WashburnPaulHW5

November 13, 2018

0.0.1 CSCI E-82 Homework 5 on CNNs

0.0.2 Due by 11/13/18 at 11:59pm EST to the Canvas dropbox

0.1 This is an individual homework so there should be no collaboration for this homework.

0.1.1 Under each problem, we have a place for you to write the answer, or write runnable code that will produce the answer. Show your work.

This is a busy time of year with homework and an exam coming up. We are looking for a successful working result that builds upon the section code and enables you to gain some proficiency with this important and growing field of deep learning.

Depending on your computer, some of the runs may still take a few minutes per epoch. As a result, Problem 4 may take the better part of a day to run, so plan accordingly.

0.2 Your Name:

Paul M. Washburn

Some ideas for project:

- trump tweets vs. stock and bond markets
- topic extraction on trump tweets

0.3 Dataset

WikiArt is an amazing resource containing centuries of artwork. Since such datasets are wonderful for deep learning, Kaggle has hosted a challenge to characterize the 'fingerprints' of various artists. The Kaggle dataset contains metadata and also a set of images that have been resized so that the shorter dimension is 256 pixels. To make this homework reasonably fast even for those without GPUs, we have further reduced the images to 64 x 64. CNNs and neural networks in general prefer to have consistent sizes. To achieve this, we cut the center 256 pixels from the longer dimension and then shrunk the images by a factor of 4. This isn't a perfect solution since it did cut off a few heads as you will see.

The selected images are for portraits and landscapes. No, we're not talking about the orientation but rather the content of the images. Thanks to help from Rashmi and Dave, we have a small enough data set that should give reasonable results in a timely manner even on just a CPU.

The data were originally divided into a training and a test set. We have further divided the training set into a train and validation set. In this homework you will be using the training set and validation set to train and assess your deep learning models. At the final step, you will see how well your final training worked on the test set. In each of these directories, there is a truth.txt file that has the image name and whether it is a portrait or landscape scene.

```
In [1]: import warnings
        warnings.filterwarnings('ignore')
        import tensorflow as tf
        import pandas as pd
        import numpy as np
        import os
        import shutil
        from matplotlib import pyplot as plt
        from tensorflow import keras
        from PIL import Image
        import glob
        from tensorflow.keras.optimizers import *
        from tensorflow.keras.layers import *
        from tensorflow.keras.models import *
        from tensorflow.keras.preprocessing.image import ImageDataGenerator
        from tensorflow.keras import backend as K
        from tensorflow.keras.callbacks import TensorBoard, EarlyStopping
        from time import time
        from sklearn.metrics import roc_curve, auc
        import seaborn as sns
        sns.set(style="whitegrid")
        %matplotlib inline

        def set_mpl_preferences(ax):
            ax.grid(alpha=.3)
            sns.despine()
            ax.legend(loc='best')
            sns.set(style="whitegrid")

        if K.backend()=='tensorflow':
            K.set_image_data_format('channels_last')

        print("Tensorflow is installed and is version: ",  tf.__version__)
        print("Keras is installed and is version: ", tf.keras.__version__)
```

```
Tensorflow is installed and is version:  1.10.0
Keras is installed and is version:  2.1.6-tf
```

```
In [2]: class TrainValTensorBoard(TensorBoard):
            def __init__(self, log_dir='./logs/{}'.format(time()), **kwargs):
                # Make the original `TensorBoard` log to a subdirectory 'training'
                training_log_dir = os.path.join(log_dir, 'training')
                super(TrainValTensorBoard, self).__init__(training_log_dir, **kwargs)
```

```
            # Log the validation metrics to a separate subdirectory
            self.val_log_dir = os.path.join(log_dir, 'validation')

        def set_model(self, model):
            # Setup writer for validation metrics
            self.val_writer = tf.summary.FileWriter(self.val_log_dir)
            super(TrainValTensorBoard, self).set_model(model)

        def on_epoch_end(self, epoch, logs=None):
            # Pop the validation logs and handle them separately with
            # `self.val_writer`. Also rename the keys so that they can
            # be plotted on the same figure with the training metrics
            logs = logs or {}
            val_logs = {k.replace('val_', ''): v for k, v in logs.items() if k.startswith(
            for name, value in val_logs.items():
                summary = tf.Summary()
                summary_value = summary.value.add()
                summary_value.simple_value = value.item()
                summary_value.tag = name
                self.val_writer.add_summary(summary, epoch)
            self.val_writer.flush()

            # Pass the remaining logs to `TensorBoard.on_epoch_end`
            logs = {k: v for k, v in logs.items() if not k.startswith('val_')}
            super(TrainValTensorBoard, self).on_epoch_end(epoch, logs)

        def on_train_end(self, logs=None):
            super(TrainValTensorBoard, self).on_train_end(logs)
            self.val_writer.close()
```

```
In [3]: base_dir = 'data/images64'
        train_dir = os.path.join(base_dir, 'train')
        validation_dir = os.path.join(base_dir, 'validation')
        test_dir = os.path.join(base_dir, 'test')
```

```
In [4]: train_labels = pd.read_table('data/images64/train/truth.txt', header=None)
        val_labels = pd.read_table('data/images64/validation/truth.txt', header=None)
```

Do Not Re-run Code Below, Only Run Once

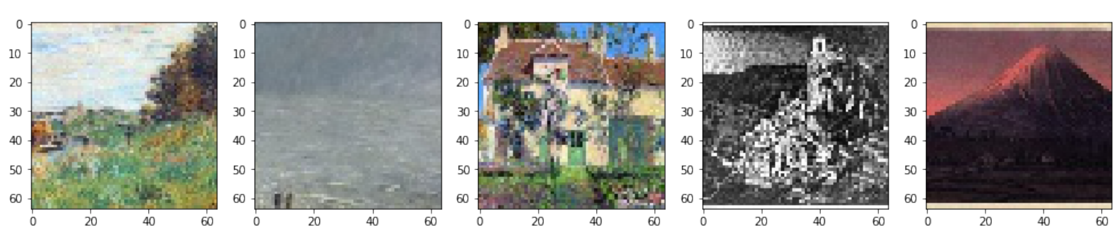
```
os.mkdir(os.path.join(train_dir, 'landscape'))
os.mkdir(os.path.join(train_dir, 'portrait'))
os.mkdir(os.path.join(validation_dir, 'landscape'))
os.mkdir(os.path.join(validation_dir, 'portrait'))

# move training files
for landscape_pic in train_labels.loc[train_labels[1]=='landscape', 0]:
    current_home = os.path.join(train_dir, landscape_pic)
    new_home = os.path.join(os.path.join(train_dir, 'landscape'), landscape_pic)
```

```
    shutil.move(current_home, new_home)

for portrait_pic in train_labels.loc[train_labels[1]=='portrait', 0]:
    current_home = os.path.join(train_dir, portrait_pic)
    new_home = os.path.join(os.path.join(train_dir, 'portrait'), portrait_pic)
    shutil.move(current_home, new_home)

# move validation files
for landscape_pic in val_labels.loc[val_labels[1]=='landscape', 0]:
    current_home = os.path.join(validation_dir, landscape_pic)
    new_home = os.path.join(os.path.join(validation_dir, 'landscape'), landscape_pic)
    shutil.move(current_home, new_home)

for portrait_pic in val_labels.loc[val_labels[1]=='portrait', 0]:
    current_home = os.path.join(validation_dir, portrait_pic)
    new_home = os.path.join(os.path.join(validation_dir, 'portrait'), portrait_pic)
    shutil.move(current_home, new_home)
```

0.4 Problem 1 (5 points)

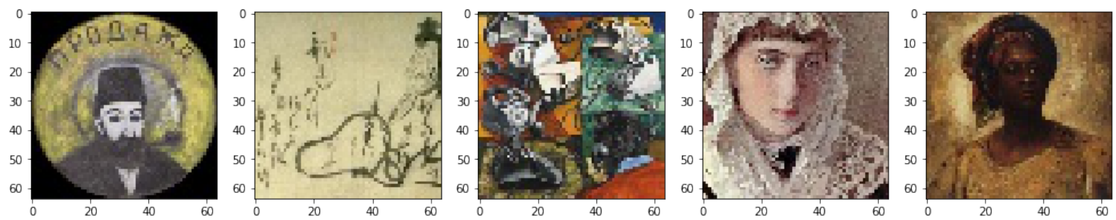
Read in and display the first 5 portraits and the first 5 landscapes. Note, if you are using the OpenCV tools, then the color may be distorted. The cvtColor() method using cv2.COLOR_BGR2RGB may be useful. However, it is likely easier to use the generator and plot_strip example from section.

```
In [5]: train_files = glob.glob('data/images64/train/landscape/*.jpg')
        fig, axes = plt.subplots(1, 5, figsize=(17, 5))
        for i, f in enumerate(train_files[:5]):
            ax = axes[i]
            img = Image.open(f)
            ax.imshow(np.asarray(img))
            img.close()

        plt.show()
```



```
In [6]: train_files = glob.glob('data/images64/train/portrait/*.jpg')
        fig, axes = plt.subplots(1, 5, figsize=(17, 5))
        for i, f in enumerate(train_files[:5]):
```

```
    ax = axes[i]
    img = Image.open(f)
    ax.imshow(np.asarray(img))
    img.close()

plt.show()
```



0.5 Problem 2 (25 points)

Construct a baseline CNN classifier using Keras for the training set and assess the validation set performance at each epoch. The goal is to correctly classify portraits from landscapes. Plot the resulting performance on the training and validation set as a function of epoch using the criteria over which you are optimizing. You should run at least 20 epochs for this problem.

In [7]:
```
# select batch_size
batch_size = 128

train_datagen = ImageDataGenerator(rescale=1./255,
                                   #featurewise_center=True,
                                   #featurewise_std_normalization=True,
                                   rotation_range=20,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   horizontal_flip=True)
train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(64, 64),
        batch_size=batch_size, # somewhat arbitrarily chosen
        class_mode='binary')

val_datagen = ImageDataGenerator(rescale=1./255,
                                #featurewise_center=True,
                                #featurewise_std_normalization=True,
                                rotation_range=20,
                                width_shift_range=0.2,
                                height_shift_range=0.2,
                                horizontal_flip=True)
validation_generator = val_datagen.flow_from_directory(validation_dir,
```

```
                                                                target_size=(64, 64),
                                                                batch_size=batch_size,
                                                                class_mode='binary')
```

```
Found 16315 images belonging to 2 classes.
Found 8158 images belonging to 2 classes.
```

```
In [8]: np.random.seed(777)
```

```
In [9]: K.clear_session()

        pool_size = (3,3)

        early_stopping = EarlyStopping(patience=3)
        tensorboard = TensorBoard(log_dir="logs/{}".format(time()))

        model = Sequential(name='cnn')

        # first convolution
        model.add(Conv2D(32, (5, 5), activation='relu',
                         input_shape=(64, 64, 3),
                         name='conv1',
                         padding='same'))
        model.add(AveragePooling2D(pool_size, name='avg_pool1'))

        # second convolution
        model.add(Conv2D(64, (5, 5), activation='relu', name = 'conv2', padding='same'))
        model.add(Conv2D(32, (5, 5), activation='relu', name = 'conv3', padding='same'))
        model.add(AveragePooling2D(pool_size, name='avg_pool2'))

        model.add(Flatten())
        model.add(Dense(256, activation='relu', name='fc1')) #128
        model.add(Dense(1, activation='sigmoid', name='fc2'))

        sgd = SGD(lr = 0.05, decay=1e-6, momentum=0.9, nesterov=True)

        model.compile(loss='binary_crossentropy',
                      #optimizer=optimizers.RMSprop(lr=1e-4),
                      #optimizer=sgd,
                      optimizer=Adam(lr=1e-4, decay=1e-6),
                      metrics=['accuracy'])


        # fit model
        history = model.fit_generator(train_generator,
                                      steps_per_epoch=100,
                                      epochs=25,
```

```
                                      validation_data=validation_generator,
                                      validation_steps=50,
                                      verbose=1,
                                      callbacks=[tensorboard, early_stopping])

model.summary()
```

```
Epoch 1/25
100/100 [==============================] - 84s 843ms/step - loss: 0.5473 - acc: 0.7303 - val_lo
Epoch 2/25
100/100 [==============================] - 83s 827ms/step - loss: 0.4341 - acc: 0.8016 - val_lo
Epoch 3/25
100/100 [==============================] - 84s 839ms/step - loss: 0.3876 - acc: 0.8259 - val_lo
Epoch 4/25
100/100 [==============================] - 83s 831ms/step - loss: 0.3623 - acc: 0.8427 - val_lo
Epoch 5/25
100/100 [==============================] - 83s 834ms/step - loss: 0.3340 - acc: 0.8574 - val_lo
Epoch 6/25
100/100 [==============================] - 79s 793ms/step - loss: 0.3306 - acc: 0.8629 - val_lo
Epoch 7/25
100/100 [==============================] - 79s 792ms/step - loss: 0.3082 - acc: 0.8728 - val_lo
Epoch 8/25
100/100 [==============================] - 79s 785ms/step - loss: 0.3155 - acc: 0.8682 - val_lo
Epoch 9/25
100/100 [==============================] - 79s 789ms/step - loss: 0.2934 - acc: 0.8770 - val_lo
Epoch 10/25
100/100 [==============================] - 79s 786ms/step - loss: 0.2974 - acc: 0.8764 - val_lo
Epoch 11/25
100/100 [==============================] - 79s 793ms/step - loss: 0.2822 - acc: 0.8873 - val_lo
Epoch 12/25
100/100 [==============================] - 78s 784ms/step - loss: 0.2735 - acc: 0.8923 - val_lo
Epoch 13/25
100/100 [==============================] - 79s 787ms/step - loss: 0.2764 - acc: 0.8917 - val_lo
Epoch 14/25
100/100 [==============================] - 79s 785ms/step - loss: 0.2788 - acc: 0.8858 - val_lo
Epoch 15/25
100/100 [==============================] - 79s 791ms/step - loss: 0.2661 - acc: 0.8950 - val_lo
Epoch 16/25
100/100 [==============================] - 78s 784ms/step - loss: 0.2656 - acc: 0.8948 - val_lo
Epoch 17/25
100/100 [==============================] - 79s 790ms/step - loss: 0.2685 - acc: 0.8922 - val_lo
Epoch 18/25
100/100 [==============================] - 79s 788ms/step - loss: 0.2628 - acc: 0.8948 - val_lo
Epoch 19/25
100/100 [==============================] - 79s 793ms/step - loss: 0.2591 - acc: 0.8971 - val_lo
Epoch 20/25
100/100 [==============================] - 78s 784ms/step - loss: 0.2516 - acc: 0.8990 - val_lo
Epoch 21/25
```

```
100/100 [==============================] - 78s 785ms/step - loss: 0.2512 - acc: 0.8991 - val_lo
Epoch 22/25
100/100 [==============================] - 79s 787ms/step - loss: 0.2431 - acc: 0.9035 - val_lo
Epoch 23/25
100/100 [==============================] - 79s 787ms/step - loss: 0.2454 - acc: 0.9038 - val_lo
Epoch 24/25
100/100 [==============================] - 79s 786ms/step - loss: 0.2403 - acc: 0.9058 - val_lo
Epoch 25/25
100/100 [==============================] - 79s 785ms/step - loss: 0.2550 - acc: 0.8972 - val_lo
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1 (Conv2D)               (None, 64, 64, 32)        896
_________________________________________________________________
avg_pool1 (AveragePooling2D) (None, 21, 21, 32)        0
_________________________________________________________________
conv2 (Conv2D)               (None, 21, 21, 64)        51264
_________________________________________________________________
conv3 (Conv2D)               (None, 21, 21, 32)        51232
_________________________________________________________________
avg_pool2 (AveragePooling2D) (None, 7, 7, 32)          0
_________________________________________________________________
flatten (Flatten)            (None, 1568)              0
_________________________________________________________________
fc1 (Dense)                  (None, 256)               401664
_________________________________________________________________
fc2 (Dense)                  (None, 1)                 257
=================================================================
Total params: 505,313
Trainable params: 505,313
Non-trainable params: 0
_________________________________________________________________
```

```
In [10]: model.save_weights('models/attempt_0.h5')
```

To start tensorboard:

```
Pauls-MacBook-Pro:HW5 pmw$ tensorboard --logdir=logs
```

0.6 Problem 3 (5 points)

From the pattern of training and validation curves, describe what is good/bad and what you plan to do next to improve the result.

0.7 Problem 4 (45 points)

This step is where we want you to do most of your personal learning. Your goal is to improve the network using a combination of architecture choices, parameter tuning, and experimenting with different optimizers/dropout/regularization/etc. Treat each of these as separate optimization/exploration steps for now. We would like to see 3 separate steps that cover different areas. The format of the 3 steps should be as follows: * State the hypothesis/strategy for how you will improve/explore a particular aspect. * Describe what types of tests you are running and why (i.e. what range of parameters are you choosing and why) * Include the code and results * State your interpretation of the results

We're not looking for research in deep learning, but we want you to gain some hands-on experience working with Keras and figuring out what works. A good example may be comparing strategies to overcome overfitting, or comparing a few different CNN architectures in terms of performance and speed, or comparing data augmentation types and results.

0.7.1 Hypothesis 1: Adding another full convolution layer

In [11]:
```
np.random.seed(777)

K.clear_session()

pool_size = (3, 3)

early_stopping = EarlyStopping(patience=3)
tensorboard = TensorBoard(log_dir="logs/{}".format(time()))

model = Sequential(name='cnn')

# first convolution
model.add(Conv2D(32, (5, 5), activation='relu',
                 input_shape=(64, 64, 3),
                 name='conv1',
                 padding='same'))
model.add(AveragePooling2D(pool_size, name='avg_pool1'))

# second convolution
model.add(Conv2D(64, (5, 5), activation='relu', name = 'conv2', padding='same'))
model.add(Conv2D(32, (5, 5), activation='relu', name = 'conv3', padding='same'))
model.add(AveragePooling2D(pool_size, name='avg_pool2'))

# third - new - convolution
model.add(Conv2D(64, (5, 5), activation='relu', name = 'conv4', padding='same'))
model.add(Conv2D(32, (5, 5), activation='relu', name = 'conv5', padding='same'))
model.add(AveragePooling2D(pool_size, name='avg_pool3'))

model.add(Flatten())
model.add(Dense(256, activation='relu', name='fc1'))
model.add(Dense(1, activation='sigmoid', name='fc2'))

sgd = SGD(lr = 0.05, decay=1e-6, momentum=0.9, nesterov=True)

model.compile(loss='binary_crossentropy',
```

```
                     #optimizer=optimizers.RMSprop(lr=1e-4),
                     #optimizer=sgd,
                     optimizer=Adam(lr=1e-4, decay=1e-6),
                     metrics=['accuracy'])


# fit model
history = model.fit_generator(train_generator,
                              steps_per_epoch=100,
                              epochs=25,
                              validation_data=validation_generator,
                              validation_steps=50,
                              verbose=1,
                              callbacks=[tensorboard, early_stopping])

model.summary()
```

```
Epoch 1/25
100/100 [==============================] - 85s 855ms/step - loss: 0.5893 - acc: 0.7049 - val_lo
Epoch 2/25
100/100 [==============================] - 86s 859ms/step - loss: 0.4429 - acc: 0.7985 - val_lo
Epoch 3/25
100/100 [==============================] - 82s 823ms/step - loss: 0.3886 - acc: 0.8257 - val_lo
Epoch 4/25
100/100 [==============================] - 84s 841ms/step - loss: 0.3762 - acc: 0.8370 - val_lo
Epoch 5/25
100/100 [==============================] - 83s 834ms/step - loss: 0.3518 - acc: 0.8489 - val_lo
Epoch 6/25
100/100 [==============================] - 83s 827ms/step - loss: 0.3418 - acc: 0.8550 - val_lo
Epoch 7/25
100/100 [==============================] - 83s 828ms/step - loss: 0.3257 - acc: 0.8648 - val_lo
Epoch 8/25
100/100 [==============================] - 84s 840ms/step - loss: 0.3170 - acc: 0.8723 - val_lo
Epoch 9/25
100/100 [==============================] - 84s 837ms/step - loss: 0.3145 - acc: 0.8670 - val_lo
Epoch 10/25
100/100 [==============================] - 83s 834ms/step - loss: 0.2963 - acc: 0.8798 - val_lo
Epoch 11/25
100/100 [==============================] - 83s 833ms/step - loss: 0.2983 - acc: 0.8791 - val_lo
Epoch 12/25
100/100 [==============================] - 84s 840ms/step - loss: 0.2995 - acc: 0.8748 - val_lo
Epoch 13/25
100/100 [==============================] - 84s 836ms/step - loss: 0.2854 - acc: 0.8838 - val_lo
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1 (Conv2D)               (None, 64, 64, 32)        896
_________________________________________________________________
```

```
avg_pool1 (AveragePooling2D) (None, 21, 21, 32)        0
_________________________________________________________________
conv2 (Conv2D)               (None, 21, 21, 64)        51264
_________________________________________________________________
conv3 (Conv2D)               (None, 21, 21, 32)        51232
_________________________________________________________________
avg_pool2 (AveragePooling2D) (None, 7, 7, 32)          0
_________________________________________________________________
conv4 (Conv2D)               (None, 7, 7, 64)          51264
_________________________________________________________________
conv5 (Conv2D)               (None, 7, 7, 32)          51232
_________________________________________________________________
avg_pool3 (AveragePooling2D) (None, 2, 2, 32)          0
_________________________________________________________________
flatten (Flatten)            (None, 128)               0
_________________________________________________________________
fc1 (Dense)                  (None, 256)               33024
_________________________________________________________________
fc2 (Dense)                  (None, 1)                 257
=================================================================
Total params: 239,169
Trainable params: 239,169
Non-trainable params: 0
_________________________________________________________________
```

```
In [12]: model.save_weights('models/attempt_1.h5')
```

0.7.2 Hypothesis 2: Using MaxPooling2D will improve the accuracy

```
In [13]: K.clear_session()

         kernel_size = (3, 3)

         early_stopping = EarlyStopping(patience=3)
         tensorboard = TensorBoard(log_dir="logs/{}".format(time()))

         model = Sequential(name='cnn')

         # first convolution
         model.add(Conv2D(32, kernel_size, activation='relu',
                          input_shape=(64, 64, 3),
                          name='conv1',
                          padding='same'))
         model.add(MaxPooling2D(kernel_size, name='max_pool1'))

         # second convolution
         model.add(Conv2D(64, (5, 5), activation='relu', name = 'conv2', padding='same'))
```

```
model.add(Conv2D(32, (5, 5), activation='relu', name = 'conv3', padding='same'))
model.add(MaxPooling2D(kernel_size, name='max_pool2'))

# third - new - convolution
model.add(Conv2D(64, (5, 5), activation='relu', name = 'conv4', padding='same'))
model.add(Conv2D(32, (5, 5), activation='relu', name = 'conv5', padding='same'))
model.add(AveragePooling2D(kernel_size, name='max_pool3'))

model.add(Flatten())
model.add(Dense(256, activation='relu', name='fc1'))
model.add(Dense(1, activation='sigmoid', name='fc2'))

sgd = SGD(lr = 0.05, decay=1e-6, momentum=0.9, nesterov=True)

model.compile(loss='binary_crossentropy',
              #optimizer=optimizers.RMSprop(lr=1e-4),
              optimizer=Adam(lr=1e-4, decay=1e-6),#sgd,
              metrics=['accuracy'])


# fit model
history = model.fit_generator(train_generator,
                              steps_per_epoch=100,
                              epochs=25,
                              validation_data=validation_generator,
                              validation_steps=50,
                              verbose=1,
                              callbacks=[tensorboard, early_stopping])

model.summary()
```

```
Epoch 1/25
100/100 [==============================] - 86s 862ms/step - loss: 0.5599 - acc: 0.7376 - val_lo
Epoch 2/25
100/100 [==============================] - 84s 843ms/step - loss: 0.3876 - acc: 0.8279 - val_lo
Epoch 3/25
100/100 [==============================] - 85s 855ms/step - loss: 0.3564 - acc: 0.8450 - val_lo
Epoch 4/25
100/100 [==============================] - 84s 841ms/step - loss: 0.3479 - acc: 0.8503 - val_lo
Epoch 5/25
100/100 [==============================] - 84s 842ms/step - loss: 0.3194 - acc: 0.8694 - val_lo
Epoch 6/25
100/100 [==============================] - 85s 851ms/step - loss: 0.3107 - acc: 0.8724 - val_lo
Epoch 7/25
100/100 [==============================] - 83s 835ms/step - loss: 0.2884 - acc: 0.8840 - val_lo
Epoch 8/25
100/100 [==============================] - 84s 845ms/step - loss: 0.3034 - acc: 0.8766 - val_lo
Epoch 9/25
```

```
100/100 [==============================] - 84s 842ms/step - loss: 0.2915 - acc: 0.8807 - val_lo
Epoch 10/25
100/100 [==============================] - 85s 849ms/step - loss: 0.2691 - acc: 0.8953 - val_lo
Epoch 11/25
100/100 [==============================] - 84s 836ms/step - loss: 0.2639 - acc: 0.8955 - val_lo
Epoch 12/25
100/100 [==============================] - 84s 841ms/step - loss: 0.2672 - acc: 0.8912 - val_lo
Epoch 13/25
100/100 [==============================] - 84s 844ms/step - loss: 0.2522 - acc: 0.8990 - val_lo
Epoch 14/25
100/100 [==============================] - 83s 831ms/step - loss: 0.2526 - acc: 0.8985 - val_lo
Epoch 15/25
100/100 [==============================] - 85s 845ms/step - loss: 0.2456 - acc: 0.8986 - val_lo
Epoch 16/25
100/100 [==============================] - 83s 830ms/step - loss: 0.2308 - acc: 0.9061 - val_lo
Epoch 17/25
100/100 [==============================] - 86s 864ms/step - loss: 0.2310 - acc: 0.9067 - val_lo
Epoch 18/25
100/100 [==============================] - 84s 837ms/step - loss: 0.2297 - acc: 0.9082 - val_lo
Epoch 19/25
100/100 [==============================] - 84s 840ms/step - loss: 0.2270 - acc: 0.9103 - val_lo
Epoch 20/25
100/100 [==============================] - 85s 846ms/step - loss: 0.2312 - acc: 0.9065 - val_lo
Epoch 21/25
100/100 [==============================] - 83s 835ms/step - loss: 0.2202 - acc: 0.9100 - val_lo
Epoch 22/25
100/100 [==============================] - 84s 842ms/step - loss: 0.2200 - acc: 0.9080 - val_lo
Epoch 23/25
100/100 [==============================] - 84s 841ms/step - loss: 0.2132 - acc: 0.9129 - val_lo
Epoch 24/25
100/100 [==============================] - 84s 844ms/step - loss: 0.2092 - acc: 0.9145 - val_lo
Epoch 25/25
100/100 [==============================] - 84s 837ms/step - loss: 0.2173 - acc: 0.9109 - val_lo
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1 (Conv2D)               (None, 64, 64, 32)        896
_________________________________________________________________
max_pool1 (MaxPooling2D)     (None, 21, 21, 32)        0
_________________________________________________________________
conv2 (Conv2D)               (None, 21, 21, 64)        51264
_________________________________________________________________
conv3 (Conv2D)               (None, 21, 21, 32)        51232
_________________________________________________________________
max_pool2 (MaxPooling2D)     (None, 7, 7, 32)          0
_________________________________________________________________
conv4 (Conv2D)               (None, 7, 7, 64)          51264
_________________________________________________________________
```

```
conv5 (Conv2D)               (None, 7, 7, 32)          51232
_________________________________________________________________
max_pool3 (AveragePooling2D) (None, 2, 2, 32)          0
_________________________________________________________________
flatten (Flatten)            (None, 128)               0
_________________________________________________________________
fc1 (Dense)                  (None, 256)               33024
_________________________________________________________________
fc2 (Dense)                  (None, 1)                 257
=================================================================
Total params: 239,169
Trainable params: 239,169
Non-trainable params: 0
_________________________________________________________________
```

```
In [14]: model.save_weights('models/attempt_2.h5')
```

0.7.3 Hypothesis 3: `kernel_size = (7, 7)` will improve the model

```
In [15]: K.clear_session()

         kernel_size = (7, 7)
         pool_size = (2, 2)

         early_stopping = EarlyStopping(patience=2)
         tensorboard = TensorBoard(log_dir="logs/{}".format(time()))

         model = Sequential(name='cnn')

         # first convolution
         model.add(Conv2D(32, kernel_size, activation='relu',
                          input_shape=(64, 64, 3),
                          name='conv1',
                          padding='same'))
         model.add(MaxPooling2D(pool_size, name='max_pool1'))

         # second convolution
         model.add(Conv2D(64, kernel_size, activation='relu', name = 'conv2', padding='same'))
         model.add(Conv2D(32, kernel_size, activation='relu', name = 'conv3', padding='same'))
         model.add(MaxPooling2D(pool_size, name='max_pool2'))

         # third - new - convolution
         model.add(Conv2D(64, kernel_size, activation='relu', name = 'conv4', padding='same'))
         model.add(Conv2D(32, kernel_size, activation='relu', name = 'conv5', padding='same'))
         model.add(AveragePooling2D(pool_size, name='max_pool3'))

         model.add(Flatten())
```

```
model.add(Dense(256, activation='relu', name='fc1'))
model.add(Dense(1, activation='sigmoid', name='fc2'))

sgd = SGD(lr = 0.05, decay=1e-6, momentum=0.9, nesterov=True)

model.compile(loss='binary_crossentropy',
              #optimizer=RMSprop(lr=1e-4),
              optimizer=Adam(lr=1e-4, decay=1e-6),#sgd,
              metrics=['accuracy'])


# fit model
history = model.fit_generator(train_generator,
                              steps_per_epoch=100,
                              epochs=25,
                              validation_data=validation_generator,
                              validation_steps=50,
                              verbose=1,
                              callbacks=[tensorboard, early_stopping])

model.summary()
```

```
Using TensorFlow backend.


Epoch 1/25
100/100 [==============================] - 310s 3s/step - loss: 0.5128 - acc: 0.7495 - val_loss
Epoch 2/25
100/100 [==============================] - 310s 3s/step - loss: 0.3852 - acc: 0.8261 - val_loss
Epoch 3/25
100/100 [==============================] - 311s 3s/step - loss: 0.3441 - acc: 0.8536 - val_loss
Epoch 4/25
100/100 [==============================] - 326s 3s/step - loss: 0.3153 - acc: 0.8680 - val_loss
Epoch 5/25
100/100 [==============================] - 327s 3s/step - loss: 0.3027 - acc: 0.8728 - val_loss
Epoch 6/25
100/100 [==============================] - 339s 3s/step - loss: 0.2771 - acc: 0.8888 - val_loss
Epoch 7/25
100/100 [==============================] - 326s 3s/step - loss: 0.2657 - acc: 0.8916 - val_loss
Epoch 8/25
100/100 [==============================] - 319s 3s/step - loss: 0.2668 - acc: 0.8903 - val_loss
Epoch 9/25
100/100 [==============================] - 321s 3s/step - loss: 0.2407 - acc: 0.9032 - val_loss
Epoch 10/25
100/100 [==============================] - 308s 3s/step - loss: 0.2372 - acc: 0.9057 - val_loss
Epoch 11/25
100/100 [==============================] - 310s 3s/step - loss: 0.2284 - acc: 0.9068 - val_loss
Epoch 12/25
```

```
100/100 [==============================] - 318s 3s/step - loss: 0.2260 - acc: 0.9061 - val_loss
Epoch 13/25
100/100 [==============================] - 308s 3s/step - loss: 0.2190 - acc: 0.9109 - val_loss
Epoch 14/25
100/100 [==============================] - 309s 3s/step - loss: 0.2118 - acc: 0.9134 - val_loss
Epoch 15/25
100/100 [==============================] - 310s 3s/step - loss: 0.2148 - acc: 0.9132 - val_loss
Epoch 16/25
100/100 [==============================] - 308s 3s/step - loss: 0.2122 - acc: 0.9134 - val_loss
Epoch 17/25
100/100 [==============================] - 312s 3s/step - loss: 0.2063 - acc: 0.9189 - val_loss
Epoch 18/25
100/100 [==============================] - 321s 3s/step - loss: 0.2039 - acc: 0.9162 - val_loss
Epoch 19/25
100/100 [==============================] - 327s 3s/step - loss: 0.2015 - acc: 0.9206 - val_loss
Epoch 20/25
100/100 [==============================] - 317s 3s/step - loss: 0.2031 - acc: 0.9155 - val_loss
Epoch 21/25
100/100 [==============================] - 310s 3s/step - loss: 0.1967 - acc: 0.9213 - val_loss
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1 (Conv2D)               (None, 64, 64, 32)        4736
_________________________________________________________________
max_pool1 (MaxPooling2D)     (None, 32, 32, 32)        0
_________________________________________________________________
conv2 (Conv2D)               (None, 32, 32, 64)        100416
_________________________________________________________________
conv3 (Conv2D)               (None, 32, 32, 32)        100384
_________________________________________________________________
max_pool2 (MaxPooling2D)     (None, 16, 16, 32)        0
_________________________________________________________________
conv4 (Conv2D)               (None, 16, 16, 64)        100416
_________________________________________________________________
conv5 (Conv2D)               (None, 16, 16, 32)        100384
_________________________________________________________________
max_pool3 (AveragePooling2D) (None, 8, 8, 32)          0
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
fc1 (Dense)                  (None, 256)               524544
_________________________________________________________________
fc2 (Dense)                  (None, 1)                 257
=================================================================
Total params: 931,137
Trainable params: 931,137
Non-trainable params: 0
_________________________________________________________________
```

```
In [16]: model.save_weights('models/attempt_3.h5')
```

0.8 Problem 5 (10 points)

Assess your best model on the test data. Plot the corresponding ROC curve from the results (since we've provided the truth). This was not directly covered in section, but will require a prediction using images in the same format as the training. We suggest referring to the Keras API else use a Google to search to find how to make predictions.

```
In [17]: test_labels = pd.read_table('data/images64/test/truth.txt', header=None)
         test_labels.head()
```

```
Out[17]:         0          1
         0   3.jpg   portrait
         1  34.jpg   portrait
         2  68.jpg   portrait
         3  87.jpg  landscape
         4  92.jpg   portrait
```

```
In [18]: test_labels[1].value_counts() / test_labels.shape[0]
```

```
Out[18]: portrait     0.531373
         landscape    0.468627
         Name: 1, dtype: float64
```

```
# run this only once
os.mkdir(os.path.join(test_dir, 'landscape'))
os.mkdir(os.path.join(test_dir, 'portrait'))

# move test files
for landscape_pic in test_labels.loc[test_labels[1]=='landscape', 0]:
    current_home = os.path.join(test_dir, landscape_pic)
    new_home = os.path.join(os.path.join(test_dir, 'landscape'), landscape_pic)
    shutil.move(current_home, new_home)

for portrait_pic in test_labels.loc[test_labels[1]=='portrait', 0]:
    current_home = os.path.join(test_dir, portrait_pic)
    new_home = os.path.join(os.path.join(test_dir, 'portrait'), portrait_pic)
    shutil.move(current_home, new_home)
```

```
In [67]: batch_size = 1

         test_datagen = ImageDataGenerator(rescale=1./255)
         test_generator = test_datagen.flow_from_directory(
                 test_dir,
                 target_size=(64, 64),
                 batch_size=128, # somewhat arbitrarily chosen
```

```
                class_mode='binary',
                shuffle=False)

        phat_test = model.predict_generator(test_generator)
        yhat_test = np.array([p > .5 for p in phat_test])
```

```
Found 7379 images belonging to 2 classes.
```

In [68]:
```
y_test = test_generator.classes
y_test
```

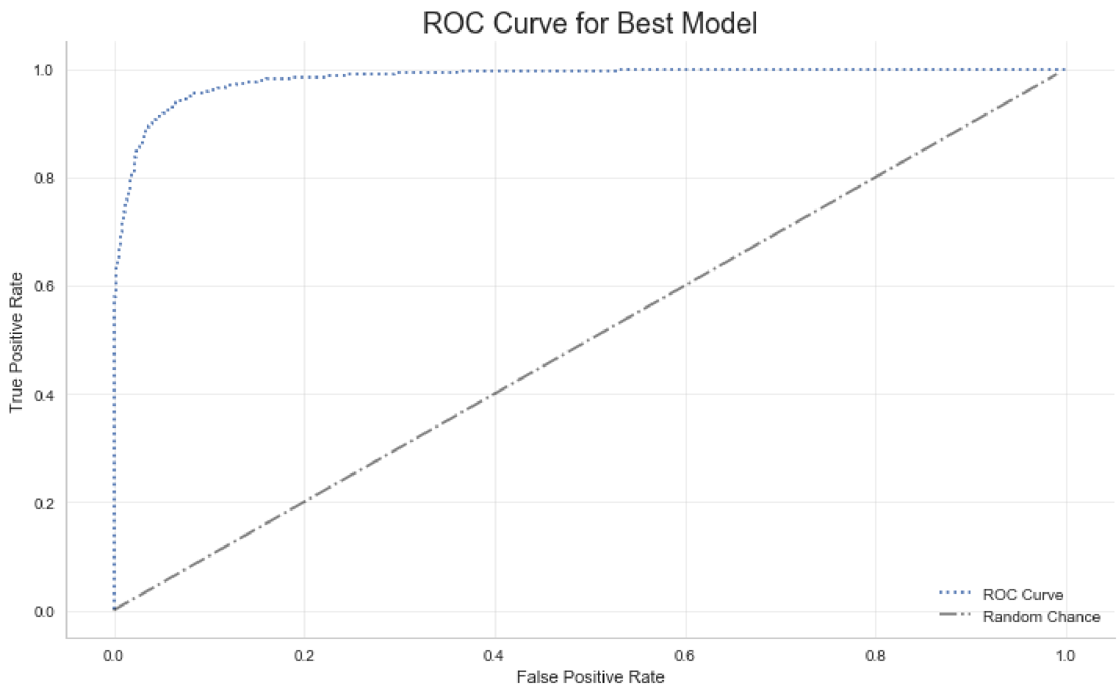
Out[68]:
```
array([0, 0, 0, ..., 1, 1, 1], dtype=int32)
```

In [69]:
```
# Compute ROC curve and Area Under the Curve
fpr, tpr, thresholds = roc_curve(y_test, phat_test)
roc_auc = auc(fpr, tpr)
```

In [70]:
```
print('''
Area Under Curve (AUC) = %.4f
'''%roc_auc)
```

```
Area Under Curve (AUC) = 0.9823
```

In [71]:
```
fig, ax = plt.subplots(figsize=(12, 7))
ax.plot(fpr, tpr, linestyle=':', label='ROC Curve')
ax.plot(np.arange(0, 1, .01), np.arange(0, 1, .01), label='Random Chance',
        linestyle='-.', alpha=.5, color='black')
set_mpl_preferences(ax)
ax.set_title('ROC Curve for Best Model', size=18)
ax.set_xlabel('False Positive Rate')
ax.set_ylabel('True Positive Rate')
plt.show()
```




```
In [74]: from sklearn.metrics import accuracy_score, f1_score

         acc = accuracy_score(y_test, yhat_test)
         f1 = f1_score(y_test, yhat_test)

         print('''
         Accuracy Score = %.4f
         F1 Score = %.4f
         ''' %(acc, f1))
```

```
Accuracy Score = 0.9331
F1 Score = 0.9365
```

The model looks pretty good! High testing accuracy score and F1 score indicate a robust model. Of course, only time will tell if this is actually the case.

0.9 Problem 6 (5 points)

Display the 5 images [worst] misclassified images for each class. Worst is in brackets since certain architectures may only make a binary decision rather than a score. In that case, plot 5 of each.

```
In [78]: test_labels['proba'] = phat_test
         test_labels['yhat'] = [int(y) for y in yhat_test]
```

```
test_labels['y'] = y_test
test_labels.head()
```

Out[78]:

	0	1	proba	yhat	y
0	3.jpg	portrait	0.001469	0	0
1	34.jpg	portrait	0.001025	0	0
2	68.jpg	portrait	0.000299	0	0
3	87.jpg	landscape	0.014885	0	0
4	92.jpg	portrait	0.000164	0	0

In [83]:
```
is_wrong = test_labels['yhat'] != test_labels['y']
is_landscape = test_labels[1] == 'landscape'
is_portrait = test_labels[1] == 'portrait'
```

Out[83]:

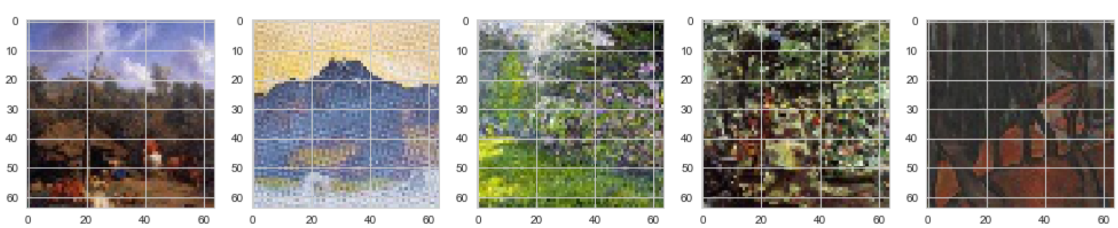
	0	1	proba	yhat	y
4655	65550.jpg	landscape	0.000490	0	1
6840	95988.jpg	landscape	0.002054	0	1
5593	79396.jpg	landscape	0.003342	0	1
4675	66023.jpg	landscape	0.006909	0	1
4060	57302.jpg	landscape	0.008769	0	1

In [90]:
```
misclassified_landscapes = test_labels.loc[is_wrong & is_landscape]
misclassified_landscapes.sort_values('proba', ascending=True, inplace=True)
misclassified_landscapes.head()
```

Out[90]:

	0	1	proba	yhat	y
4655	65550.jpg	landscape	0.000490	0	1
6840	95988.jpg	landscape	0.002054	0	1
5593	79396.jpg	landscape	0.003342	0	1
4675	66023.jpg	landscape	0.006909	0	1
4060	57302.jpg	landscape	0.008769	0	1

In [89]:
```
files = ['data/images64/test/landscape/'+str(f) for f in misclassified_landscapes[0].
fig, axes = plt.subplots(1, 5, figsize=(17, 5))
for i, f in enumerate(files):
    ax = axes[i]
    img = Image.open(f)
    ax.imshow(np.asarray(img))
    img.close()

plt.show()
```

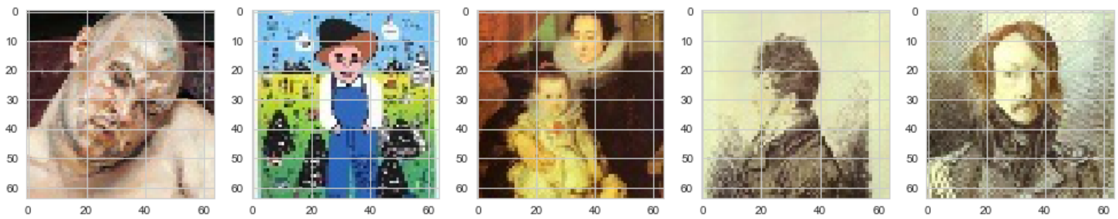


```
In [92]: misclassified_portraits = test_labels.loc[is_wrong & is_portrait]
         misclassified_portraits.sort_values('proba', ascending=False, inplace=True)
         misclassified_portraits.head()
```

Out[92]:

	0	1	proba	yhat	y
431	5650.jpg	portrait	0.996865	1	0
1862	26252.jpg	portrait	0.985408	1	0
1537	21756.jpg	portrait	0.978881	1	0
2159	30115.jpg	portrait	0.976010	1	0
1759	24966.jpg	portrait	0.974267	1	0

```
In [94]: files = ['data/images64/test/portrait/'+str(f) for f in misclassified_portraits[0].hea
         fig, axes = plt.subplots(1, 5, figsize=(17, 5))
         for i, f in enumerate(files):
             ax = axes[i]
             img = Image.open(f)
             ax.imshow(np.asarray(img))
             img.close()

         plt.show()
```



0.10 Problem 7 (2 points)

How many hours did this homework take you? The answer to this question will not affect your grade.

```
In [80]: print('''
         About 12 hours.
         ''')
```

```
About 12 hours.
```

0.11 Last step (3 points)

Save this notebook as LastnameFirstnameHW5.ipynb such as PriceDavid.ipynb. Create a pdf of this notebook named similarly. Submit both the python notebook and the pdf version to the Canvas dropbox. We require both versions.

```
In [81]: print('''
         OK!
         ''')
```

```
OK!
```