Imports that will be used throughout the Notebook.

I have been looking at past assignments and I feel like I am making progress and going towards the right direction. This is what I have so far.

```
import os
import PIL
import numpy as np
import pandas as pd
import matplotlib.pylab as plt
import skimage
import keras
import tensorflow
import shutil
from PIL import Image
from PIL import ImageOps
from skimage import io
from random import shuffle
from keras import layers
from keras import models
```

Versions that are being used.

```
print('Pillow Version:', PIL.__version__)
print('Skimage Version: ', skimage.__version__)
```

```
Pillow Version: 7.0.0
Skimage Version:  0.16.2
```

Connecting to Google Drive account.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
drive.mount("/content/drive",
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", f
```

Gets the data ready for training.

```
!ls "/content/drive/My Drive/460Project/letters/train1"
```

Gets an image and prints the type and shape of the image. Then flips the image and puts it in a new directory.

```
num = 7
#index = 151
while num <= 10:
  im = Image.open('/content/drive/My Drive/460Project/let/z_{}.png'.format(num))
  im_mirror = ImageOps.mirror(im)
  im_mirror.save('/content/drive/My Drive/460Project/letters/train1/z_{}.png'.format(num), quality=95)
  #im.save('/content/drive/My Drive/460Project/letters/train1/{}.png'.format(index), quality=95)
  num = num + 1
  #index = index + 1

print('Done.')
```

```
Done.
```

```
im = Image.open('/content/drive/My Drive/460Project/let/w_3.png')
im_mirror = ImageOps.mirror(im)
im_mirror.save('/content/drive/My Drive/460Project/letters/train1/w_3.png', quality=95)
```

```
TRAIN_DIR = '/content/drive/My Drive/460Project/lettes/train1/'
#TEST_DIR = '??' # whenever we get the rest of the data fixed
IMG_SIZE = 64
#LR = 1e-3

#MODEL_NAME =
```

```
num = 1
while num <= 172:
    img = np.array(Image.open('/content/drive/My Drive/460Project/letters/train1/{}.png'.format(num)))
    print(type(img))
    print(img.shape)
    print(img.size)
    print(num)
    num = num+1
```

Starts training the data.

```
import shutil, sys
# Directory with our training a pictures
train_a_dir = '/content/drive/My Drive/460Project/letters/train/a/'
# Copy a images to train_a_dir
fnames = ['a_{}.png'.format(i) for i in range(1,11)]
for fname in fnames:
    src = '/content/drive/My Drive/460Project/letters/train1/' + fname
    dst = train_a_dir + fname
    shutil.copyfile(src, dst)

# Directory with our training b pictures
train_b_dir = '/content/drive/My Drive/460Project/letters/train/b/'
# Copy a images to train_b_dir
fnames = ['b_{}.png'.format(i) for i in range(1,11)]
for fname in fnames:
    src = '/content/drive/My Drive/460Project/letters/train1/' + fname
    dst = train_b_dir + fname
    shutil.copyfile(src, dst)

# Directory with our training c pictures
train_c_dir = '/content/drive/My Drive/460Project/letters/train/c/'
# Copy a images to train_c_dir
fnames = ['c_{}.png'.format(i) for i in range(1,11)]
for fname in fnames:
    src = '/content/drive/My Drive/460Project/letters/train1/' + fname
    dst = train_c_dir + fname
```

```
    shutil.copyfile(src, dst)

# Directory with our training d pictures
train_d_dir = '/content/drive/My Drive/460Project/letters/train/d/'
# Copy a images to train_d_dir
fnames = ['d_{}.png'.format(i) for i in range(1,11)]
for fname in fnames:
    src = '/content/drive/My Drive/460Project/letters/train1/' + fname
    dst = train_d_dir + fname
    shutil.copyfile(src, dst)

# Directory with our training e pictures
train_e_dir = '/content/drive/My Drive/460Project/letters/train/e/'
# Copy a images to train_e_dir
fnames = ['e_{}.png'.format(i) for i in range(1,11)]
for fname in fnames:
    src = '/content/drive/My Drive/460Project/letters/train1/' + fname
    dst = train_e_dir + fname
    shutil.copyfile(src, dst)

# Directory with our training f pictures
train_f_dir = '/content/drive/My Drive/460Project/letters/train/f/'
# Copy a images to train_f_dir
fnames = ['f_{}.png'.format(i) for i in range(1,11)]
for fname in fnames:
    src = '/content/drive/My Drive/460Project/letters/train1/' + fname
    dst = train_f_dir + fname
    shutil.copyfile(src, dst)

# Directory with our training g pictures
train_g_dir = '/content/drive/My Drive/460Project/letters/train/g/'
# Copy a images to train_g_dir
fnames = ['g_{}.png'.format(i) for i in range(1,11)]
for fname in fnames:
    src = '/content/drive/My Drive/460Project/letters/train1/' + fname
    dst = train_g_dir + fname
    shutil.copyfile(src, dst)

# Directory with our training h pictures
```

```
train_h_dir = '/content/drive/My Drive/460Project/letters/train/h/'
# Copy a images to train_h_dir
fnames = ['h_{}.png'.format(i) for i in range(1,11)]
for fname in fnames:
    src = '/content/drive/My Drive/460Project/letters/train1/' + fname
    dst = train_h_dir + fname
    shutil.copyfile(src, dst)

# Directory with our training i pictures
train_i_dir = '/content/drive/My Drive/460Project/letters/train/i/'
# Copy a images to train_i_dir
fnames = ['i_{}.png'.format(i) for i in range(1,11)]
for fname in fnames:
    src = '/content/drive/My Drive/460Project/letters/train1/' + fname
    dst = train_i_dir + fname
    shutil.copyfile(src, dst)

# Directory with our training j pictures
train_j_dir = '/content/drive/My Drive/460Project/letters/train/j/'
# Copy a images to train_j_dir
fnames = ['j_{}.png'.format(i) for i in range(1,11)]
for fname in fnames:
    src = '/content/drive/My Drive/460Project/letters/train1/' + fname
    dst = train_j_dir + fname
    shutil.copyfile(src, dst)

# Directory with our training k pictures
train_k_dir = '/content/drive/My Drive/460Project/letters/train/k/'
# Copy a images to train_k_dir
fnames = ['k_{}.png'.format(i) for i in range(1,11)]
for fname in fnames:
    src = '/content/drive/My Drive/460Project/letters/train1/' + fname
    dst = train_k_dir + fname
    shutil.copyfile(src, dst)

# Directory with our training l pictures
train_l_dir = '/content/drive/My Drive/460Project/letters/train/l/'
# Copy a images to train_l_dir
fnames = ['l_{}.png'.format(i) for i in range(1,11)]
```

```
for fname in fnames:
    src = '/content/drive/My Drive/460Project/letters/train1/' + fname
    dst = train_l_dir + fname
    shutil.copyfile(src, dst)

# Directory with our training m pictures
train_m_dir = '/content/drive/My Drive/460Project/letters/train/m/'
# Copy a images to train_m_dir
fnames = ['m_{}.png'.format(i) for i in range(1,11)]
for fname in fnames:
    src = '/content/drive/My Drive/460Project/letters/train1/' + fname
    dst = train_m_dir + fname
    shutil.copyfile(src, dst)

# Directory with our training n pictures
train_n_dir = '/content/drive/My Drive/460Project/letters/train/n/'
# Copy a images to train_n_dir
fnames = ['n_{}.png'.format(i) for i in range(1,11)]
for fname in fnames:
    src = '/content/drive/My Drive/460Project/letters/train1/' + fname
    dst = train_n_dir + fname
    shutil.copyfile(src, dst)

# Directory with our training o pictures
train_o_dir = '/content/drive/My Drive/460Project/letters/train/o/'
# Copy a images to train_o_dir
fnames = ['o_{}.png'.format(i) for i in range(1,11)]
for fname in fnames:
    src = '/content/drive/My Drive/460Project/letters/train1/' + fname
    dst = train_o_dir + fname
    shutil.copyfile(src, dst)

# Directory with our training p pictures
train_p_dir = '/content/drive/My Drive/460Project/letters/train/p/'
# Copy a images to train_p_dir
fnames = ['p_{}.png'.format(i) for i in range(1,11)]
for fname in fnames:
    src = '/content/drive/My Drive/460Project/letters/train1/' + fname
    dst = train_p_dir + fname
    shutil.copyfile(src, dst)
```

```
    shutil.copyfile(src, dst)

# Directory with our training q pictures
train_q_dir = '/content/drive/My Drive/460Project/letters/train/q/'
# Copy a images to train_q_dir
fnames = ['q_{}.png'.format(i) for i in range(1,11)]
for fname in fnames:
    src = '/content/drive/My Drive/460Project/letters/train1/' + fname
    dst = train_q_dir + fname
    shutil.copyfile(src, dst)

# Directory with our training r pictures
train_r_dir = '/content/drive/My Drive/460Project/letters/train/r/'
# Copy a images to train_r_dir
fnames = ['r_{}.png'.format(i) for i in range(1,11)]
for fname in fnames:
    src = '/content/drive/My Drive/460Project/letters/train1/' + fname
    dst = train_r_dir + fname
    shutil.copyfile(src, dst)

# Directory with our training s pictures
train_s_dir = '/content/drive/My Drive/460Project/letters/train/s/'
# Copy a images to train_s_dir
fnames = ['s_{}.png'.format(i) for i in range(1,11)]
for fname in fnames:
    src = '/content/drive/My Drive/460Project/letters/train1/' + fname
    dst = train_s_dir + fname
    shutil.copyfile(src, dst)

# Directory with our training t pictures
train_t_dir = '/content/drive/My Drive/460Project/letters/train/t/'
# Copy a images to train_t_dir
fnames = ['t_{}.png'.format(i) for i in range(1,11)]
for fname in fnames:
    src = '/content/drive/My Drive/460Project/letters/train1/' + fname
    dst = train_t_dir + fname
    shutil.copyfile(src, dst)

# Directory with our training u pictures
train_u_dir = '/content/drive/My Drive/460Project/letters/train/u/'
```

```
train_u_dir = '/content/drive/My Drive/460Project/letters/train/u/'
# Copy a images to train_u_dir
fnames = ['u_{}.png'.format(i) for i in range(1,11)]
for fname in fnames:
    src = '/content/drive/My Drive/460Project/letters/train1/' + fname
    dst = train_u_dir + fname
    shutil.copyfile(src, dst)

# Directory with our training v pictures
train_v_dir = '/content/drive/My Drive/460Project/letters/train/v/'
# Copy a images to train_v_dir
fnames = ['v_{}.png'.format(i) for i in range(1,11)]
for fname in fnames:
    src = '/content/drive/My Drive/460Project/letters/train1/' + fname
    dst = train_v_dir + fname
    shutil.copyfile(src, dst)

# Directory with our training w pictures
train_w_dir = '/content/drive/My Drive/460Project/letters/train/w/'
# Copy a images to train_w_dir
fnames = ['w_{}.png'.format(i) for i in range(1,11)]
for fname in fnames:
    src = '/content/drive/My Drive/460Project/letters/train1/' + fname
    dst = train_w_dir + fname
    shutil.copyfile(src, dst)

# Directory with our training x pictures
train_x_dir = '/content/drive/My Drive/460Project/letters/train/x/'
# Copy a images to train_x_dir
fnames = ['x_{}.png'.format(i) for i in range(1,11)]
for fname in fnames:
    src = '/content/drive/My Drive/460Project/letters/train1/' + fname
    dst = train_x_dir + fname
    shutil.copyfile(src, dst)

# Directory with our training y pictures
train_y_dir = '/content/drive/My Drive/460Project/letters/train/y/'
# Copy a images to train_y_dir
fnames = ['y_{}.png'.format(i) for i in range(1,11)]
for fname in fnames:
```

```
for fname in fnames:
    src = '/content/drive/My Drive/460Project/letters/train1/' + fname
    dst = train_y_dir + fname
    shutil.copyfile(src, dst)

# Directory with our training z pictures
train_z_dir = '/content/drive/My Drive/460Project/letters/train/z/'
# Copy a images to train_z_dir
fnames = ['z_{}.png'.format(i) for i in range(1,11)]
for fname in fnames:
    src = '/content/drive/My Drive/460Project/letters/train1/' + fname
    dst = train_z_dir + fname
    shutil.copyfile(src, dst)


import shutil, sys
# Directory with our training cat pictures
train_b_dir = '/content/drive/My Drive/460Project/letters/train/b/'

# Copy a images to train_a_dir
fnames = ['b_{}.png'.format(i) for i in range(1,11)]
for fname in fnames:
    src = '/content/drive/My Drive/460Project/letters/train1/' + fname
    dst = train_b_dir + fname
    shutil.copyfile(src, dst)


from keras import layers
from keras import models

model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
```

```
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 15, 15, 128)       147584
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 7, 7, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 6272)              0
_________________________________________________________________
dense (Dense)                (None, 512)               3211776
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 3,453,121
Trainable params: 3,453,121
Non-trainable params: 0
_________________________________________________________________
```

```
from keras import optimizers

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

```
from keras.preprocessing.image import ImageDataGenerator

train_dir = '/content/drive/My Drive/460Project/letters/train/'
validation_dir = '/content/drive/My Drive/460Project/letters/validate/'

# All images will be rescaled by 1./255
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        # This is the target directory
        train_dir,
        # All images will be resized to 150x150
        target_size=(150, 150),
        batch_size=20,
        # Since we use binary_crossentropy loss, we need binary labels
        class_mode='categorical')

validation_generator = test_datagen.flow_from_directory(
        validation_dir,
        target_size=(150, 150),
        batch_size=20,
        class_mode='categorical')
```

```
Found 261 images belonging to 26 classes.
Found 0 images belonging to 28 classes.
```

```
for data_batch, labels_batch in train_generator:
    print('data batch shape:', data_batch.shape)
    print('labels batch shape:', labels_batch.shape)
    break
```

```
data batch shape: (20, 150, 150, 3)
labels batch shape: (20, 26)
```

```
history = model.fit_generator(
      train_generator,
```

```
        train_generator,
        steps_per_epoch=100,
        epochs=10,
        validation_data=validation_generator,
        validation_steps=50)
```

```
Epoch 1/10
 14/100 [===>.........................] - ETA: 3:40 - loss: 0.1648 - acc: 0.9615WARNING:tensorflow:Your inpu
 14/100 [===>.........................] - 36s 3s/step - loss: 0.1648 - acc: 0.9615
```

```
import matplotlib.pyplot as plt

acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()

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

plt.show()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-79-ce9171dbf527> in <module>()
      2
      3 acc = history.history['acc']
----> 4 val_acc = history.history['val_acc']
      5 loss = history.history['loss']
```

```
datagen = ImageDataGenerator(
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest')


# This is module with image preprocessing utilities
from keras.preprocessing import image

fnames = ['/content/drive/My Drive/460Project/letters/train/a/']

# We pick one image to "augment"
img_path = fnames[9]

# Read the image and resize it
img = image.load_img(img_path, target_size=(150, 150))

# Convert it to a Numpy array with shape (150, 150, 3)
x = image.img_to_array(img)

# Reshape it to (1, 150, 150, 3)
x = x.reshape((1,) + x.shape)

# The .flow() command below generates batches of randomly transformed images.
# It will loop indefinitely, so we need to `break` the loop at some point!
i = 0
for batch in datagen.flow(x, batch_size=1):
    plt.figure(i)
    imgplot = plt.imshow(image.array_to_img(batch[0]))
```

```
    imgplot = plt.imshow(image.array_to_img(batch[0]))
    i += 1
    if i % 4 == 0:
        break

plt.show()
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-84-5fc930ad52fa> in <module>()
      5
      6 # We pick one image to "augment"
----> 7 img_path = fnames[9]
      8
      9 # Read the image and resize it

IndexError: list index out of range
```

SEARCH STACK OVERFLOW

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',
                        input_shape=(150, 150, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))

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

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,)

# Note that the validation data should not be augmented!
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        # This is the target directory
        train_dir,
        # All images will be resized to 150x150
        target_size=(150, 150),
        batch_size=32,
        # Since we use binary_crossentropy loss, we need binary labels
        class_mode='binary')

validation_generator = test_datagen.flow_from_directory(
        validation_dir,
        target_size=(150, 150),
        batch_size=32,
        class_mode='binary')

history = model.fit_generator(
      train_generator,
      steps_per_epoch=100,
      epochs=100,
      validation_data=validation_generator,
      validation_steps=50)


model.save('train1.h5')


acc = history.history['acc']
val_acc = history.history['val_acc']
```

```
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()

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

plt.show()
```