DL_Microscopy_Solution

June 12, 2019

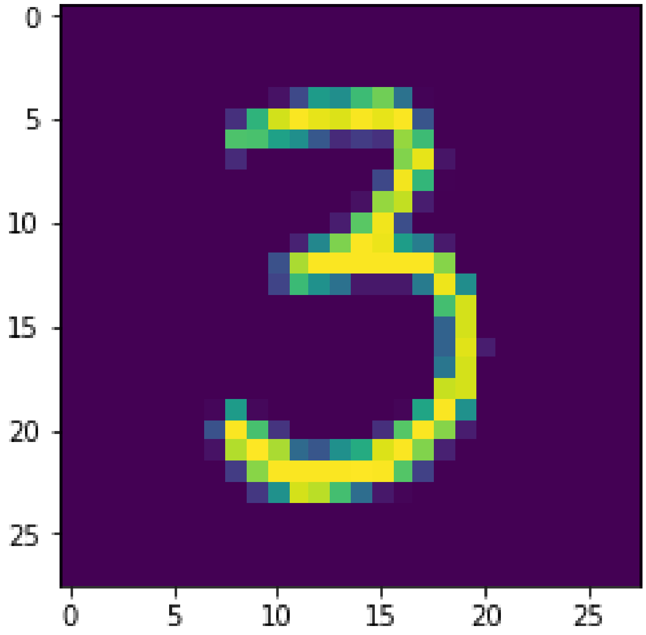
```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        from skimage.transform import resize
```

```
In [2]: # Download the dataset
        from keras.datasets import mnist
        (x_train, y_train), (x_test, y_test) = mnist.load_data()
```

```
Using TensorFlow backend.
```

```
In [3]: # View example digit
        plt.imshow(x_train[50])
```

```
Out[3]: <matplotlib.image.AxesImage at 0x13035b3c8>
```

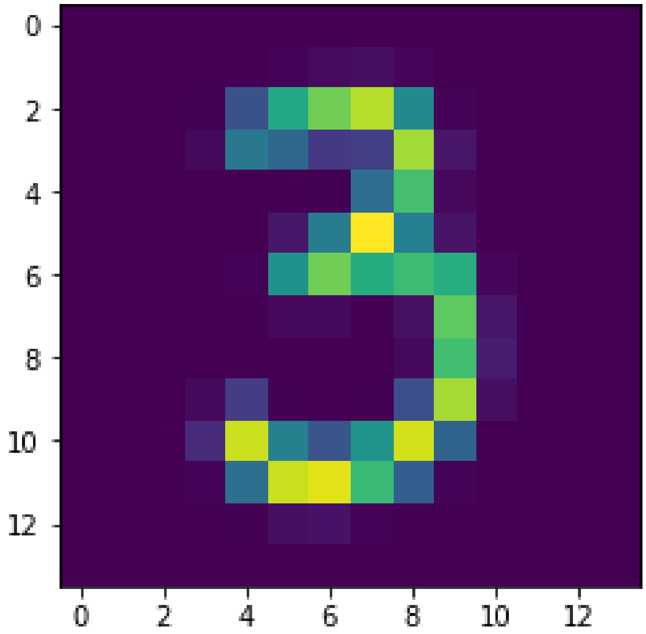


```
In [4]: # Resize the example digit to 14 x 14
        low_res = resize(x_train[50], (14, 14), anti_aliasing=True)
```

```
/Users/alican/.virtualenvs/ml/lib/python3.6/site-packages/skimage/transform/_warps.py:105: Use
  warn("The default mode, 'constant', will be changed to 'reflect' in "
```

```
In [5]: # View low resolution example image
        plt.imshow(low_res)
```

```
Out[5]: <matplotlib.image.AxesImage at 0x1304666d8>
```



```
In [6]: np.max(low_res)
```

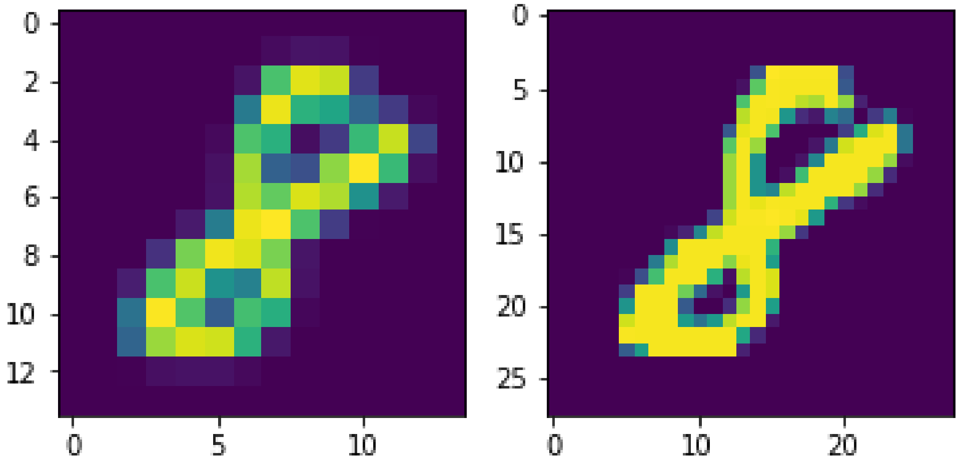
```
Out[6]: 0.8500000000000008
```

```
In [7]: # Resize all training images to 14 x 14
        x_train_lowres = resize(x_train, (x_train.shape[0], 14, 14), anti_aliasing=True)
```

```
In [8]: # Resize all test images to 14 x 14
        x_test_lowres = resize(x_test, (x_test.shape[0], 14, 14), anti_aliasing=True)
```

```
In [9]: # View the dataset
        index = np.random.randint(0,x_train.shape[0])
        f, axarr = plt.subplots(1,2)
        axarr[0].imshow(x_train_lowres[index])
        axarr[1].imshow(x_train[index])
        print(y_train[index])
```

```
8
```



```
In [10]: # Build a neural network
         # That takes low resolution images (14 x 14)
         # Outputs high resolution images (28 x 28)
```

```
In [11]: from keras.models import Sequential
         from keras.layers import Conv2D, Activation, UpSampling2D
```

```
In [12]: model = Sequential()
         model.add(Conv2D(20, (3,3), padding='same', input_shape=(14,14,1)))
         model.add(Activation('relu'))
         model.add(UpSampling2D((2,2)))
         model.add(Conv2D(20, (3,3), padding='same'))
         model.add(Activation('relu'))
         model.add(Conv2D(10, (3,3), padding='same'))
         model.add(Activation('relu'))
         model.add(Conv2D(1, (3,3), padding='same'))
         model.add(Activation('relu'))
         model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 14, 14, 20)        200
_________________________________________________________________
activation_1 (Activation)    (None, 14, 14, 20)        0
_________________________________________________________________
up_sampling2d_1 (UpSampling2 (None, 28, 28, 20)        0
_________________________________________________________________
```

```
conv2d_2 (Conv2D)            (None, 28, 28, 20)        3620      
_________________________________________________________________
activation_2 (Activation)    (None, 28, 28, 20)        0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 28, 28, 10)        1810      
_________________________________________________________________
activation_3 (Activation)    (None, 28, 28, 10)        0         
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 28, 28, 1)         91        
_________________________________________________________________
activation_4 (Activation)    (None, 28, 28, 1)         0         
=================================================================
Total params: 5,721
Trainable params: 5,721
Non-trainable params: 0
_________________________________________________________________
```

```
In [13]: model.compile(optimizer='adam', loss='mse')
```

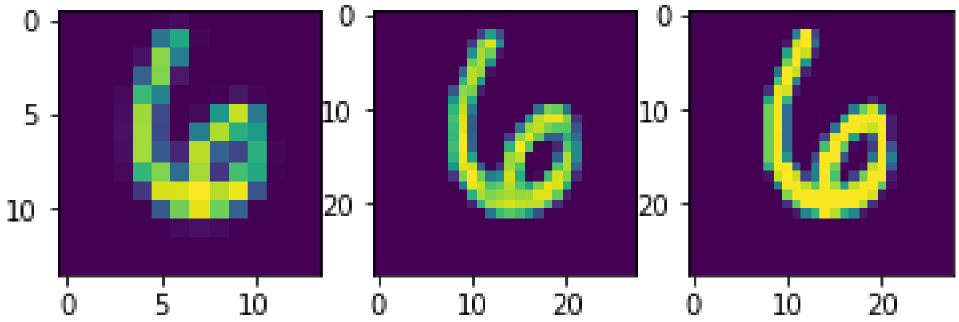
```
In [14]: x_train_lowres = x_train_lowres.reshape(-1,14,14,1)
         x_train = x_train.reshape(-1,28,28,1)

         x_test_lowres = x_test_lowres.reshape(-1,14,14,1)
         x_test = x_test.reshape(-1,28,28,1)
```

```
In [15]: H = model.fit(x_train_lowres, x_train, batch_size=32, epochs=2, validation_data=(x_tes
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/2
60000/60000 [==============================] - 183s 3ms/step - loss: 641.5780 - val_loss: 201.9
Epoch 2/2
60000/60000 [==============================] - 186s 3ms/step - loss: 192.9078 - val_loss: 178.
```

```
In [16]: # Input, Prediction, Label
         index = np.random.randint(0,x_test.shape[0])
         f, axarr = plt.subplots(1,3)
         axarr[0].imshow(x_test_lowres[index].reshape(14,14))
         axarr[1].imshow(model.predict(x_test_lowres[index:index+1]).reshape(28,28))
         axarr[2].imshow(x_test[index].reshape(28,28))
         print(y_test[index])
```

```
6
```



```
In [ ]:
```