Kaggle_niquette

August 3, 2020

0.0.1 Description

The goal of this problem was to use a CNN to classify each image as either containing metastatic tissue or not. Our data comes in the form of .tif images while the training files show image_id's with their associated labels (1: DOES contain metatstatic tissue, 0: Does NOT contain metastatic tissue). Each image is 96x96 pixels with RGB components making it (96,96,3)

[66]:
```
import pandas as pd
from keras import Sequential
from keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import RMSprop
import matplotlib.pyplot as plt
```

[33]:
```
# Extract labels from csv training data
train_labels = pd.read_csv('histopathologic-cancer-detection/train_labels.csv')
train_labels
```

[33]:

	id	label
0	f38a6374c348f90b587e046aac6079959adf3835	0
1	c18f2d887b7ae4f6742ee445113fa1aef383ed77	1
2	755db6279dae599ebb4d39a9123cce439965282d	0
3	bc3f0c64fb968ff4a8bd33af6971ecae77c75e08	0
4	068aba587a4950175d04c680d38943fd488d6a9d	0
…	…	…
220020	53e9aa9d46e720bf3c6a7528d1fca3ba6e2e49f6	0
220021	d4b854fe38b07fe2831ad73892b3cec877689576	1
220022	3d046cead1a2a5cbe00b2b4847cfb7ba7cf5fe75	0
220023	f129691c13433f66e1e0671ff1fe80944816f5a2	0
220024	a81f84895ddcd522302ddf34be02eb1b3e5af1cb	1

[220025 rows x 2 columns]

0.0.2 Preliminary Analysis

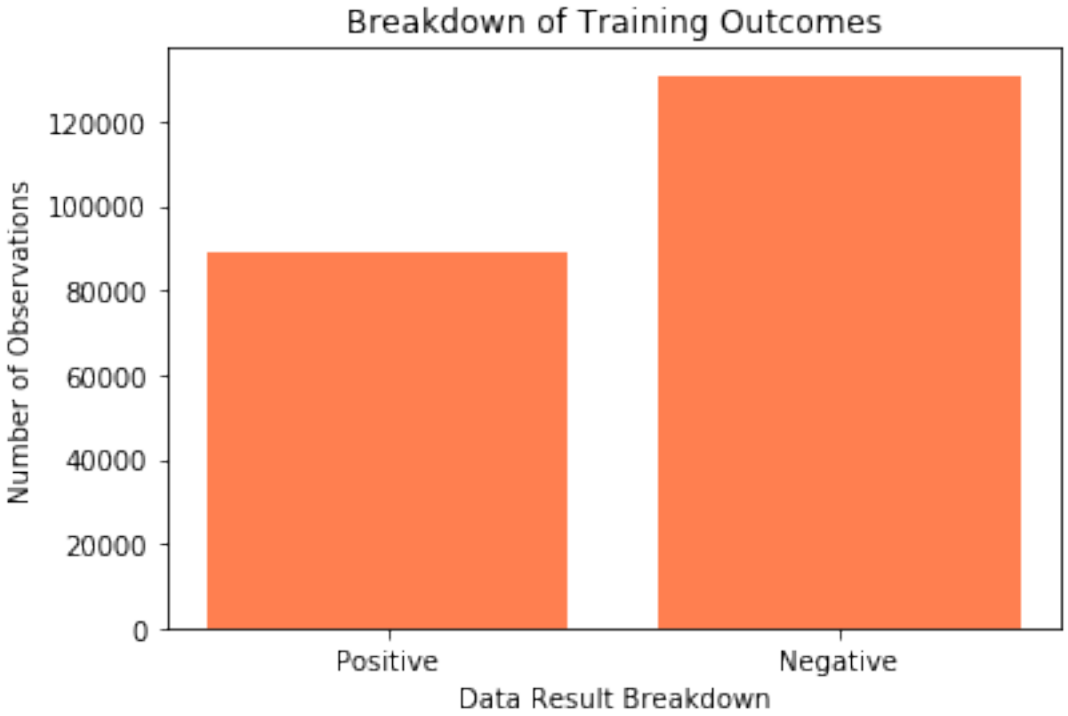
First I would like to look at the breakdown of training data to see if we have a roughly balanced training set or if there is a skew in the outcomes of the training samples.

[34]:
```
posCount = train_labels[train_labels['label'] == 1].shape[0]
negCount = train_labels[train_labels['label'] == 0].shape[0]


pos = train_labels[train_labels['label'] == 1]
neg = train_labels[train_labels['label'] == 0]


x = ['Positive', 'Negative']
x_pos = [0,1]

plt.bar(x_pos, [posCount,negCount], color='coral')
plt.xlabel("Data Result Breakdown")
plt.ylabel("Number of Observations")
plt.title("Breakdown of Training Outcomes")
plt.xticks(x_pos, x)
plt.show()
```



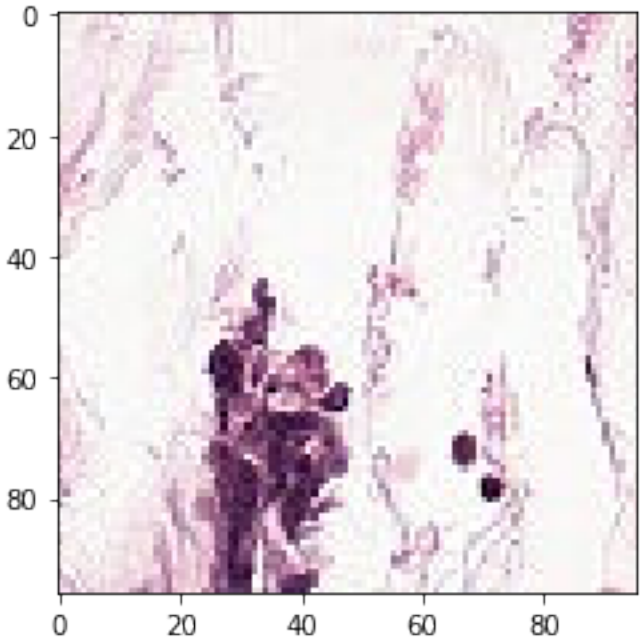
The bar chart above shows that we do have more negative training data points than positive so this is something to keep in mind as we review our results.

0.1 Image Inspection

Next I would like to see the images that we will be reading in. I will figure out the size of the images as well as get a feel for what they look like and the features we will be trying to extract.

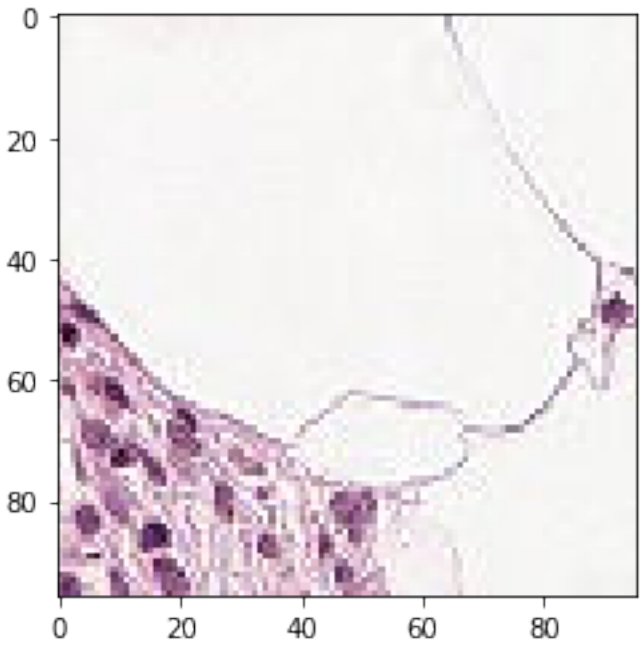
[35]:
```
img = plt.imread('histopathologic-cancer-detection/train/'+pos.iloc[0]['id']+'.
↪tif')
plt.imshow(img)
```

[35]:
```
<matplotlib.image.AxesImage at 0x1536877b8>
```



[84]:
```
img = plt.imread('histopathologic-cancer-detection/train/'+neg.iloc[0]['id']+'.
↪tif')
plt.imshow(img)
print(img.shape)
```

```
(96, 96, 3)
```



1 Model Building Explanation

Now I come to the part of the exercise where I will build my model. Being that we are in the convolutional neural network unit, it seemed only right to use a CNN for this task. I becan with the pattern of $[Convolution - Convolution - MaxPool]_n$ as stated in the lecture slides that is recommended for this type of image analysis. We need the convolution layers to be able to extract the notable features of the image, and thus we repeat this pattern 4 times.

I chose padding to be 'same' because that means that there will be padding added to the edge of the picture so that the size of the image does not shrink after we apply the filter to it. As far as the kernel size, I somewhat arbitrarily chose 3 but this was also based on an example that we did in the lecture slides that suggested a 3x3 kernel would work. I had read that the number of filters should be larger on each subequent layer so I doubled the amount of filters for each set of convolutions.

I use the Dropout as a way to reduce overfitting. We take some random weights and set them to be zero so that we end up regularizing out neural net.

Per the suggested architecture from the lecture slides, I used ReLU as the activation function for the hidden layers and sigmoid as the activation function for the output layer because the nature of our output is binary. I also chose to use RMSprop as the optimizer.

Finally, after our convolution layers, I use Flatten() to flatten the output into one long vector which is then input into a standard neural net which uses 2 dense hidden layers.

[87]:
```
# Inspiration was taken from the Kaggle notebook titled, "CNN with Keras SGD"␣
↪as there was relevant information on model building and loading in data
```

```
config = [Conv2D(filters=16, kernel_size=3, padding='same', activation='relu',␣
 ↪input_shape = (96,96,3)),
          Conv2D(filters=16, kernel_size=3, padding='same', activation='relu'),
          Dropout(0.3),
          MaxPooling2D(pool_size= 3),

          Conv2D(filters=32, kernel_size=3, padding='same', activation='relu'),
          Conv2D(filters=32, kernel_size=3, padding='same', activation='relu'),
          Dropout(0.3),
          MaxPooling2D(pool_size= 3),

          Conv2D(filters=64, kernel_size=3, padding='same', activation='relu'),
          Conv2D(filters=64, kernel_size=3, padding='same', activation='relu'),
          Dropout(0.3),
          MaxPooling2D(pool_size= 3),

          Conv2D(filters=128, kernel_size=3, padding='same', activation='relu'),
          Conv2D(filters=128, kernel_size=3, padding='same', activation='relu'),
          Dropout(0.3),

          Flatten(),
          Dense(128, activation='relu'),
          Dropout(0.3),
          Dense(1, activation='sigmoid')]

model = Sequential(config)
opt = RMSprop(learning_rate=0.001)
model.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy'])
print(model.summary())
```

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_48 (Conv2D)           (None, 96, 96, 16)        448
_________________________________________________________________
conv2d_49 (Conv2D)           (None, 96, 96, 16)        2320
_________________________________________________________________
dropout_25 (Dropout)         (None, 96, 96, 16)        0
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 32, 32, 16)        0
_________________________________________________________________
conv2d_50 (Conv2D)           (None, 32, 32, 32)        4640
_________________________________________________________________
conv2d_51 (Conv2D)           (None, 32, 32, 32)        9248
_________________________________________________________________
dropout_26 (Dropout)         (None, 32, 32, 32)        0
```

```
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 10, 10, 32)        0
_________________________________________________________________
conv2d_52 (Conv2D)           (None, 10, 10, 64)        18496
_________________________________________________________________
conv2d_53 (Conv2D)           (None, 10, 10, 64)        36928
_________________________________________________________________
dropout_27 (Dropout)         (None, 10, 10, 64)        0
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 3, 3, 64)          0
_________________________________________________________________
conv2d_54 (Conv2D)           (None, 3, 3, 128)         73856
_________________________________________________________________
conv2d_55 (Conv2D)           (None, 3, 3, 128)         147584
_________________________________________________________________
dropout_28 (Dropout)         (None, 3, 3, 128)         0
_________________________________________________________________
flatten_5 (Flatten)          (None, 1152)              0
_________________________________________________________________
dense_10 (Dense)             (None, 128)               147584
_________________________________________________________________
dropout_29 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 129
=================================================================
Total params: 441,233
Trainable params: 441,233
Non-trainable params: 0
_________________________________________________________________
None
```

1.1 Loading in Data

[27]:
```
train_x = []
print('Num of training images: ',train_labels.shape[0])
for idx in range(train_labels.shape[0]):
    img = plt.imread('histopathologic-cancer-detection/train/'+train_labels.
 ↪iloc[idx]['id']+'.tif')
    train_x.append(img)
train_subset = train_x[:100000]
label_subset = train_labels[:100000]
```

```
Num of training images:  220025
```

1.2 Training CNN on Training Images

[39]:
```
train_subset = np.array(train_subset)
print(len(train_labels['label']))
model.fit(x=train_subset, y=label_subset['label'], epochs=10, batch_size=64,␣
 ↪verbose=1)
```

```
220025
Epoch 1/10
1563/1563 [==============================] - 547s 350ms/step - loss: 0.4871 -
accuracy: 0.7797
Epoch 2/10
1563/1563 [==============================] - 540s 346ms/step - loss: 0.4808 -
accuracy: 0.7914
Epoch 3/10
1563/1563 [==============================] - 587s 376ms/step - loss: 0.4637 -
accuracy: 0.7996
Epoch 4/10
1563/1563 [==============================] - 554s 355ms/step - loss: 0.4539 -
accuracy: 0.8048
Epoch 5/10
1563/1563 [==============================] - 564s 361ms/step - loss: 0.4871 -
accuracy: 0.8103
Epoch 6/10
1563/1563 [==============================] - 579s 370ms/step - loss: 0.5381 -
accuracy: 0.8121
Epoch 7/10
1563/1563 [==============================] - 550s 352ms/step - loss: 0.9122 -
accuracy: 0.8206
Epoch 8/10
1563/1563 [==============================] - 552s 353ms/step - loss: 0.7474 -
accuracy: 0.8182
Epoch 9/10
1563/1563 [==============================] - 557s 356ms/step - loss: 1.7956 -
accuracy: 0.7856
Epoch 10/10
1563/1563 [==============================] - 563s 360ms/step - loss: 0.6741 -
accuracy: 0.7622
```

[39]:
```
<tensorflow.python.keras.callbacks.History at 0x14e8cbf60>
```

1.3 Loading in Testing Data

[71]:
```
from glob import glob
test_ims = []
submission = pd.DataFrame()
```

```
2      19709bec800f372d0b1d085da6933dd3ef108846  0.537493
3      7a34fc34523063f13f0617f7518a0330f6187bd3  0.541234
4      93be720ca2b95fe2126cf2e1ed752bd759e9b0ed  0.530335
…                                              …       …
57453  2581931c6ef068f105a872f2c5500275fc678242  0.260064
57454  11b250a664d09ab59fd2afbdb2f8d786763b185d  0.553156
57455  18a6030935ec1ef1ce486ec51bc95abb4008fbf1  0.503618
57456  f541404e501e23a0188c852eb37eac94053cfdc0  0.260064
57457  3cb6f5e2db8ad046c946b581fa12d20df5ce2927  0.546968

[57458 rows x 2 columns]
```

[ ]:

```
  [206 133 202]
  [180 114 178]
  …
  [192 112 181]
  [190 120 180]
  [208 143 199]]

 [[194 123 191]
  [197 128 195]
  [162 102 164]
  …
  [174  87 158]
  [198 118 181]
  [200 125 184]]]


[[[222 188 225]
  [225 195 229]
  [226 201 231]
  …
  [231 228 235]
  [234 231 240]
  [234 231 240]]

 [[232 178 227]
  [219 167 213]
  [214 167 211]
  …
  [238 233 240]
  [233 225 236]
  [227 219 230]]

 [[252 180 243]
  [237 166 226]
  [241 172 229]
  …
  [237 221 234]
  [235 219 232]
  [231 213 227]]

 …

 [[227 143 203]
  [216 130 193]
  [210 121 189]
  …
  [207 164 244]
  [103  68 152]
```

```
  [ 95  65 153]]

 [[189 122 176]
  [219 148 206]
  [228 149 214]
  …
  [176 130 218]
  [ 96  58 143]
  [ 96  67 151]]

 [[242 188 240]
  [248 187 244]
  [240 168 232]
  …
  [166 117 208]
  [ 74  34 120]
  [ 75  43 126]]]


[[[243 179 179]
  [211 146 154]
  [254 191 208]
  …
  [138  92 129]
  [126  76 113]
  [118  66 104]]

 [[227 164 175]
  [216 153 170]
  [212 150 173]
  …
  [149 103 140]
  [174 128 164]
  [230 184 220]]

 [[192 128 155]
  [180 117 146]
  [215 154 185]
  …
  [209 165 200]
  [232 192 226]
  [210 175 208]]

 …

 [[120  65  97]
  [126  69 101]
  [169 106 137]
```

```
   …
   [145  91 123]
   [ 80  34  70]
   [124  83 125]]

  [[113  58 100]
   [118  60 100]
   [154  90 127]
   …
   [236 180 209]
   [132  82 117]
   [149 107 145]]

  [[201 147 197]
   [171 112 160]
   [160  92 139]
   …
   [247 187 215]
   [213 164 194]
   [115  74 108]]]


…


 [[[126  87 108]
   [145 107 132]
   [196 157 188]
   …
   [ 80  47  78]
   [122  85 116]
   [217 180 211]]

  [[164 126 147]
   [129  91 114]
   [185 148 179]
   …
   [ 90  50  85]
   [156 114 150]
   [196 154 190]]

  [[228 191 209]
   [228 193 215]
   [215 179 207]
   …
   [110  67 110]
   [208 163 206]
   [136  91 132]]
```

```
 …

 [[ 86  53  84]
  [ 70  33  67]
  [ 74  34  69]
  …
  [ 89  45  98]
  [127  81 128]
  [ 96  48  90]]

 [[ 82  46  94]
  [116  78 127]
  [123  84 131]
  …
  [ 97  50 105]
  [108  60 112]
  [125  73 121]]

 [[ 73  37  99]
  [ 98  60 121]
  [108  68 128]
  …
  [111  62 117]
  [101  51 102]
  [ 99  45  94]]]


[[[101  51 136]
  [129  79 164]
  [143  91 176]
  …
  [227 213 228]
  [228 221 229]
  [240 235 239]]

 [[129  76 166]
  [108  55 145]
  [161 107 195]
  …
  [218 201 217]
  [238 228 237]
  [244 239 243]]

 [[102  43 137]
  [120  63 157]
  [155  98 192]
  …
```

```
  [220 203 219]
  [250 238 248]
  [244 238 242]]

 …

 [[223 217 221]
  [240 230 238]
  [241 227 240]
  …
  [229 209 234]
  [214 201 219]
  [237 230 238]]

 [[255 248 250]
  [255 249 255]
  [255 235 251]
  …
  [236 218 240]
  [223 212 228]
  [229 227 232]]

 [[255 242 246]
  [130 107 117]
  [165 136 154]
  …
  [246 231 250]
  [229 223 235]
  [219 220 222]]]


[[[174 118 189]
  [175 122 190]
  [175 124 190]
  …
  [125  78 170]
  [140  91 180]
  [121  68 158]]

 [[149  93 166]
  [188 135 205]
  [181 132 198]
  …
  [ 73  37 125]
  [ 53  10  99]
  [119  73 163]]

 [[137  80 159]
```

```
 [117  61 136]
 [159 108 177]
 …
 [ 67  41 130]
 [ 70  37 126]
 [ 84  45 134]]

…

[[ 76  26 123]
 [120  70 169]
 [125  77 176]
 …
 [ 62  19 109]
 [ 84  40 129]
 [144  98 186]]

[[118  43 136]
 [143  72 166]
 [126  61 155]
 …
 [ 63  33 121]
 [ 57  27 115]
 [118  88 176]]

[[206 111 203]
 [178  89 181]
 [205 124 216]
 …
 [ 83  60 148]
 [ 65  46 135]
 [137 118 207]]]]
```

1.4 Predicting on Testing Images

[75]:
```
print(len(test_ims))
preds = []
for idx in range(len(test_ims)):

    if idx % 1000 == 0:
        print(idx)

    preds.append( model.predict(np.expand_dims(test_ims[idx]/255.
↪0,axis=0))[0][0] )
```

```
57458
0
1000
```

2000
3000
4000
5000
6000
7000
8000
9000
10000
11000
12000
13000
14000
15000
16000
17000
18000
19000
20000
21000
22000
23000
24000
25000
26000
27000
28000
29000
30000
31000
32000
33000
34000
35000
36000
37000
38000
39000
40000
41000
42000
43000
44000
45000
46000
47000
48000
49000

```
50000
51000
52000
53000
54000
55000
56000
57000
```

[85]:
```
submission['label'] = preds
submission
```

[85]:
```
                                                    path  \
0      histopathologic-cancer-detection/test/fd0a060e…
1      histopathologic-cancer-detection/test/1f9ee06f…
2      histopathologic-cancer-detection/test/19709bec…
3      histopathologic-cancer-detection/test/7a34fc34…
4      histopathologic-cancer-detection/test/93be720c…
…                                                    …
57453  histopathologic-cancer-detection/test/2581931c…
57454  histopathologic-cancer-detection/test/11b250a6…
57455  histopathologic-cancer-detection/test/18a60309…
57456  histopathologic-cancer-detection/test/f541404e…
57457  histopathologic-cancer-detection/test/3cb6f5e2…

                                          id    labels     label
0      fd0a060ef9c30c9a83f6b4bfb568db74b099154d  0.553017  0.553017
1      1f9ee06f06d329eb7902a2e03ab3835dd0484581  0.517382  0.517382
2      19709bec800f372d0b1d085da6933dd3ef108846  0.537493  0.537493
3      7a34fc34523063f13f0617f7518a0330f6187bd3  0.541234  0.541234
4      93be720ca2b95fe2126cf2e1ed752bd759e9b0ed  0.530335  0.530335
…                                           …         …         …
57453  2581931c6ef068f105a872f2c5500275fc678242  0.260064  0.260064
57454  11b250a664d09ab59fd2afbdb2f8d786763b185d  0.553156  0.553156
57455  18a6030935ec1ef1ce486ec51bc95abb4008fbf1  0.503618  0.503618
57456  f541404e501e23a0188c852eb37eac94053cfdc0  0.260064  0.260064
57457  3cb6f5e2db8ad046c946b581fa12d20df5ce2927  0.546968  0.546968

[57458 rows x 4 columns]
```

[88]:
```
actual_submission = submission[['id', 'label']]
actual_submission.to_csv('submission_niquette.csv', index=False, header=True)
```

[89]:
```
print(actual_submission)
```

```
                                          id     label
0      fd0a060ef9c30c9a83f6b4bfb568db74b099154d  0.553017
1      1f9ee06f06d329eb7902a2e03ab3835dd0484581  0.517382
```

```
2      19709bec800f372d0b1d085da6933dd3ef108846  0.537493
3      7a34fc34523063f13f0617f7518a0330f6187bd3  0.541234
4      93be720ca2b95fe2126cf2e1ed752bd759e9b0ed  0.530335
…                                              …       …
57453  2581931c6ef068f105a872f2c5500275fc678242  0.260064
57454  11b250a664d09ab59fd2afbdb2f8d786763b185d  0.553156
57455  18a6030935ec1ef1ce486ec51bc95abb4008fbf1  0.503618
57456  f541404e501e23a0188c852eb37eac94053cfdc0  0.260064
57457  3cb6f5e2db8ad046c946b581fa12d20df5ce2927  0.546968

[57458 rows x 2 columns]
```

[ ]: