - 12s 62ms/step - loss: 0.3708
- acc: 0.8428 - val_loss: 0.4152 - val_acc: 0.8062
Epoch 30/64
200/200 [==============================] - 13s 63ms/step - loss: 0.3595
- acc: 0.8450 - val_loss: 0.4564 - val_acc: 0.8012
Epoch 31/64
200/200 [==============================] - 11s 53ms/step - loss: 0.3725
- acc: 0.8381 - val_loss: 0.4203 - val_acc: 0.7987
Epoch 32/64
200/200 [==============================] - 10s 51ms/step - loss: 0.3614
- acc: 0.8388 - val_loss: 0.4548 - val_acc: 0.8000
Epoch 33/64
200/200 [==============================] - 11s 56ms/step - loss: 0.3658
- acc: 0.8372 - val_loss: 0.4388 - val_acc: 0.7987
Epoch 34/64
200/200 [==============================] - 12s 60ms/step - loss: 0.3614
- acc: 0.8413 - val_loss: 0.4311 - val_acc: 0.7950
Epoch 35/64
200/200 [==============================] - 12s 62ms/step - loss: 0.3532
- acc: 0.8469 - val_loss: 0.4794 - val_acc: 0.7850
Epoch 36/64
200/200 [==============================] - 11s 56ms/step - loss: 0.3514
- acc: 0.8566 - val_loss: 0.4298 - val_acc: 0.8075
Epoch 37/64
200/200 [==============================] - 12s 58ms/step - loss: 0.3566
- acc: 0.8366 - val_loss: 0.4517 - val_acc: 0.8000
Epoch 38/64
200/200 [==============================] - 11s 57ms/step - loss: 0.3558
- acc: 0.8381 - val_loss: 0.5275 - val_acc: 0.7675
```

```
Epoch 39/64
200/200 [==============================] - 11s 57ms/step - loss: 0.3434
- acc: 0.8434 - val_loss: 0.4048 - val_acc: 0.8125
Epoch 40/64
200/200 [==============================] - 11s 57ms/step - loss: 0.3520
- acc: 0.8472 - val_loss: 0.4495 - val_acc: 0.7875
Epoch 41/64
200/200 [==============================] - 11s 57ms/step - loss: 0.3551
- acc: 0.8447 - val_loss: 0.4447 - val_acc: 0.8037
Epoch 42/64
200/200 [==============================] - 10s 52ms/step - loss: 0.3572
- acc: 0.8388 - val_loss: 0.4214 - val_acc: 0.8087
Epoch 43/64
200/200 [==============================] - 10s 51ms/step - loss: 0.3513
- acc: 0.8409 - val_loss: 0.4392 - val_acc: 0.7975
Epoch 44/64
200/200 [==============================] - 10s 52ms/step - loss: 0.3476
- acc: 0.8462 - val_loss: 0.4527 - val_acc: 0.8000
Epoch 45/64
200/200 [==============================] - 10s 50ms/step - loss: 0.3523
- acc: 0.8431 - val_loss: 0.4370 - val_acc: 0.7962
Epoch 46/64
200/200 [==============================] - 10s 51ms/step - loss: 0.3565
- acc: 0.8428 - val_loss: 0.4145 - val_acc: 0.7987
Epoch 47/64
200/200 [==============================] - 11s 54ms/step - loss: 0.3498
- acc: 0.8494 - val_loss: 0.4360 - val_acc: 0.8012
Epoch 48/64
200/200 [==============================] - 11s 55ms/step - loss: 0.3554
- acc: 0.8459 - val_loss: 0.4538 - val_acc: 0.8062
Epoch 49/64
200/200 [==============================] - 11s 55ms/step - loss: 0.3503
- acc: 0.8469 - val_loss: 0.5011 - val_acc: 0.8012
Epoch 50/64
200/200 [==============================] - 11s 53ms/step - loss: 0.3501
- acc: 0.8453 - val_loss: 0.3996 - val_acc: 0.8112
Epoch 51/64
200/200 [==============================] - 11s 53ms/step - loss: 0.3427
- acc: 0.8469 - val_loss: 0.4394 - val_acc: 0.8050
Epoch 52/64
200/200 [==============================] - 11s 53ms/step - loss: 0.3450
- acc: 0.8528 - val_loss: 0.4295 - val_acc: 0.8087
Epoch 53/64
200/200 [==============================] - 11s 57ms/step - loss: 0.3492
- acc: 0.8491 - val_loss: 0.4288 - val_acc: 0.8025
Epoch 54/64
200/200 [==============================] - 12s 62ms/step - loss: 0.3353
- acc: 0.8575 - val_loss: 0.4168 - val_acc: 0.8263
Epoch 55/64
200/200 [==============================] - 11s 56ms/step - loss: 0.3439
- acc: 0.8528 - val_loss: 0.4282 - val_acc: 0.8112
Epoch 56/64
200/200 [==============================] - 11s 55ms/step - loss: 0.3353
- acc: 0.8566 - val_loss: 0.4288 - val_acc: 0.8050
Epoch 57/64
200/200 [==============================] - 11s 55ms/step - loss: 0.3494
- acc: 0.8494 - val_loss: 0.4168 - val_acc: 0.8112
```

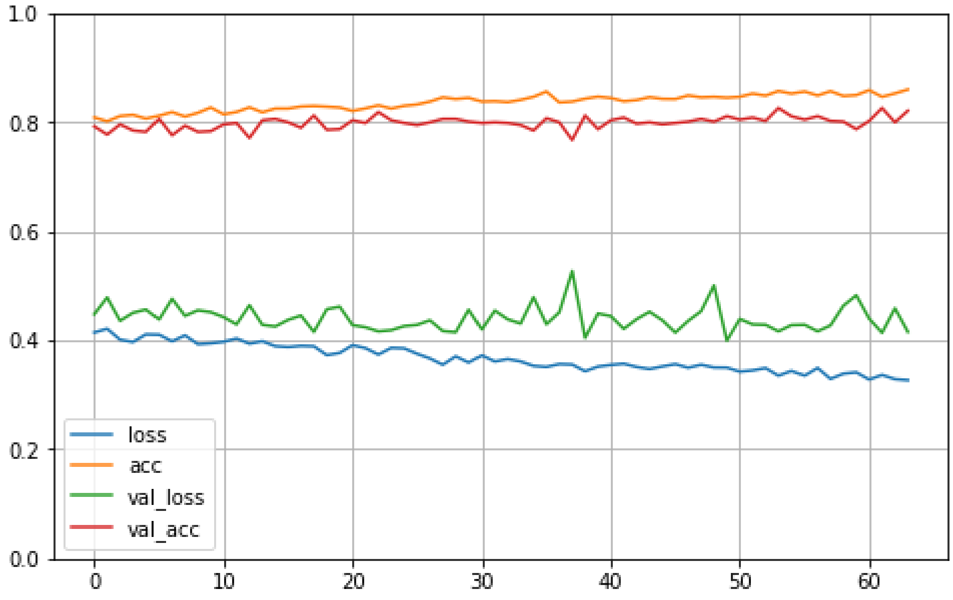
```
Epoch 58/64
200/200 [==============================] - 11s 56ms/step - loss: 0.3293
- acc: 0.8572 - val_loss: 0.4277 - val_acc: 0.8025
Epoch 59/64
200/200 [==============================] - 13s 66ms/step - loss: 0.3390
- acc: 0.8484 - val_loss: 0.4638 - val_acc: 0.8012
Epoch 60/64
200/200 [==============================] - 12s 62ms/step - loss: 0.3415
- acc: 0.8500 - val_loss: 0.4830 - val_acc: 0.7875
Epoch 61/64
200/200 [==============================] - 12s 61ms/step - loss: 0.3282
- acc: 0.8591 - val_loss: 0.4401 - val_acc: 0.8025
Epoch 62/64
200/200 [==============================] - 13s 66ms/step - loss: 0.3366
- acc: 0.8469 - val_loss: 0.4141 - val_acc: 0.8263
Epoch 63/64
200/200 [==============================] - 13s 63ms/step - loss: 0.3289
- acc: 0.8531 - val_loss: 0.4592 - val_acc: 0.8000
Epoch 64/64
200/200 [==============================] - 12s 62ms/step - loss: 0.3270
- acc: 0.8603 - val_loss: 0.4157 - val_acc: 0.8213
```

In [137]:

```
#model 2 curves
```

In [138]:

```
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1) # set the vertical range to [0-1]
plt.show()
```



In [139]:

```
#model 2 predictions
```

In [141]:

```
predictions2 = model2.predict(test_generator)
```

In [143]:

```
ids = range(1, len(X_test) + 1)
solution = pd.DataFrame({"id": ids, "label":list(predictions2)})
cols = ['label']
for col in cols:
    solution[col] = solution[col].map(lambda x: str(x).lstrip('[').rstrip(']')).ast
ype(float)
solution.to_csv("kaggle_submission2.csv", index = False)
```

In [ ]:

```
#kaggle score 0.69347
```

In [ ]:

```
#model 3, 20 epochs , 8 layers
```

In [ ]:

```
model3 = keras.models.Sequential()
model3.add(keras.layers.Conv2D(32, (3, 3), activation='relu',input_shape=(size, siz
e, channels)))
model3.add(keras.layers.MaxPooling2D((2, 2)))
model3.add(keras.layers.Conv2D(5, (3, 3), activation='relu'))
model3.add(keras.layers.MaxPooling2D((2, 2)))
model3.add(keras.layers.Conv2D(128, (3, 3), activation='relu'))
model3.add(keras.layers.MaxPooling2D((2, 2)))
model3.add(keras.layers.Flatten())
model3.add(keras.layers.Dropout(0.5))  #Dropout for regularization
model3.add(keras.layers.Dense(256, activation='relu'))
model3.add(keras.layers.Dense(1, activation='sigmoid'))
```

In [159]:

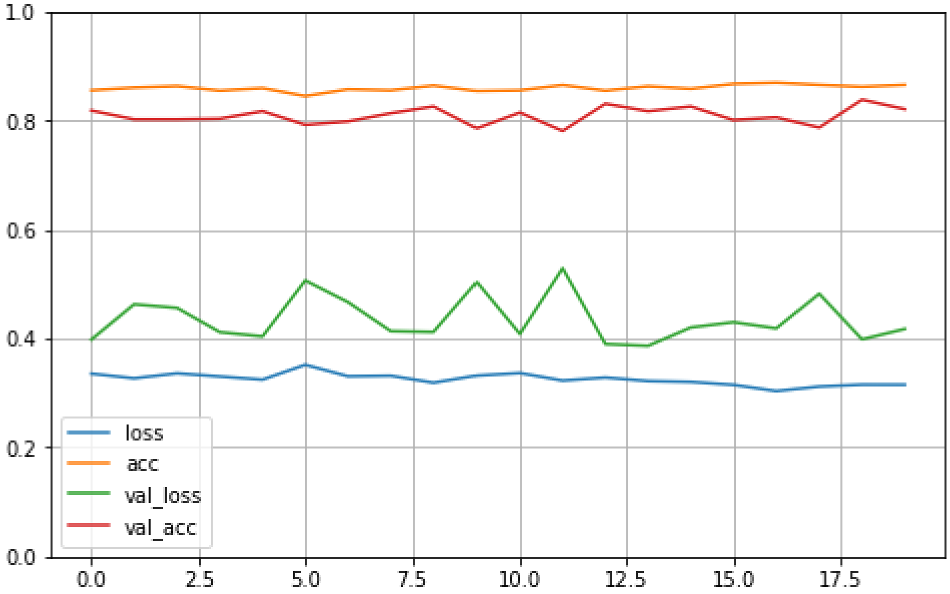
```
batch = 16
ntrain = len(X_train)
nval = len(X_val)

model3.compile(loss='binary_crossentropy', optimizer=keras.optimizers.RMSprop(lr=1e
-4), metrics=['acc'])

val_datagen = keras.preprocessing.image.ImageDataGenerator(rescale=1./255)

train_datagen = keras.preprocessing.image.ImageDataGenerator(rescale=1./255,
                                                             rotation_range=20,
                                                             zoom_range=0.1,
                                                             width_shift_range=0.1,
                                                             height_shift_range=0.1
,

                                                             shear_range=0.1,
                                                             horizontal_flip=True,
                                                             fill_mode="nearest")

train_generator = train_datagen.flow(X_train, y_train, batch_size=batch)

val_generator = val_datagen.flow(X_val, y_val, batch_size=batch)


m3_start = datetime.now()

history = model1.fit_generator(train_generator,
                               steps_per_epoch=ntrain // batch,
                               epochs=20,
                               validation_data=val_generator,
                               validation_steps=nval // batch)

m3_end = datetime.now()
```

```
Epoch 1/20
200/200 [==============================] - 11s 55ms/step - loss: 0.3356
- acc: 0.8559 - val_loss: 0.3984 - val_acc: 0.8188
Epoch 2/20
200/200 [==============================] - 13s 64ms/step - loss: 0.3271
- acc: 0.8609 - val_loss: 0.4632 - val_acc: 0.8025
Epoch 3/20
200/200 [==============================] - 12s 62ms/step - loss: 0.3362
- acc: 0.8637 - val_loss: 0.4565 - val_acc: 0.8025
Epoch 4/20
200/200 [==============================] - 12s 58ms/step - loss: 0.3304
- acc: 0.8553 - val_loss: 0.4120 - val_acc: 0.8037
Epoch 5/20
200/200 [==============================] - 11s 57ms/step - loss: 0.3246
- acc: 0.8600 - val_loss: 0.4043 - val_acc: 0.8175
Epoch 6/20
200/200 [==============================] - 10s 50ms/step - loss: 0.3519
- acc: 0.8453 - val_loss: 0.5070 - val_acc: 0.7925
Epoch 7/20
200/200 [==============================] - 10s 50ms/step - loss: 0.3308
- acc: 0.8578 - val_loss: 0.4672 - val_acc: 0.7987
Epoch 8/20
200/200 [==============================] - 10s 50ms/step - loss: 0.3318
- acc: 0.8559 - val_loss: 0.4138 - val_acc: 0.8138
Epoch 9/20
200/200 [==============================] - 10s 49ms/step - loss: 0.3189
- acc: 0.8647 - val_loss: 0.4123 - val_acc: 0.8263
Epoch 10/20
200/200 [==============================] - 10s 50ms/step - loss: 0.3320
- acc: 0.8547 - val_loss: 0.5037 - val_acc: 0.7862
Epoch 11/20
200/200 [==============================] - 10s 50ms/step - loss: 0.3367
- acc: 0.8559 - val_loss: 0.4087 - val_acc: 0.8150
Epoch 12/20
200/200 [==============================] - 10s 51ms/step - loss: 0.3229
- acc: 0.8656 - val_loss: 0.5295 - val_acc: 0.7812
Epoch 13/20
200/200 [==============================] - 10s 51ms/step - loss: 0.3285
- acc: 0.8553 - val_loss: 0.3901 - val_acc: 0.8313
Epoch 14/20
200/200 [==============================] - 10s 50ms/step - loss: 0.3220
- acc: 0.8634 - val_loss: 0.3868 - val_acc: 0.8175
Epoch 15/20
200/200 [==============================] - 10s 51ms/step - loss: 0.3204
- acc: 0.8587 - val_loss: 0.4206 - val_acc: 0.8263
Epoch 16/20
200/200 [==============================] - 10s 51ms/step - loss: 0.3150
- acc: 0.8678 - val_loss: 0.4303 - val_acc: 0.8012
Epoch 17/20
200/200 [==============================] - 10s 51ms/step - loss: 0.3036
- acc: 0.8700 - val_loss: 0.4189 - val_acc: 0.8062
Epoch 18/20
200/200 [==============================] - 10s 51ms/step - loss: 0.3119
- acc: 0.8659 - val_loss: 0.4827 - val_acc: 0.7875
Epoch 19/20
200/200 [==============================] - 10s 51ms/step - loss: 0.3154
- acc: 0.8625 - val_loss: 0.3988 - val_acc: 0.8388
```

```
Epoch 20/20
200/200 [==============================] - 10s 51ms/step - loss: 0.3153
- acc: 0.8659 - val_loss: 0.4179 - val_acc: 0.8213
```

In [164]:

```
# Plot learning curves
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1) # set the vertical range to [0-1]
plt.show()
```



In [177]:

```
predictions = model3.predict(X_test, verbose=0)
ids = range(1, len(X_test) + 1)
solution = pd.DataFrame({"id": ids, "label":list(predictions)})
cols = ['label']
for col in cols:
    solution[col] = solution[col].map(lambda x: str(x).lstrip('[').rstrip(']')).ast
ype(float)
solution.to_csv("ubmission5.csv", index = False)
```

In [ ]:

```
#kaggle score 0.08874
```

In [ ]:

```
#Model 4: 8 layers with dropout regularization and batch normalization
```

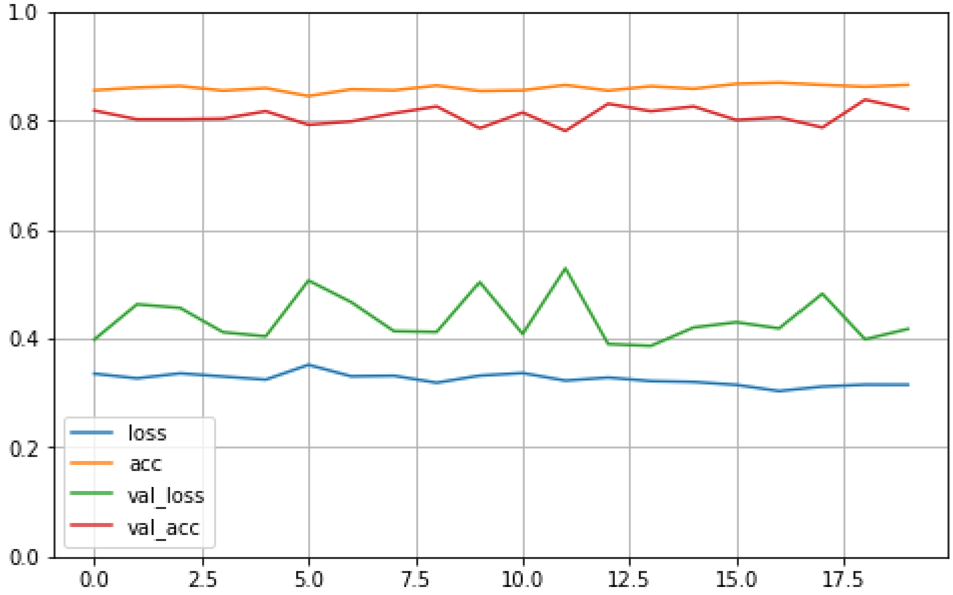
In [180]:

```
model_4 = Sequential()

model_4.add(Conv2D(32, (3, 3), input_shape=(img_width, img_height, 3)))
model_4.add(Activation('relu'))
model_4.add(MaxPooling2D(pool_size=(2, 2)))

model_4.add(Conv2D(32, (3, 3)))
model_4.add(Activation('relu'))
model_4.add(MaxPooling2D(pool_size=(2, 2)))

model_4.add(Conv2D(64, (3, 3)))
model_4.add(Activation('relu'))
model_4.add(MaxPooling2D(pool_size=(2, 2)))

model_4.add(Conv2D(64, (3, 3)))
model_4.add(Activation('relu'))
model_4.add(MaxPooling2D(pool_size=(2, 2)))

model_4.add(Flatten())
model_4.add(Dense(64))
model_4.add(BatchNormalization())
model_4.add(Activation('relu'))
model_4.add(Dropout(0.5))
model_4.add(Dense(1))
model_4.add(Activation('sigmoid'))

model_4.compile(loss='binary_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])

model_4.summary()
```

```
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_12 (Conv2D)           (None, 222, 222, 32)      896       
_________________________________________________________________
activation_5 (Activation)    (None, 222, 222, 32)      0         
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 111, 111, 32)      0         
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 109, 109, 32)      9248      
_________________________________________________________________
activation_6 (Activation)    (None, 109, 109, 32)      0         
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 54, 54, 32)        0         
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 52, 52, 64)        18496     
_________________________________________________________________
activation_7 (Activation)    (None, 52, 52, 64)        0         
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 26, 26, 64)        0         
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 24, 24, 64)        36928     
_________________________________________________________________
activation_8 (Activation)    (None, 24, 24, 64)        0         
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 12, 12, 64)        0         
_________________________________________________________________
flatten_7 (Flatten)          (None, 9216)              0         
_________________________________________________________________
dense_17 (Dense)             (None, 64)                589888    
_________________________________________________________________
batch_normalization_1 (Batch (None, 64)                256       
_________________________________________________________________
activation_9 (Activation)    (None, 64)                0         
_________________________________________________________________
dropout_13 (Dropout)         (None, 64)                0         
_________________________________________________________________
dense_18 (Dense)             (None, 1)                 65        
_________________________________________________________________
activation_10 (Activation)   (None, 1)                 0         
=================================================================
Total params: 655,777
Trainable params: 655,649
Non-trainable params: 128
_________________________________________________________________
```

In [193]:

```
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1) # set the vertical range to [0-1]
plt.show()
```



In [194]:

```
predictions4 = model_4.predict(X_test, verbose=0)
```

In [195]:

```
ids = range(1, len(X_test) + 1)
solution = pd.DataFrame({"id": ids, "label":list(predictions4)})
cols = ['label']
for col in cols:
    solution[col] = solution[col].map(lambda x: str(x).lstrip('[').rstrip(']')).ast
ype(float)
solution.to_csv("kaggle_submission_4.csv", index = False)
```

In [ ]:

```
#kaggle score username Ankita Avadhani 0.08465
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]: