Allison Roeser

Kaggle username: aroeser

Dogs vs. Cats: Image Processing with CNN

Data Preparation, Exploration, Visualization:

The Dog vs. Cats Kaggle training dataset contains 2,000 photographs of pets. Each image displays either a dog or a cat, and a label of dog or cat is provided. Figure 1 shows the first twenty-five images in the dataset. There are 1,014 dog images and 986 cat images in the training dataset. Because each digit is roughly equally represented in the training set, frequency of occurrence should not have a strong impact on predicted value. The test set contains 1,000 unlabeled images.

One challenge with the dataset is the inconsistencies in the images. In some images only the animal's face is present. In other images, the entire body is included. In other images, a person is holding the animal. These differences will make animal distinction more difficult because the computer must determine which variations between images are the because of differences between species and which variations result from the placement of the animal. The pet is generally centered in each picture. This consistency is helpful for image recognition.

The size of the images also varies considerably. Figure 2 shows the aspect ratio (width / height) for each image. The mean aspect ratio 1.15. Figure 3 graphs width versus height for each image to again show the disparity in image sizes. Inconsistent sizing and scaling will make image recognition more difficult. (Kostadinov, 2016).

Implementation and Programming:

Full code is attached in the Appendix to show the specific implementation. The TensorFlow Keras package was leveraged to create four different Neural Network structures for image identification (dog or cat). The Neural Networks differed in the number of filters per layer (128 or 256) and the activation function utilized (relu or tahn) . Time to fit each model was also measured in order to weight the tradeoff between improved accuracy from more complex models with the longer processing times generally required. Training, validation, and test sets were utilized to verify that the models generalized well.

Confusion matrixes to were used to evaluate the classification accuracy of potential models. Learning curve plots were used to illustrate the improvement in training and validation set accuracy as the number of epochs used in modeling increased. Pandas and Seaborn were leveraged for exploratory data analysis and visualization.

Review Research Design and Modeling Methods:

Four Neural Networks were created using the TensorFlow Keras API. The design of the experiment was to determine how differing the number of filters and the activation function impacted processing time and training / validation / testing accuracy. The model structures were: 128 filters per layer with relu activation, 256 filters per layer with tahn activation, 128 filters per layer with tahn activation, and 256 filters per layer with tahn activation.

Each model was run for 5 epochs. Learning curve plots were created to illustrate how the training and validation accuracy improved with each epoch. Using additional epochs would have provided increased accuracy. However, processing time per epoch was around 3 minutes in duration. For efficiency purposes, the models were constrained to 5 epochs. Confusion matrixes were created for the training and validation sets to understand how well each model classified the images as cat or dog.

Review Results, Evaluate Models:

The chart below displays the results of the four tests. All four neural networks were highly successful in correctly identifying if the image was a dog or a cat. Validation and test set accuracy

Model Number	Filter Count	Activation	Processing Time (Min)	Training Set Accuracy	Validation Set Accuracy	Test Set Accuracy
1	128	relu	15.57	0.7636	0.773	0.757
2	256	tahn	49.71	0.7704	0.765	0.765
3	128	tahn	16.21	0.7765	0.757	0.747
4	256	relu	63.86	0.7881	0.766	0.766

scores were very close to the training scores. These models generalized well and were not overfit to the training data. However, the computation time for models with 256 filters per layer was substantially higher than the models with only 128 filters per layer. The most accurate model was Model 4 with 256 filters per layer and a relu activation function. Figures 5 displays detailed information for Model 4, including the number of layers and the various hyperparameters used. Figure 6 includes the learning curve plots for Model 4 by epoch. The accuracy scores improved significantly thru the second epoch and then began to plateau. The confusion matrices for training and validation data for Model 4 are shown in Figures 7 & 8.

<u>Exposition, Problem Description and Management Recommendations</u>:

Increasing the number of filters produced more accurate results but had a substantial impact on processing time. Because accuracy is the most important consideration in modeling for this application, 256 filters per layers should be used despite the increased computation time. Relu activation performed only slightly better on the test set than tahn activation. Either choice of activation function will provide a solid model. In the modeling process, 5 epochs were used because of modeling time constraints. However, because computation time is not a concern for management during model deployment, more epochs should be used to increased model accuracy.

As much as possible, the consistency of the input images should be improved for model accuracy improvements. The images in this set included full body images as well as face shots. An optimal modeling practice would be to either require full images of the animal or only face images. Also, do not have any other individuals or animals in the shots.

References

Kostadinov. (2016). Create dataset with Tensorflow. Dogs vs. Cats Redux: Kernels Edition. Retrieved from: https://www.kaggle.com/freeman89/create-dataset-with-tensorflow

Appendix

Figure 1: 5-by-5 matrix of first 25 images in the training dataset labeled as Dog or Cat

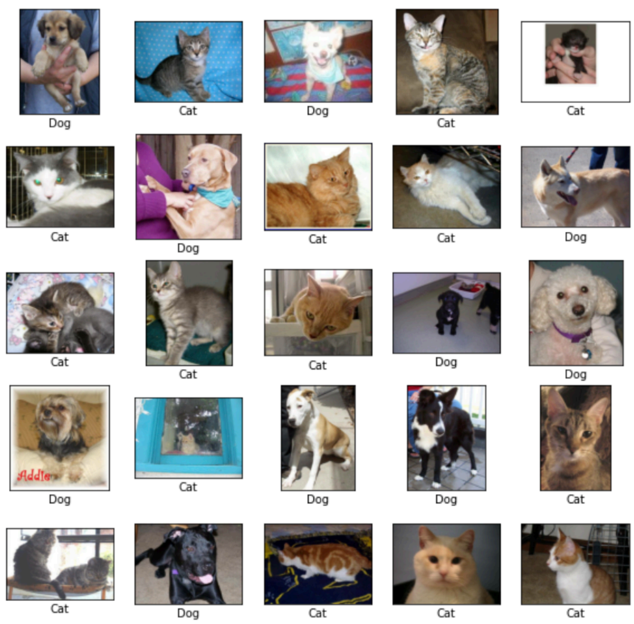


Figure 2: Mean aspect ratio for images the 3000 images in the training & test sets (Kostadinov, 2016)

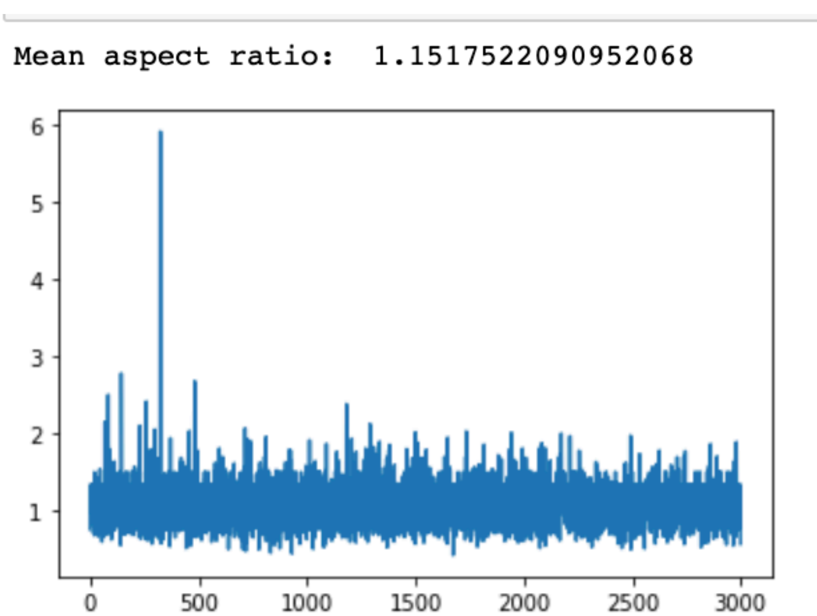


Figure 3: Pixel width vs. height for the 3000 images in the training and test sets (Kostadinov, 2016)

```
Mean width: 403.5473333333333
Mean height: 362.759
```

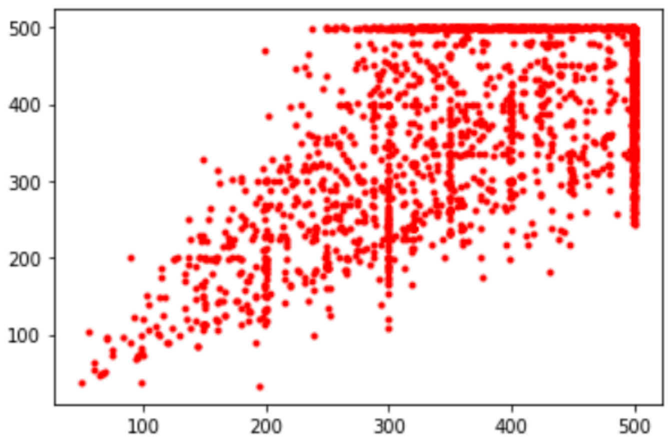


Figure 4: Summary table for the four models

Model Number	Filter Count	Activation	Processing Time (Min)	Training Set Accuracy	Validation Set Accuracy	Test Set Accuracy
1	128	relu	15.57	0.7636	0.773	0.757
2	256	tahn	49.71	0.7704	0.765	0.765
3	128	tahn	16.21	0.7765	0.757	0.747
4	256	relu	63.86	0.7881	0.766	0.766

Figure 5: Summary information for Model 4 (champion model / most accurate model)

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 64, 64, 64)        3200
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 256)       147712
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 256)       590080
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 256)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 256)       590080
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 16, 16, 256)       590080
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 8, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 16384)             0
_________________________________________________________________
dense (Dense)                (None, 128)               2097280
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 65
=================================================================
Total params: 4,026,753
Trainable params: 4,026,753
Non-trainable params: 0
```

Figure 6: Learning curve plot for Model 4 (champion model / most accurate model)

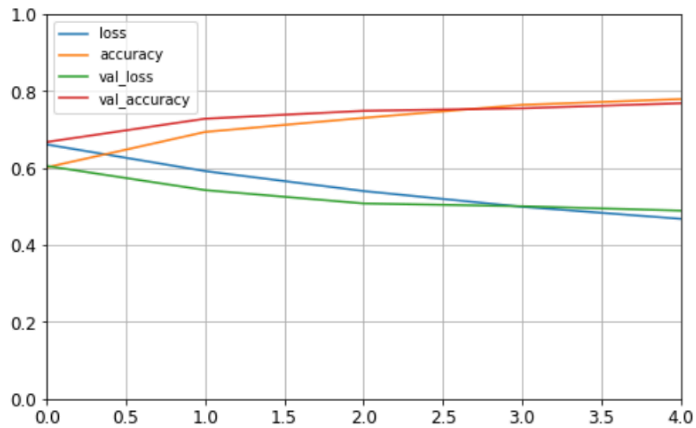


Figure 7: Confusion matrix for Model 4 training set

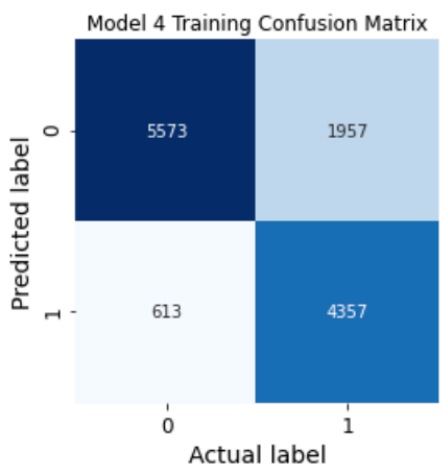
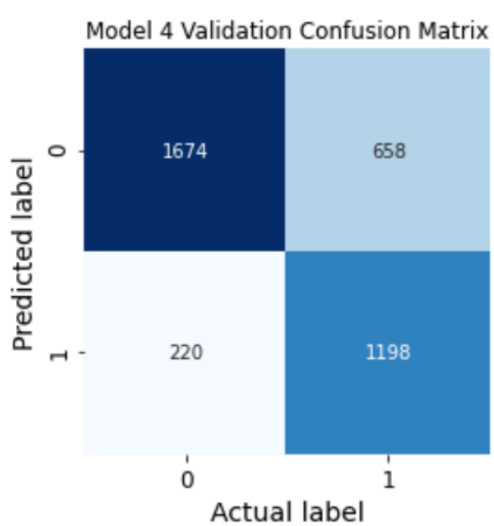


Figure 8: Confusion matrix for Model 4 validation set



In [34]:
```
# Northwestern University
# Predict 422
# W7 Part2 Classify Dog and Cat images using CNN
# Author Christopher Fiore
# -------------------------------------------------------------
# S1 Run SetUp Script to Install Packages
import pandas as pd  # data frame operations
import sklearn
import plotly
import plotly.graph_objs as go
import time
import numpy as np
import os
import sys
import re # regular expressions
import scipy
import seaborn as sns  # pretty plotting, including heat map
from functools import partial
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score, confusion_matrix, mean_squared_e

# Python ≥3.5 is required
assert sys.version_info >= (3, 5)
# Scikit-Learn ≥0.20 is required
assert sklearn.__version__ >= "0.20"

try:
    # %tensorflow_version only exists in Colab.
    %tensorflow_version 2.x
    IS_COLAB = True
except Exception:
    IS_COLAB = False

# TensorFlow ≥2.0 is required
import tensorflow as tf
from tensorflow import keras
#assert tf.__version__ >= "2.0"

# To plot pretty figures
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
mpl.rc('axes', labelsize=14)
mpl.rc('xtick', labelsize=12)
mpl.rc('ytick', labelsize=12)

#Set enviorment varaibles
random_seed=1
#height = 64
#width = 64

# to make this notebook's output stable across runs
#np.random.seed(random_seed)
#tf.random.set_seed(random_seed)
```

In [35]:
```
#S3 Establish working directory
os.getcwd()
%cd /content/gdrive/My Drive/NWU_ta/MSDS422_PML/wk7
!pwd
!ls
print('Working Directory')
print(os.getcwd())
work_dir = "/content/gdrive/My Drive/NWU_ta/MSDS422_PML/wk7/working/"
data_dir = work_dir+"kgdata/"
chp_id = "cnn"

#S3a Define Function to Create CNN - Soure Geron Chap14
def save_fig(fig_id, tight_layout=True, fig_extension="png", resolution=300
    path = os.path.join(work_dir, fig_id + "." + fig_extension)
    print("Saving figure", fig_id)
    if tight_layout:
        plt.tight_layout()
    plt.savefig(path, format=fig_extension, dpi=resolution)

def plot_image(image):
    plt.imshow(image, cmap="gray", interpolation="nearest")
    plt.axis("off")

def plot_color_image(image):
    plt.imshow(image, interpolation="nearest")
    plt.axis("off")

def feature_map_size(input_size, kernel_size, strides=1, padding="SAME"):
    if padding == "SAME":
        return (input_size - 1) // strides + 1
    else:
        return (input_size - kernel_size) // strides + 1

def dist_plot(var1, var2, var3):
    tmp_plt=sns.countplot(var1, palette="Blues").set_title(var2)
    tmp_fig = tmp_plt.get_figure()
    tmp_fig.savefig(var3 + ".png",
        bbox_inches = 'tight', dpi=None, facecolor='w', edgecolor='b',
        orientation='portrait', papertype=None, format=None,
        transparent=True, pad_inches=0.25)
    return(tmp_plt)

def pad_before_and_padded_size(input_size, kernel_size, strides=1):
    fmap_size = feature_map_size(input_size, kernel_size, strides)
    padded_size = max((fmap_size - 1) * strides + kernel_size, input_size)
    pad_before = (padded_size - input_size) // 2
    return pad_before, padded_size

def manual_same_padding(images, kernel_size, strides=1):
    if kernel_size == 1:
        return images.astype(np.float32)
    batch_size, height, width, channels = images.shape
    top_pad, padded_height = pad_before_and_padded_size(height, kernel_size
    left_pad, padded_width  = pad_before_and_padded_size(width, kernel_size
    padded_shape = [batch_size, padded_height, padded_width, channels]
    padded_images = np.zeros(padded_shape, dtype=np.float32)
```

```
    padded_images[:, top_pad:height+top_pad, left_pad:width+left_pad, :] =
    return padded_images
#Tensorboard Logs
root_logdir = os.path.join(os.curdir, "tf_logs")
def get_run_logdir():
    import time
    run_id = time.strftime("run_%Y_%m_%d-%H_%M_%S")
    return os.path.join(root_logdir, run_id)
```

```
[Errno 2] No such file or directory: '/content/gdrive/My Drive/NWU_ta/MSD
S422_PML/wk7'
/Users/allisonroeser/Desktop
/Users/allisonroeser/Desktop
$RECYCLE.BIN
1.jpg
11.712_ExerciseFiles
Assignment 7 code.pdf
Image prep.ipynb
MicroStrategy
Northwestern
PRD422_Assign7-CNN.ipynb
PRD422_Assign7_DNN.ipynb
PRD422_Assign7_ImgPrep.ipynb
Pages.app
SAS
Screen Shot 2020-05-21 at 10.31.50 PM.png
Screen Shot 2020-05-21 at 10.35.23 PM.png
Screen Shot 2020-05-21 at 10.35.40 PM.png
Screen Shot 2020-05-21 at 11.40.15 PM.png
Screen Shot 2020-05-21 at 11.50.50 PM.png
Screen Shot 2020-05-21 at 5.58.05 PM.png
Screen Shot 2020-05-21 at 7.14.12 AM.png
Screen Shot 2020-05-21 at 7.15.01 AM.png
Screen Shot 2020-05-21 at 7.16.34 AM.png
Screen Shot 2020-05-22 at 1.48.53 PM.png
Screen Shot 2020-05-22 at 10.19.20 AM.png
Screen Shot 2020-05-22 at 12.04.12 AM.png
Screen Shot 2020-05-22 at 12.08.39 AM.png
Screen Shot 2020-05-22 at 6.50.23 PM.png
Screen Shot 2020-05-22 at 6.56.01 PM.png
Screen Shot 2020-05-22 at 6.56.36 PM.png
Screen Shot 2020-05-22 at 6.57.25 PM.png
Screen Shot 2020-05-22 at 7.00.36 PM.png
Screen Shot 2020-05-22 at 7.35.04 PM.png
Screen Shot 2020-05-23 at 9.53.09 AM.png
Screen Shot 2020-05-23 at 9.54.09 AM.png
Screen Shot 2020-05-23 at 9.55.15 AM.png
TestCMHL2NPL300100.png
TestDistCatDog.png
TrainDistCatDog.png
Untitled.ipynb
ValidDistCatDog.png
Week 7 Cat & Dogs (1).ipynb
Week 7 Jumpstart (1).ipynb
Week 7 May 22 - Jupyter Notebook Final.pdf
Week 7 May 22 - Jupyter Notebook.pdf
Week 7 May 22.ipynb
Week 7 May 22.ipynb copy
```

```
cats
cats_dogs_64-128
cats_dogs_arrays
catsdogs_1.csv
catsdogs_1.xlsx
catsdogs_2.xlsx
catsdogs_3.xlsx
catsdogs_4.xlsx
desktop.ini
dogs
myHealthAdvisor
test
test.ipynb
testcats_dogs_arrays
tf_logs
tmp
train
traincats_dogs_arrays
~$Week 2.docx
~$Week 3.docx
~$Week 4.docx
~$Week 9.docx
~$ansformation Plan.docx
~$armacy study.docx
~$ek 1 COVID.docx
~$ek 7 discussion.docx
~$ek 7 paper.docx
~$havior Change Challenge.docx
~$itanic.docx
Working Directory
/Users/allisonroeser/Desktop
```

In [39]:
```
#S5 Load/Import data created from PRD422CD_Prep notebook
cats_1000_64_64_1 = np.load('/Users/allisonroeser/Desktop/train' +'cats_dog
dogs_1000_64_64_1 = np.load('/Users/allisonroeser/Desktop/train' +'cats_dog

# Examine first cat and first dog grayscale images
plot_image(cats_1000_64_64_1[0,:,:,0])
```

In [4]:
```
#S5 Load/Import data created from PRD422CD_Prep notebook
cats_1000_64_64_1_test = np.load('/Users/allisonroeser/Desktop/test' +'cats
dogs_1000_64_64_1_test = np.load('/Users/allisonroeser/Desktop/test' +'cats

# Examine first cat and first dog grayscale images
plot_image(cats_1000_64_64_1_test[0,:,:,0])
```

In [5]:
```
plot_image(dogs_1000_64_64_1[0,:,:,0])
```

In [6]:
```
#S6 Create modeling dataset - stack cat and dog array
X_cat_dog= np.concatenate((cats_1000_64_64_1, dogs_1000_64_64_1), axis = 0)
#Drop last column in array will add back after scaling process
X_cat_dog=X_cat_dog[:,:,:,-1]
X_cat_dog.shape

#Assign labels
y_cat_dog = np.concatenate((np.zeros((12500), dtype = np.int32),
                            np.ones((12500), dtype = np.int32)), axis = 0)
#S7 Split Train, Validate and Test
X_train, X_test_ds, y_train, y_test_ds= train_test_split(X_cat_dog, y_cat_d
                                                          test_size=0.5, ran
X_test, X_valid, y_test, y_valid = train_test_split(X_test_ds, y_test_ds,
                                                     test_size=0.30, random_
#S8 Scale images/numpy array
X_mean = X_train.mean(axis=0, keepdims=True)
X_std = X_train.std(axis=0, keepdims=True) + 1e-7
X_train = (X_train - X_mean) / X_std
X_valid = (X_valid - X_mean) / X_std
X_test = (X_test - X_mean) / X_std

X_train = X_train[..., np.newaxis]
X_valid = X_valid[..., np.newaxis]
X_test = X_test[..., np.newaxis]

#Review Distribution
print(X_train.shape)
print(X_test.shape)
print(X_valid.shape)
print(y_train.shape)
print(y_test.shape)
print(y_valid.shape)
```

```
(12500, 64, 64, 1)
(8750, 64, 64, 1)
(3750, 64, 64, 1)
(12500,)
(8750,)
(3750,)
```

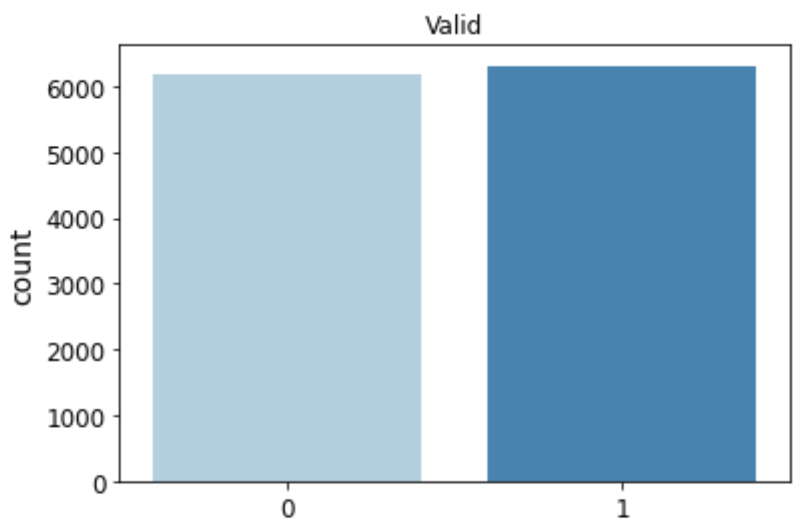
In [7]:
```
#S9 Check distribtion of test , valid and train
cd_plt_trn=dist_plot(y_train, 'Train', "TrainDistCatDog")
cd_plt_trn.get_figure().show()

cd_plt_tst=dist_plot(y_test, 'Test', "TestDistCatDog")
cd_plt_tst.get_figure().show()

cd_plt_vld=dist_plot(y_valid, 'Valid', "ValidDistCatDog")
cd_plt_vld.get_figure().show()
```

```
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:3: UserWarning: Matplotlib is currently using module://ipyker
nel.pylab.backend_inline, which is a non-GUI backend, so cannot show the
figure.
  This is separate from the ipykernel package so we can avoid doing impor
ts until
/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:6: UserWarning: Matplotlib is currently using module://ipyker
nel.pylab.backend_inline, which is a non-GUI backend, so cannot show the
figure.

/Users/allisonroeser/opt/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:9: UserWarning: Matplotlib is currently using module://ipyker
nel.pylab.backend_inline, which is a non-GUI backend, so cannot show the
figure.
  if __name__ == '__main__':
```



In [8]:
```
#S10 Compile Model
model = keras.models.Sequential([
    keras.layers.Conv2D(filters=64, kernel_size=7, activation='relu', paddi
    keras.layers.MaxPooling2D(pool_size=2),
    keras.layers.Conv2D(filters=128, kernel_size=3, activation='relu', padd
    keras.layers.Conv2D(filters=128, kernel_size=3, activation='relu', padd
    keras.layers.MaxPooling2D(pool_size=2),
    keras.layers.Conv2D(filters=128, kernel_size=3, activation='relu', padd
    keras.layers.Conv2D(filters=128, kernel_size=3, activation='relu', padd
    keras.layers.MaxPooling2D(pool_size=2),
    keras.layers.Flatten(),
    keras.layers.Dense(units=128, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(units=64, activation='relu'),
    keras.layers.Dropout(0.5),
    #keras.layers.Dense(units=2, activation='softmax'),
    keras.layers.Dense(1, activation='sigmoid'),
])
```

In [9]:
```
#S11 Clear and Reset log
keras.backend.clear_session()
np.random.seed(1)
#tf.random.set_random_seed(1)
#Reset Log Directory
run_logdir = get_run_logdir()
```

In [11]:
```
#S12 Execution with early Stopping
start_time_1 = time.process_time()
tensorboard_cb = keras.callbacks.TensorBoard(run_logdir)
checkpoint_cb = keras.callbacks.ModelCheckpoint(work_dir+"tmp/my_keras_mode
early_stopping_cb=keras.callbacks.EarlyStopping(monitor='loss', mode ='min'
#optimizer = keras.optimizers.Nadam(lr=1e-4, beta_1=0.9, beta_2=0.999)
optimizer = keras.optimizers.RMSprop(lr=1e-4, rho=0.9)
n_epochs = 5

model.compile(loss='binary_crossentropy', optimizer =optimizer, metrics=["a
history = model.fit(X_train, y_train, epochs=n_epochs,
                    validation_data=(X_test, y_test),
                    callbacks=[checkpoint_cb, tensorboard_cb, early_stoppin
score = model.evaluate(X_valid, y_valid)
X_new = X_test[:10] # pretend we have new images
y_pred = model.predict(X_new)
end_time_1 = time.process_time()
model_1_time = start_time_1 - end_time_1
```

```
Epoch 1/5
391/391 [==============================] - 166s 424ms/step - loss: 0.6696
- accuracy: 0.5876 - val_loss: 0.6377 - val_accuracy: 0.6360
Epoch 2/5
391/391 [==============================] - 214s 547ms/step - loss: 0.6186
- accuracy: 0.6623 - val_loss: 0.5841 - val_accuracy: 0.6947
Epoch 3/5
391/391 [==============================] - 161s 411ms/step - loss: 0.5804
- accuracy: 0.7065 - val_loss: 0.5460 - val_accuracy: 0.7226
Epoch 4/5
391/391 [==============================] - 162s 414ms/step - loss: 0.5301
- accuracy: 0.7462 - val_loss: 0.5292 - val_accuracy: 0.7349
Epoch 5/5
391/391 [==============================] - 161s 412ms/step - loss: 0.4963
- accuracy: 0.7636 - val_loss: 0.4894 - val_accuracy: 0.7726
118/118 [==============================] - 11s 95ms/step - loss: 0.4968 -
accuracy: 0.7565
```

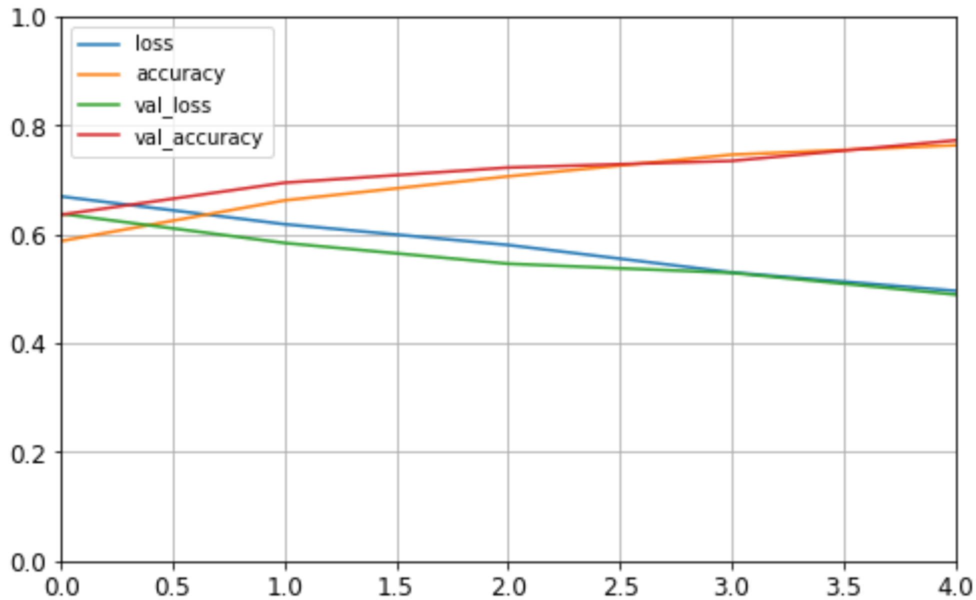
In [30]:
```
model_1_time = (end_time_1 - start_time_1)/600
model_1_time
```

Out[30]:
```
15.569164363333332
```

In [13]:
```
#Model Summary
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 64, 64, 64)        3200      
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 32, 32, 64)        0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 128)       73856     
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 128)       147584    
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 128)       0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 128)       147584    
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 16, 16, 128)       147584    
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 8, 128)         0         
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0         
_________________________________________________________________
dense (Dense)                (None, 128)               1048704   
_________________________________________________________________
dropout (Dropout)            (None, 128)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256      
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0         
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 65        
=================================================================
Total params: 1,576,833
Trainable params: 1,576,833
Non-trainable params: 0
_________________________________________________________________
```

In [14]:
```
#S13 View History
history.params
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
#save_fig("keras_learning_curves_plot")
plt.show()
```



In [15]:
```
#S14 Create Predicited Probabilties
y_proba = model.predict(X_valid)
y_proba.round(2)
```

Out[15]:
```
array([[0.75],
       [0.7 ],
       [0.33],
       ...,
       [0.4 ],
       [0.83],
       [0.63]], dtype=float32)
```

In [16]:
```
#Create Predicted Value
y_pred = model.predict_classes(X_valid)
```

```
WARNING:tensorflow:From <ipython-input-16-b5bbed21ec1c>:2: Sequential.pre
dict_classes (from tensorflow.python.keras.engine.sequential) is deprecat
ed and will be removed after 2021-01-01.
Instructions for updating:
Please use instead:* `np.argmax(model.predict(x), axis=-1)`,   if your mo
del does multi-class classification   (e.g. if it uses a `softmax` last-l
ayer activation).* `(model.predict(x) > 0.5).astype("int32")`,   if your
model does binary classification   (e.g. if it uses a `sigmoid` last-laye
r activation).
```

In [17]:
```
#View actual to predicted
print("Predicted classes:", np.reshape(y_pred[:20], (1, 20)))
print("Actual classes:   ", y_valid[:20])
```

```
Predicted classes: [[1 1 0 1 0 0 1 1 0 1 0 1 1 0 0 0 0 0 0 1]]
Actual classes:    [1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 0 0 0 1 0]
```

#Load the TensorBoard notebook extension %load_ext tensorboard %tensorboard --logdir tf_logs

In [18]:
```
#Score test dataset
scr=model.predict_classes(X_test)
#Conver array to Pandas dataframe with submission titles
pd_scr=pd.DataFrame(scr)
pd_scr.index.name = 'ImageId'
pd_scr.columns = ['label']
print(pd_scr)
#Export to Excel
pd_scr.to_excel("catsdogs_1.xlsx")
```

```
         label
ImageId
0            0
1            0
2            0
3            0
4            1
...        ...
8745         0
8746         0
8747         0
8748         1
8749         1

[8750 rows x 1 columns]
```

In [19]:
```
model_1_train_acc = max(history.history["accuracy"])
model_1_train_acc
```

Out[19]:
```
0.7635999917984009
```

In [20]:
```
model_1_val_acc = max(history.history["val_accuracy"])
model_1_val_acc
```

Out[20]: 0.7725714445114136

In [24]:
```
model_1_test_loss, model_1_test_acc = score
model_1_test_acc
```

Out[24]: 0.7565333247184753

In [ ]:

In [ ]:

In [ ]:

In [40]:
```
#S10 Compile Model
model_2 = keras.models.Sequential([
    keras.layers.Conv2D(filters=64, kernel_size=7, activation='relu', paddi
    keras.layers.MaxPooling2D(pool_size=2),
    keras.layers.Conv2D(filters=256, kernel_size=3, activation='tanh', padd
    keras.layers.Conv2D(filters=256, kernel_size=3, activation='tanh', padd
    keras.layers.MaxPooling2D(pool_size=2),
    keras.layers.Conv2D(filters=256, kernel_size=3, activation='tanh', padd
    keras.layers.Conv2D(filters=256, kernel_size=3, activation='tanh', padd
    keras.layers.MaxPooling2D(pool_size=2),
    keras.layers.Flatten(),
    keras.layers.Dense(units=128, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(units=64, activation='relu'),
    keras.layers.Dropout(0.5),
    #keras.layers.Dense(units=2, activation='softmax'),
    keras.layers.Dense(1, activation='sigmoid'),
])
```

In [41]:
```
#S11 Clear and Reset log
keras.backend.clear_session()
np.random.seed(1)
#tf.random.set_random_seed(1)
#Reset Log Directory
run_logdir = get_run_logdir()
```

In [42]:
```
#S12 Execution with early Stopping
start_time_2 = time.process_time()
tensorboard_cb = keras.callbacks.TensorBoard(run_logdir)
checkpoint_cb = keras.callbacks.ModelCheckpoint(work_dir+"tmp/my_keras_mode
early_stopping_cb=keras.callbacks.EarlyStopping(monitor='loss', mode ='min'
#optimizer = keras.optimizers.Nadam(lr=1e-4, beta_1=0.9, beta_2=0.999)
optimizer = keras.optimizers.RMSprop(lr=1e-4, rho=0.9)
n_epochs = 5

model_2.compile(loss='binary_crossentropy', optimizer =optimizer, metrics=[
history_2 = model_2.fit(X_train, y_train, epochs=n_epochs,
                    validation_data=(X_test, y_test),
                    callbacks=[checkpoint_cb, tensorboard_cb, early_stoppin
score_2 = model_2.evaluate(X_valid, y_valid)
X_new = X_test[:10] # pretend we have new images
y_pred = model.predict(X_new)
end_time_2 = time.process_time()
model_2_time = (end_time_2 - start_time_2)/600
```

```
Epoch 1/5
391/391 [==============================] - 516s 1s/step - loss: 0.6568 -
accuracy: 0.6150 - val_loss: 0.5751 - val_accuracy: 0.7057
Epoch 2/5
391/391 [==============================] - 440s 1s/step - loss: 0.5748 -
accuracy: 0.7041 - val_loss: 0.5944 - val_accuracy: 0.6746
Epoch 3/5
391/391 [==============================] - 478s 1s/step - loss: 0.5256 -
accuracy: 0.7457 - val_loss: 0.5058 - val_accuracy: 0.7503
Epoch 4/5
391/391 [==============================] - 492s 1s/step - loss: 0.4812 -
accuracy: 0.7747 - val_loss: 0.5066 - val_accuracy: 0.7520
Epoch 5/5
391/391 [==============================] - 465s 1s/step - loss: 0.4414 -
accuracy: 0.7978 - val_loss: 0.4826 - val_accuracy: 0.7704
118/118 [==============================] - 33s 279ms/step - loss: 0.4864
- accuracy: 0.7653
```

In [43]:
```
model_2_time
```

Out[43]:
```
49.714304373333334
```

In [44]:
```
#S14 Create Predicited Probabilties
y_proba_2 = model_2.predict(X_valid)
y_proba_2.round(2)
```

Out[44]:
```
array([[0.67],
       [0.96],
       [0.07],
       ...,
       [0.35],
       [0.95],
       [0.48]], dtype=float32)
```

In [45]:
```
#Score test dataset
scr=model_2.predict_classes(X_test)
#Conver array to Pandas dataframe with submission titles
pd_scr=pd.DataFrame(scr)
pd_scr.index.name = 'ImageId'
pd_scr.columns = ['label']
print(pd_scr)
#Export to Excel
pd_scr.to_excel("catsdogs_2.xlsx")
```

```
         label
ImageId
0            0
1            1
2            0
3            0
4            1
...        ...
8745         0
8746         0
8747         0
8748         1
8749         1

[8750 rows x 1 columns]
```

In [46]:
```
model_2_train_acc = max(history_2.history["accuracy"])
model_2_train_acc
```

Out[46]: 0.7978399991989136

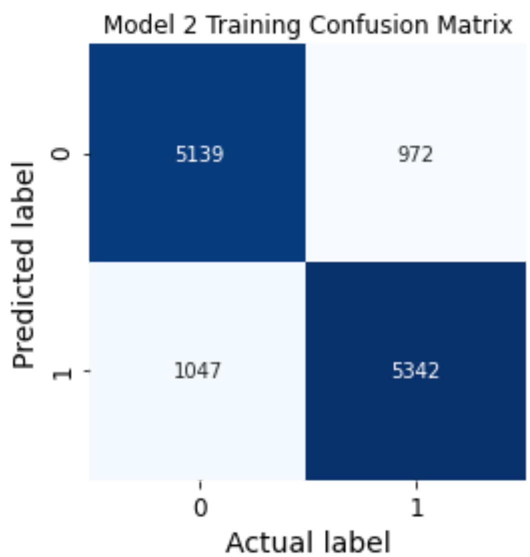
In [50]:
```
model_2_train_acc = max(history_2.history["val_accuracy"])
model_2_train_acc
```

Out[50]: 0.7703999876976013

In [48]:
```
model_2_test_loss, model_2_test_acc = score_2
model_2_test_acc
```

Out[48]: 0.765333354473114

In [51]:
```
#Plot Confusion Matrix DNN
cm_tst = confusion_matrix(y_train, model_2.predict_classes(X_train))
cm_tst_plt=sns.heatmap(cm_tst.T, square=True, annot=True, fmt='d', cbar=Fal
plt.xlabel('Actual label')
plt.ylabel('Predicted label')
plt.title("Model 2 Training Confusion Matrix");
fig3 = cm_tst_plt.get_figure()
fig3.savefig('TestCMHL2NPL300100.png',
        bbox_inches = 'tight', dpi=None, facecolor='w', edgecolor='b',
        orientation='portrait', papertype=None, format=None,
        transparent=True, pad_inches=0.25)
```




```
#S13 View History
history_2.params
pd.DataFrame(history_2.history_2).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
#save_fig("keras_learning_curves_plot")
plt.show()
```

In [ ]:

In [ ]:

In [53]:
```
#S10 Compile Model
model_3 = keras.models.Sequential([
    keras.layers.Conv2D(filters=64, kernel_size=7, activation='relu', paddi
    keras.layers.MaxPooling2D(pool_size=2),
    keras.layers.Conv2D(filters=128, kernel_size=3, activation='tanh', padd
    keras.layers.Conv2D(filters=128, kernel_size=3, activation='tanh', padd
    keras.layers.MaxPooling2D(pool_size=2),
    keras.layers.Conv2D(filters=128, kernel_size=3, activation='tanh', padd
    keras.layers.Conv2D(filters=128, kernel_size=3, activation='tanh', padd
    keras.layers.MaxPooling2D(pool_size=2),
    keras.layers.Flatten(),
    keras.layers.Dense(units=128, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(units=64, activation='relu'),
    keras.layers.Dropout(0.5),
    #keras.layers.Dense(units=2, activation='softmax'),
    keras.layers.Dense(1, activation='sigmoid'),
])
```

In [54]:
```
#S11 Clear and Reset log
keras.backend.clear_session()
np.random.seed(1)
#tf.random.set_random_seed(1)
#Reset Log Directory
run_logdir = get_run_logdir()
```

In [55]:
```
#S12 Execution with early Stopping
start_time_3 = time.process_time()
tensorboard_cb = keras.callbacks.TensorBoard(run_logdir)
checkpoint_cb = keras.callbacks.ModelCheckpoint(work_dir+"tmp/my_keras_mode
early_stopping_cb=keras.callbacks.EarlyStopping(monitor='loss', mode ='min'
#optimizer = keras.optimizers.Nadam(lr=1e-4, beta_1=0.9, beta_2=0.999)
optimizer = keras.optimizers.RMSprop(lr=1e-4, rho=0.9)
n_epochs = 5

model_3.compile(loss='binary_crossentropy', optimizer =optimizer, metrics=[
history_3 = model_3.fit(X_train, y_train, epochs=n_epochs,
                    validation_data=(X_test, y_test),
                    callbacks=[checkpoint_cb, tensorboard_cb, early_stoppin
score_3 = model_3.evaluate(X_valid, y_valid)
X_new = X_test[:10] # pretend we have new images
y_pred = model.predict(X_new)
end_time_3 = time.process_time()
model_3_time = (end_time_3 - start_time_3)/60
```

```
Epoch 1/5
391/391 [==============================] - 193s 494ms/step - loss: 0.6581
- accuracy: 0.6074 - val_loss: 0.6117 - val_accuracy: 0.6723
Epoch 2/5
391/391 [==============================] - 183s 468ms/step - loss: 0.5938
- accuracy: 0.6870 - val_loss: 0.5597 - val_accuracy: 0.7144
Epoch 3/5
391/391 [==============================] - 173s 442ms/step - loss: 0.5421
- accuracy: 0.7327 - val_loss: 0.5262 - val_accuracy: 0.7431
Epoch 4/5
391/391 [==============================] - 173s 442ms/step - loss: 0.5041
- accuracy: 0.7561 - val_loss: 0.5041 - val_accuracy: 0.7570
Epoch 5/5
391/391 [==============================] - 179s 457ms/step - loss: 0.4754
- accuracy: 0.7765 - val_loss: 0.5158 - val_accuracy: 0.7461
118/118 [==============================] - 12s 106ms/step - loss: 0.5118
- accuracy: 0.7475
```

In [62]:
```
model_3_time = model_3_time/10
model_3_time
```

Out[62]:
```
16.210448116666655
```

In [57]:
```
#S14 Create Predicited Probabilties
y_proba_3 = model_3.predict(X_valid)
y_proba_3.round(2)
```

Out[57]:
```
array([[0.5 ],
       [0.94],
       [0.07],
       ...,
       [0.31],
       [0.92],
       [0.14]], dtype=float32)
```

In [58]:
```
#Score test dataset
scr=model_3.predict_classes(X_test)
#Conver array to Pandas dataframe with submission titles
pd_scr=pd.DataFrame(scr)
pd_scr.index.name = 'ImageId'
pd_scr.columns = ['label']
print(pd_scr)
#Export to Excel
pd_scr.to_excel("catsdogs_3.xlsx")
```

```
         label
ImageId
0            0
1            0
2            0
3            0
4            1
...        ...
8745         0
8746         0
8747         0
8748         1
8749         1

[8750 rows x 1 columns]
```

In [77]:
```
model_3_train_acc = max(history_3.history["accuracy"])
model_3_train_acc
```

Out[77]: 0.7764800190925598

In [78]:
```
model_3_val_acc = max(history_3.history["val_accuracy"])
model_3_val_acc
```

Out[78]: 0.7570285797119141

In [61]:
```
model_3_test_loss, model_3_test_acc = score_3
model_3_test_acc
```

Out[61]: 0.7474666833877563

In [ ]:

In [ ]:

In [ ]:

In [63]:
```
#S10 Compile Model
model_4 = keras.models.Sequential([
    keras.layers.Conv2D(filters=64, kernel_size=7, activation='relu', paddi
    keras.layers.MaxPooling2D(pool_size=2),
    keras.layers.Conv2D(filters=256, kernel_size=3, activation='relu', padd
    keras.layers.Conv2D(filters=256, kernel_size=3, activation='relu', padd
    keras.layers.MaxPooling2D(pool_size=2),
    keras.layers.Conv2D(filters=256, kernel_size=3, activation='relu', padd
    keras.layers.Conv2D(filters=256, kernel_size=3, activation='relu', padd
    keras.layers.MaxPooling2D(pool_size=2),
    keras.layers.Flatten(),
    keras.layers.Dense(units=128, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(units=64, activation='relu'),
    keras.layers.Dropout(0.5),
    #keras.layers.Dense(units=2, activation='softmax'),
    keras.layers.Dense(1, activation='sigmoid'),
])
```

In [64]:
```
#S11 Clear and Reset log
keras.backend.clear_session()
np.random.seed(1)
#tf.random.set_random_seed(1)
#Reset Log Directory
run_logdir = get_run_logdir()
```

In [65]:
```
#S12 Execution with early Stopping
start_time_4 = time.process_time()
tensorboard_cb = keras.callbacks.TensorBoard(run_logdir)
checkpoint_cb = keras.callbacks.ModelCheckpoint(work_dir+"tmp/my_keras_mode
early_stopping_cb=keras.callbacks.EarlyStopping(monitor='loss', mode ='min'
#optimizer = keras.optimizers.Nadam(lr=1e-4, beta_1=0.9, beta_2=0.999)
optimizer = keras.optimizers.RMSprop(lr=1e-4, rho=0.9)
n_epochs = 5

model_4.compile(loss='binary_crossentropy', optimizer =optimizer, metrics=[
history_4 = model_4.fit(X_train, y_train, epochs=n_epochs,
                    validation_data=(X_test, y_test),
                    callbacks=[checkpoint_cb, tensorboard_cb, early_stoppin
score_4 = model_4.evaluate(X_valid, y_valid)
X_new = X_test[:10] # pretend we have new images
y_pred = model.predict(X_new)
end_time_4 = time.process_time()
model_4_time = (end_time_4 - start_time_4)/600
```

```
Epoch 1/5
391/391 [==============================] - 528s 1s/step - loss: 0.6728 -
accuracy: 0.5806 - val_loss: 0.6701 - val_accuracy: 0.5936
Epoch 2/5
391/391 [==============================] - 605s 2s/step - loss: 0.6166 -
accuracy: 0.6693 - val_loss: 0.6327 - val_accuracy: 0.6403
Epoch 3/5
391/391 [==============================] - 771s 2s/step - loss: 0.5592 -
accuracy: 0.7224 - val_loss: 0.5463 - val_accuracy: 0.7243
Epoch 4/5
391/391 [==============================] - 581s 1s/step - loss: 0.5088 -
accuracy: 0.7586 - val_loss: 0.4716 - val_accuracy: 0.7786
Epoch 5/5
391/391 [==============================] - 1240s 3s/step - loss: 0.4638 -
accuracy: 0.7881 - val_loss: 0.4839 - val_accuracy: 0.7760
118/118 [==============================] - 33s 279ms/step - loss: 0.4913
- accuracy: 0.7659
```

In [66]:
```
model_4_time
```

Out[66]:
```
63.855385590000004
```

In [67]:
```
#S14 Create Predicited Probabilties
y_proba_4 = model_4.predict(X_valid)
y_proba_4.round(2)
```

Out[67]:
```
array([[0.72],
       [0.85],
       [0.29],
       ...,
       [0.21],
       [0.92],
       [0.49]], dtype=float32)
```

In [68]:
```
#Score test dataset
scr=model_4.predict_classes(X_test)
#Conver array to Pandas dataframe with submission titles
pd_scr=pd.DataFrame(scr)
pd_scr.index.name = 'ImageId'
pd_scr.columns = ['label']
print(pd_scr)
#Export to Excel
pd_scr.to_excel("catsdogs_4.xlsx")
```

```
         label
ImageId       
0            0
1            0
2            0
3            0
4            1
...        ...
8745         0
8746         0
8747         0
8748         1
8749         1

[8750 rows x 1 columns]
```

In [69]:
```
model_4_train_acc = max(history_4.history["accuracy"])
```

In [70]:
```
model_4_val_loss, model_4_val_acc = model_4.evaluate(X_valid, y_valid)
model_4_val_acc
```

```
118/118 [==============================] - 34s 291ms/step - loss: 0.4913
- accuracy: 0.7659
```

Out[70]: 0.7658666372299194

In [71]:
```
model_4_test_loss, model_4_test_acc = score_4
model_4_test_acc
```

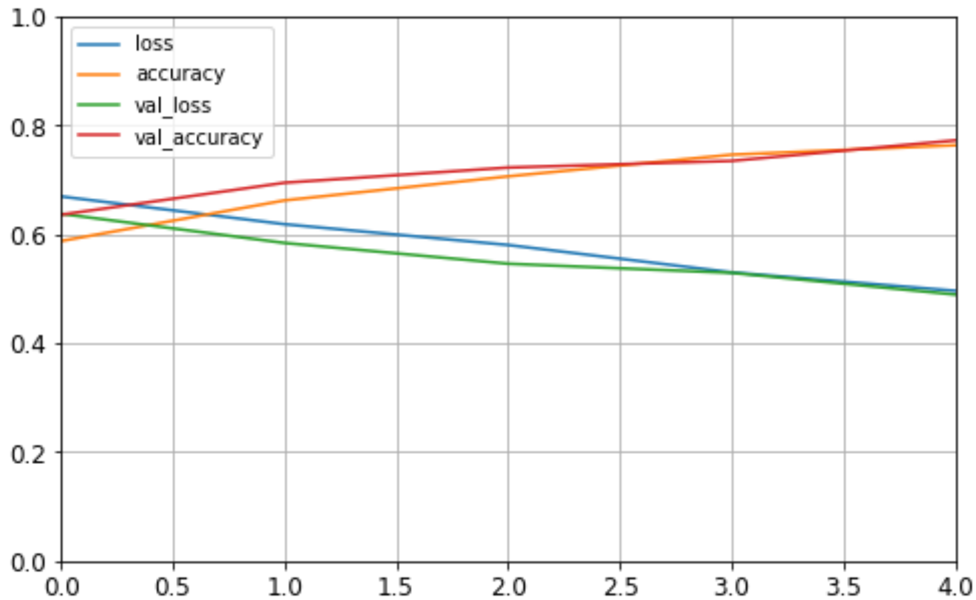
Out[71]: 0.7658666372299194

In [84]:
```
model_1_time = (end_time_1 - start_time_1)/600
model_2_time = (end_time_2 - start_time_2)/600
model_3_time = (end_time_3 - start_time_3)/600
model_4_time = (end_time_4 - start_time_4)/600
```

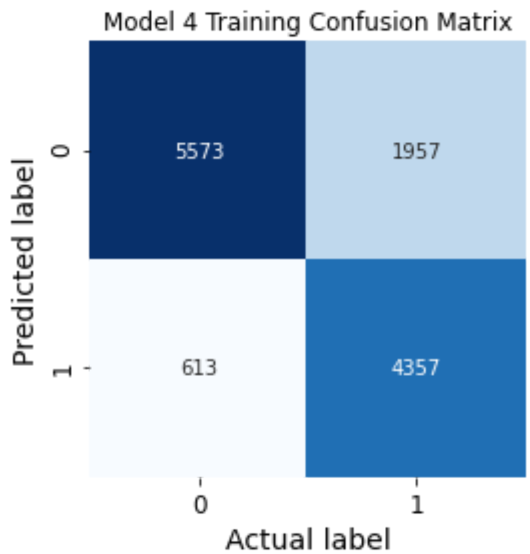
In [73]:
```
model_4.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 64, 64, 64)        3200      
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 32, 32, 64)        0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 256)       147712    
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 32, 32, 256)       590080    
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 256)       0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 256)       590080    
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 16, 16, 256)       590080    
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 8, 256)         0         
_________________________________________________________________
flatten (Flatten)            (None, 16384)             0         
_________________________________________________________________
dense (Dense)                (None, 128)               2097280   
_________________________________________________________________
dropout (Dropout)            (None, 128)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256      
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0         
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 65        
=================================================================
Total params: 4,026,753
Trainable params: 4,026,753
Non-trainable params: 0
_________________________________________________________________
```

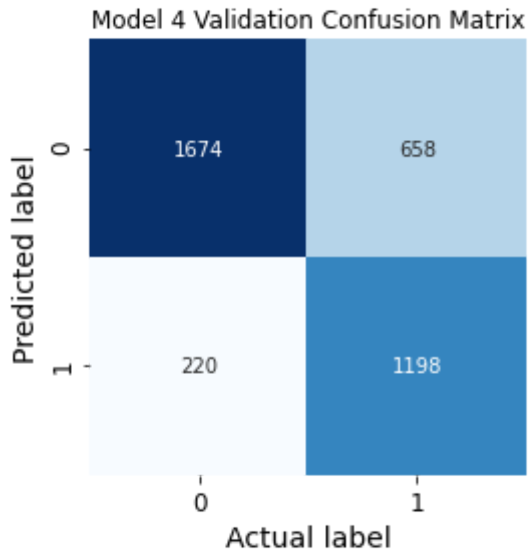
In [74]:
```
#S13 View History
history_4.params
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
#save_fig("keras_learning_curves_plot")
plt.show()
```



In [75]:
```
#Plot Confusion Matrix DNN
cm_tst = confusion_matrix(y_train, model_4.predict_classes(X_train))
cm_tst_plt=sns.heatmap(cm_tst.T, square=True, annot=True, fmt='d', cbar=Fal
plt.xlabel('Actual label')
plt.ylabel('Predicted label')
plt.title("Model 4 Training Confusion Matrix");
fig3 = cm_tst_plt.get_figure()
fig3.savefig('TestCMHL2NPL300100.png',
        bbox_inches = 'tight', dpi=None, facecolor='w', edgecolor='b',
        orientation='portrait', papertype=None, format=None,
        transparent=True, pad_inches=0.25)
```



In [88]:
```
#Plot Confusion Matrix DNN
cm_tst = confusion_matrix(y_valid, model_4.predict_classes(X_valid))
cm_tst_plt=sns.heatmap(cm_tst.T, square=True, annot=True, fmt='d', cbar=Fal
plt.xlabel('Actual label')
plt.ylabel('Predicted label')
plt.title("Model 4 Validation Confusion Matrix");
fig3 = cm_tst_plt.get_figure()
fig3.savefig('TestCMHL2NPL300100.png',
        bbox_inches = 'tight', dpi=None, facecolor='w', edgecolor='b',
        orientation='portrait', papertype=None, format=None,
        transparent=True, pad_inches=0.25)
```



In [89]:
```
data = [[1,128, "relu", round(model_1_time,2), round(model_1_train_acc,4),
         [2,256, "tahn",round(model_2_time,2), round(model_2_train_acc,4), r
         [3,128, "tahn", round(model_3_time,2), round(model_3_train_acc,4),
         [4,256, "relu",round(model_4_time,2),  round(model_4_train_acc,4),


df = pd.DataFrame(data, columns =
                    ['Model Number' ,'Filter Count', 'Activation', 'Processin
                     'Training Set Accuracy', "Validation Set Accuracy", "Tes

df
```

Out[89]:

	Model Number	Filter Count	Activation	Processing Time (Min)	Training Set Accuracy	Validation Set Accuracy	Test Set Accuracy
0	1	128	relu	15.57	0.7636	0.773	0.757
1	2	256	tahn	49.71	0.7704	0.765	0.765
2	3	128	tahn	16.21	0.7765	0.757	0.747
3	4	256	relu	63.86	0.7881	0.766	0.766

In [86]:
```
data = [[1,2, 10,round(model_1_time,2), round(model_1_train_acc,3), round(m
         [2,6, 10,round(model_2_time,2), round(model_2_train_acc,3), round(m
         [3,2, 40,round(model_3_time,2), round(model_3_train_acc,3), round(m
         [4,6, 40,round(model_4_time,2), round(model_4_train_acc,3), round(m
df = pd.DataFrame(data, columns =
                    ['Model Number' ,'Filter Count', 'Nodes per Layer', 'Proc
                     'Training Set Accuracy', "Validation Set Accuracy", "Tes
df
```

Out[86]:

	Model Number	Number of Hidden Layers	Nodes per Layer	Processing Time (Min)	Training Set Accuracy	Validation Set Accuracy	Test Set Accuracy
0	1	2	10	15.57	0.764	0.773	0.756533
1	2	6	10	49.71	0.770	0.765	0.765333
2	3	2	40	16.21	0.776	0.757	0.747467
3	4	6	40	63.86	0.788	0.766	0.765867

In [ ]:

In [9]:
```
import os, random
import numpy as np
import tensorflow as tf
from matplotlib import pyplot as plt
%matplotlib inline
import pickle
import tensorflow as tf
from datetime import datetime
import os
import keras
import utils
from keras.models import Sequential
import pandas as pd
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dense, Flatten, Dropout, Embedding,LST
import cv2, itertools
from keras.utils import np_utils
from matplotlib import pyplot
from matplotlib.image import imread
import re
from keras import backend as K
from sklearn.model_selection import train_test_split
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
```

```
WARNING:tensorflow:From /Users/allisonroeser/opt/anaconda3/lib/python3.7/
site-packages/tensorflow/python/compat/v2_compat.py:96: disable_resource_
variables (from tensorflow.python.ops.variable_scope) is deprecated and w
ill be removed in a future version.
Instructions for updating:
non-resource variables are not supported in the long term
```

In [2]:
```
TRAIN_DIR = '/Users/allisonroeser/Desktop/train/'
TEST_DIR = '/Users/allisonroeser/Desktop/test/'

ROWS = 64
COLS = 64
CHANNELS = 3
```

using code from to import data and get setup: [https://www.kaggle.com/freeman89/create-dataset-with-tensorflow (https://www.kaggle.com/freeman89/create-dataset-with-tensorflow)](https://www.kaggle.com/freeman89/create-dataset-with-tensorflow)

In [3]:
```
# On the kaggle notebook
# we only take the first 2000 from the training set
# and only the first 1000 from the test set
# REMOVE [0:2000] and [0:1000] when running locally
train_image_file_names = [TRAIN_DIR+i for i in os.listdir(TRAIN_DIR)][0:200
test_image_file_names = [TEST_DIR+i for i in os.listdir(TEST_DIR)][0:1000]
```

In [4]:
```
# Slow, yet simple implementation with tensorflow
# could be rewritten to be much faster
# (which is not really needed as it takes less than 5 minutes on my laptop)
def decode_image(image_file_names, resize_func=None):

    images = []

    graph = tf.Graph()
    with graph.as_default():
        file_name = tf.placeholder(dtype=tf.string)
        file = tf.read_file(file_name)
        image = tf.image.decode_jpeg(file)
        if resize_func != None:
            image = resize_func(image)

    with tf.Session(graph=graph) as session:
        tf.initialize_all_variables().run()
        for i in range(len(image_file_names)):
            images.append(session.run(image, feed_dict={file_name: image_fi
            if (i+1) % 1000 == 0:
                print('Images processed: ',i+1)

        session.close()

    return images
```

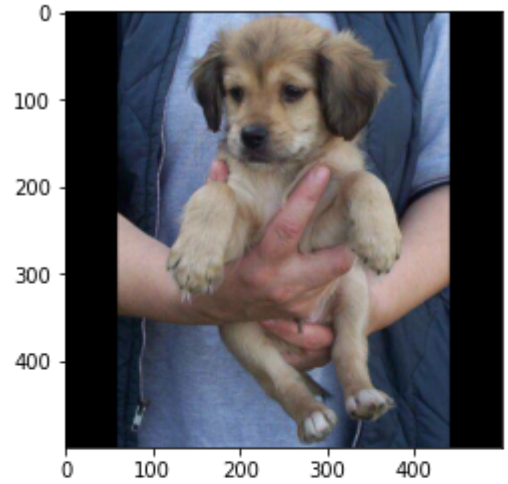
In [ ]:
```
train_images = decode_image(train_image_file_names)
test_images = decode_image(test_image_file_names)
all_images = train_images + test_images
```

In [10]:
```
WIDTH=500
HEIGHT=500
resize_func = lambda image: tf.image.resize_image_with_crop_or_pad(image, H
```

In [11]:
```
processed_train_images = decode_image(train_image_file_names, resize_func=r
processed_test_images = decode_image(test_image_file_names, resize_func=res
```

```
WARNING:tensorflow:From /Users/allisonroeser/opt/anaconda3/lib/python3.7/
site-packages/tensorflow/python/util/tf_should_use.py:235: initialize_all
_variables (from tensorflow.python.ops.variables) is deprecated and will
be removed after 2017-03-02.
Instructions for updating:
Use `tf.global_variables_initializer` instead.
Images processed:  1000
Images processed:  2000
Images processed:  1000
```

In [12]:
```
# Let's check how the images look like
for i in range(10):
    plt.imshow(processed_train_images[i])
    plt.show()
```



In [13]:
```
labels = [1 if 'dog' in name else 0 for name in train_image_file_names]
```

In [14]:
```
label_names = ["Dog" if 'dog' in name else "Cat" for name in train_image_fi
```

In [15]:
```
len(train_image_file_names)
```

Out[15]: 2000

In [16]:
```
len(test_image_file_names)
```

Out[16]: 1000

In [17]:
```
len(processed_train_images)
```

Out[17]: 2000

In [18]:
```
labels
```

Out[18]:
```
[1,
 0,
 1,
 0,
 0,
 0,
 1,
 0,
 0,
 1,
 0,
 0,
 0,
 1,
 1,
 1,
 0,
 1,
 1,
```

```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(label_names[i])
plt.show()
```

In [20]:
```
label_df = pd.DataFrame()
label_df["label"] = label_names
```

In [21]:
```
label_df["label"].value_counts()
```

Out[21]:
```
Dog    1014
Cat     986
Name: label, dtype: int64
```

In [22]:
```
label_df["label"]
```

Out[22]:
```
0       Dog
1       Cat
2       Dog
3       Cat
4       Cat
       ...
1995    Cat
1996    Cat
1997    Dog
1998    Dog
1999    Dog
Name: label, Length: 2000, dtype: object
```

In [23]:
```
X = np.array(processed_train_images).reshape(-1, 80,80,1)
```

In [24]:
```
len(X)
```

Out[24]:
```
234375
```

```
model_1 = Sequential()

model_1.add(Dense(units=32, activation='relu', input_shape= X.shape[1:]))
model_1.add(Dense(units=num_classes, activation='relu'))
model_1.add(Dense(units=num_classes, activation='relu'))
model_1.add(Dense(units=num_classes, activation='softmax'))
model_1.summary()
```

In [ ]:
```
model = Sequential()

model.add(Conv2D(32, (3, 3), input_shape= X.shape[1:]))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (3, 3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(16))
model.add(Activation("relu"))
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation("sigmoid"))

model.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics=["accuracy"])
```

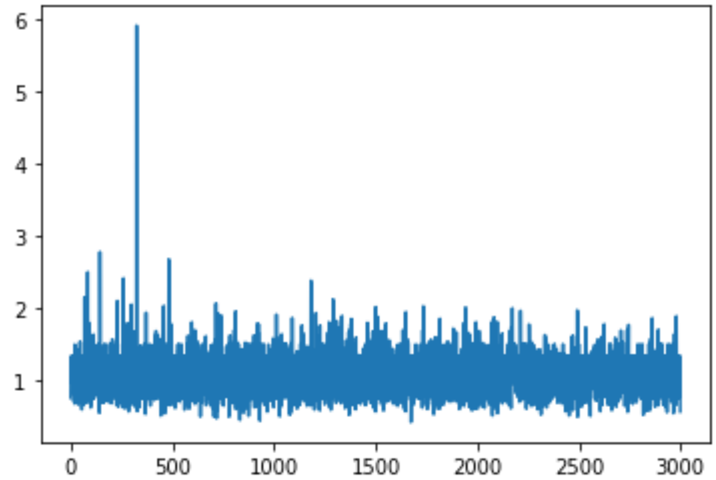
In [28]:
```
train_images = decode_image(train_image_file_names)
test_images = decode_image(test_image_file_names)
all_images = train_images + test_images
```

```
Images processed:  1000
Images processed:  2000
Images processed:  1000
```

In [29]:
```
# Check mean aspect ratio (width/height), mean width and mean height
width = []
height = []
aspect_ratio = []
for image in all_images:
    h, w, d = np.shape(image)
    aspect_ratio.append(float(w) / float(h))
    width.append(w)
    height.append(h)
```

In [30]:
```
print('Mean aspect ratio: ',np.mean(aspect_ratio))
plt.plot(aspect_ratio)
plt.show()
```

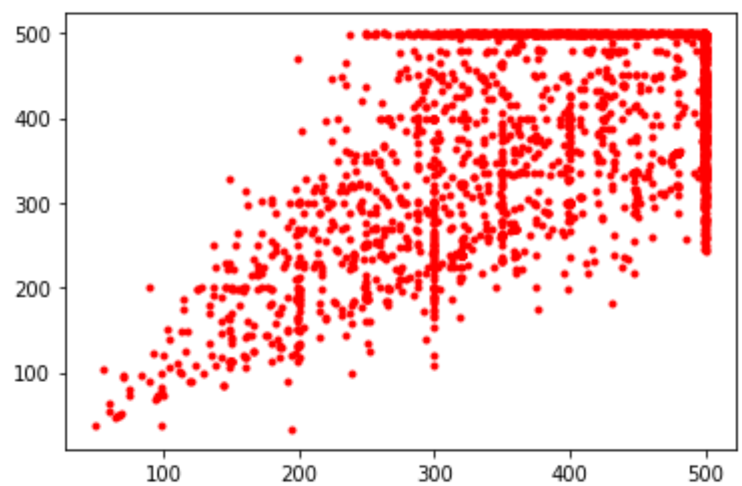
```
Mean aspect ratio:  1.1517522090952068
```



In [ ]:

In [31]:
```
print('Mean width:',np.mean(width))
print('Mean height:',np.mean(height))
plt.plot(width, height, '.r')
plt.show()
```

```
Mean width: 403.5473333333333
Mean height: 362.759
```



In [ ]:

In [ ]: