Reference Laboratory Project - Section 11: Classification of Blood Cells using Convoluted Neural Networks

" *A tutorial on the extensive processing of labelled image files and subsequent usage with Convoluted Neural Networks to create a workable prediction model that identifies if human lymphocytes are deemed 'malignant' or 'normal'* "

Ryan Breen

October 5th, 2022

Part One: The goal of this specific project is to simply evaluate the CNN model performance when classifying lympocytes solely on colorization.

Disclaimer: No real patient data used, only public repository for hematology training. This report is solely to test the power of neural network learning algorithms and in no serves as an form of clinical investigation.

Background

Cell classification is the process of identifying specific blood cells in a patient's blood streams typically performed using light microscopy and a blood film (a stained artifact of a patient's blood). This project is the initial proposal and modelling of such white blood cells to determine the malignancy of a specific blood cell called a lymphocyte. Malignant lymphocytes are virtually associated with cancerous states in patients where normal lymphocytes typically constitute anywhere from 20 - 80% of the white blood cell population in human beings.

Furthermore, blood test from a commonly ordered laboratory panel (a set of tests) called a complete blood count has been integrated in the model to see if the presence of such values create a more accurate or predictable model. Note, that a spectrum of abnormality in terms of cellular appearance exists for white blood cells where cells noted as 'reactive' are non-malignant, but appear as an intermediary form between malignant and normal cells. Reactive cells are typically cells just 'doing their job' anthromorphologically speaking - these cells typically respond ot viruses in the body.

Below the the difference between 'malignant', 'reactive', and 'normal' lymphocytes are depicted. Note, a general trend exists where malignant lymphocytes are more *immature* cells that are precursors formed in the bone marrow that *pour* out during malignancy due to the increase in the number of malignant cells in the body. The malignant cells, as well as immature cells, are typically larger, deeper in the blue/violet spectrum, and have a more coarse appearance as described by a more pourous nucleus (the darker central button containing the DNA). Normal cells are smaller, lighter in the blue spectrum, but have a marker finer darker nuclues.

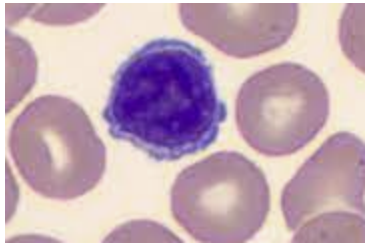
For the purposes of this study only appearance in terms of color intesity and and color hue will be used as distinguising features as opposed to using cell size since cell diameters have not been provided. The colorization of cells is detecting using numerical bits representing a color hue in the Red-Green-Blue(RGB) schematic commonly used in most computer systems to produce colors using 8 bits of information for each red,green, and blue

combination. Overall, this produces 2^8 or 256 possibilities for each of the red,green, or blue color choices and using independent assortment a total of 256 x 256 x 256 = 16777216 total possibilities. Note, that each bit within each color acts as a sole estimator within neural network where the neural network creates weights using the *Convoluted Neural Network(CNN)*.

The CNN modelling algorithm utilizes.....

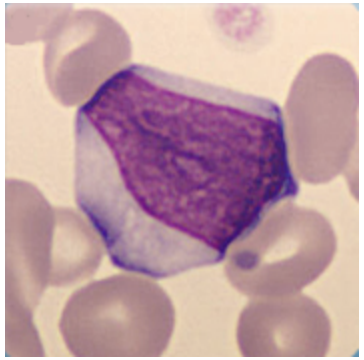
Normal Lymphocyte (note appearance may be larger)

The lymphocyte is the 'purplish-blue' cell with the darker nucleus region of the cell in the center and what is referred to as the cytoplasm on the outer range. Normal lymphocytes may have much larger light blue cytoplasm that are commonly referred to as 'large lymphocytes' that are completely 'normal' pathogenically speaking.



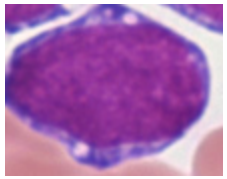
Reactive Lymphocyte (appears in infections)

This particular cell type serves to test the robustness of the model by exhibiting characterists of both normal and malignant cell types. This is the initial phase of modelling using CNN to classify cell types. It is expected that subsequent modelling and possible data exploration could be necessary to explore effects of such intermediary cells. Again, red blood cells found in the background of the picture, normally found in blood films, may also create diminishing power of the model, but may be subsequently investigated in the "post-hoc" investigation by performing image processing such as diminishing values in the "red channel" to disipate the red cells from the background. However, staying a firm believer in rducing the number of variables in a study, the initial model will simply contain cell images "in-situ".



Malignant Lymphocyte (Blasts) (present in cancers)

Malignant lymphocytes are either morphologically altered normal lymphocytes that have changed due to a pathogenic environment (presence of molecules that prompt cellular change in the face of malignancy) or immature lymphocytes that are precursor cells to the normal lymphocyte. More common, the precursor cells depcited by a chronological lineage of cells, contains morphologically/visual changes that somewhat organize as one examines the cells further back within the lineage. As cells become more immature, the colorization of the cell tends to be darker in nature (dark purple and blue hues), larger (to be measured in future studies), and contains more abnormal features within the cells themselves. Most commonly, the features within immature cells is a very large nucleus that is described as "coarge" in appearance - almost with a grind glass cartoonish like sharpness. Furthermore, as seen with the cell below, abnormal additive features such as the bubble like inclusions seen below called vacuoles are common in malignant cells.



Purpose

Classifying malignant cancer cells is typically performed by screen such cells under a microscope with a special staining of a blood sample from the patient by laboratory professionals. The process works well when performed by trained experienced individuals, but like all human endevaors is open to the possibility of human error. Another constraint on medical laboratory screening of cancer cells is the cost factor alongside the laborious process to create such slides. Reducing the blood film to a simple .png image eliminates materials and time needed to created such slides. Furthermore, digital blood films are able to last indefinitely and may even be sent electronic much more rapidly for further consultation.

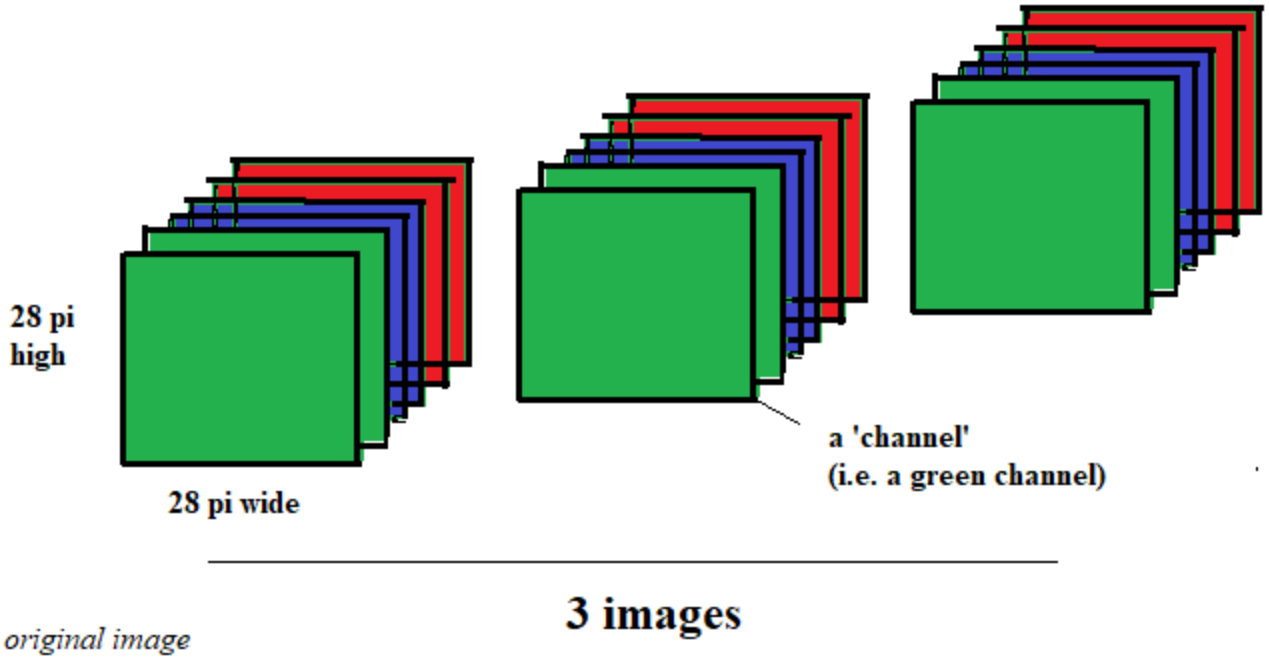
Obtaining and Processing Image Files

Major Goal Convert .png imags to NumPy arrays with shape (width,height,channel)

Below, highlights how each NumPy array should be structured for use with Keras modelling. For channel last conventional arrays, the image attributes are listed in the following order:

As an example, a 28 x 28 RBG format png is depicted as an array below:

- image*array* = (num of samples, height(pixels), width(pixels), color channel) _
- num_of_samples: *The number of image files within the array*
- height:*The height of image in pixels that is 28 in* `NumPy`
- width:*The width of image in pixels that is 28 in* `NumPy`
- channel: *The method or schematic to represent color in this case the RGB system explained earlier*



Libraries and Precursors

In [ ]:
```
#conda install OpenCV
```

In [110…
```
### DELETE ME AFTER ###
import os
import numpy as np
import pandas as pd
from PIL import Image
#import cv2
from matplotlib import pyplot as plt
from keras.preprocessing.image import load_img
from keras.preprocessing.image import save_img
from keras.preprocessing.image import img_to_array
bold = '\033[1m'
end = '\033[0m'
```

Create File Directory

In [70]:
```
import os
neg_file = os.getcwd() + '\\neg\\' # directory forneg images
pos_file = os.getcwd() + '\\pos\\' # directory for pos images
```

```
# imagine we only want to load PNG files (or JPEG or whatever...)
EXTENSION = '.png'
neg_files = os.listdir(neg_file)
neg_paths = [neg_file+file for file in neg_files]
pos_files = os.listdir(pos_file)
pos_paths = [pos_file+file for file in pos_files]
```

Create the Arrays using Pillow

Divide the 255 values to standardize by max value 255

In [71]:
```
neg_images = [Image.open(path).convert('RGB') for path in neg_paths]
neg_arrays = [np.array(image)/255 for image in neg_images]
pos_images = [Image.open(path).convert('RGB') for path in pos_paths]
pos_arrays = [np.array(image)/255 for image in pos_images]
```

Explore Data Shape

In [72]:
```
neg_shapes = [array.shape for array in neg_arrays[:5]]
pos_shapes = [array.shape for array in pos_arrays[:5]]
print('First 5 shapes for neg images:', neg_shapes[:5])
print('First 5 shapes for pos images:', pos_shapes[:5])
```

```
First 5 shapes for neg images: [(177, 177, 3), (160, 224, 3), (115, 113, 3), (115, 113, 3), (115, 113, 3)]
First 5 shapes for pos images: [(86, 109, 3), (92, 108, 3), (99, 91, 3), (99, 91, 3), (99, 91, 3)]
```

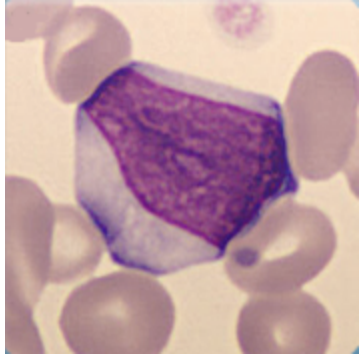
As we can see we have imags of different size widths and heights - this will be a problem for the Keras that requires a square input. Let's crop the images before exploring some more.

Resizing the Images with cv2 Library

Note: ther image processing will be tried post-hoc after initial modelling

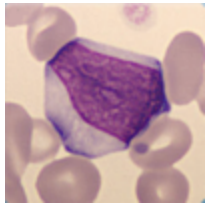
In [73]:
```
first_neg = neg_images[0]
print(bold+"Original Image"+end)
display(first_neg)
```

Original Image

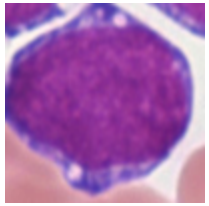


In [74]:

```
(width, height) = (100,100)
neg_cropped = [image.resize((width, height)) for image in neg_images]
neg_arrays = [np.array(cropped) for cropped in neg_cropped]
pos_cropped = [image.resize((width, height)) for image in pos_images]
pos_arrays = [np.array(cropped) for cropped in pos_cropped]
display(neg_cropped[0])
print(neg_arrays[0].shape)
display(pos_cropped[0])
print(pos_arrays[0].shape)
```



```
(100, 100, 3)
```



```
(100, 100, 3)
```

Create Labels for Negative (normal) and Positive (malignant):

Labels will be alternated to match instances

In [98]:

```
neg_labels = [0]*224
pos_labels = [1]*224
labels=[]
for n,p in zip (neg_labels, pos_labels):
    labels.append(n)
    labels.append(p)
```

Train/Test Split

First, negative and positive instances are evenly distburshed and then 75% of instances with labels are reserved for the training data.

In [92]:
```
data = []
for n,p in zip(neg_arrays,pos_arrays):
    data.append(n)
    data.append(p)
print('75% of the data is :',len(data)*.75)
X_train = np.array(data[:336])
X_test = np.array(data[336:])
print('Length of entire data set: ',len(X_train)+len(X_test))
print(X_train.shape)
print(X_test.shape)
```

```
75% of the data is : 336.0
Length of entire data set:  448
(336, 100, 100, 3)
(112, 100, 100, 3)
```

In [99]:
```
Y_train = np.array(labels[:336])
Y_train = Y_train.reshape(336,1)
Y_test = np.array(labels[336:])
Y_test = Y_test.reshape(112,1)
print('Length of labels: ',len(Y_train)+len(Y_test))
print(Y_train.shape)
print(Y_test.shape)
```

```
Length of labels:  448
(336, 1)
(112, 1)
```

Creating Model

Import Necessary Libraries

In [87]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
```

Create the Model

Here a convolutional-pooling filtering is used that scans 3x3 submatrices within the feature map and chooses only the max value to be represented in a much smaller 2x2 matrix per each 3x3 matrix that was scanned. The aforementioned filtering drastically reduces computational constraints and even

though this data set is very small it has been implemented purely for heuristic reasons. Note, the use of the relu activation function creates an ouput of 0 or 1 for the presence of each feature reducing the uncertainty if intermediary values between 0 and 1 were used. Also, the input shape must match the input of the 100 x 100 x 3 image files for the first network layer. Finally, a sigmoidal curve is used to output are binary class that will determine if the image represents a lymphocyte that is normal (negative) or malignant (positive).

In [88]:
```
# input for a 100x100 RBG image
model = Sequential ([
    Conv2D(32, (3,3), activation='relu',
    input_shape = (100,100,3)),
    MaxPooling2D((2,2)),

    Conv2D(32, (3,3), activation='relu'),
    MaxPooling2D((2,2)),

    Flatten(),
    Dense(64, activation = 'relu'),
    Dense(1, activation = 'sigmoid')
])
```

Compilation of the model

Compilation of the model incorporates the loss function into the model architecture to determine the optimal model

In [89]:
```
# compile mode
# categorical output = loss and metrics
model.compile(loss ='binary_crossentropy',
    optimizer='adam', metrics=['accuracy']
)
```

Fit Data to Model

In [101…
```
# fit training data to model
# rerun this cell to 'continue training' - don't recompile
model.fit(X_train, Y_train, epochs=5, batch_size=64)
```

```
Epoch 1/5
6/6 [==============================] - 1s 209ms/step - loss: 0.1979 - accuracy: 0.9405
Epoch 2/5
6/6 [==============================] - 1s 205ms/step - loss: 0.1052 - accuracy: 0.9792
Epoch 3/5
6/6 [==============================] - 1s 202ms/step - loss: 0.0428 - accuracy: 0.9881
Epoch 4/5
6/6 [==============================] - 1s 208ms/step - loss: 0.0115 - accuracy: 1.0000
```

```
Epoch 5/5
6/6 [==============================] - 1s 202ms/step - loss: 0.0078 - accuracy: 1.0000
```
Out[101…
```
<keras.callbacks.History at 0x20f87b43eb0>
```

Evaluate the Model

Initial accuracy of 100% indicates near perfect modelling; however, as one knows that we may be overfitting the data or just may need more data points. However, let's check the model performance:

In [102…
```
# evaluate performance
model.evaluate(X_test, Y_test)
```

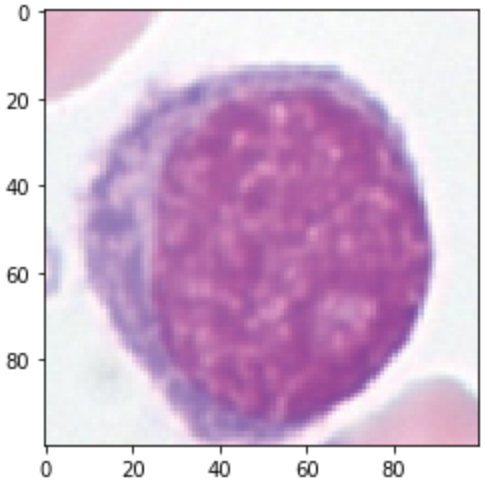
```
4/4 [==============================] - 0s 35ms/step - loss: 0.5580 - accuracy: 0.9018
```
Out[102…
```
[0.5579643249511719, 0.9017857313156128]
```

Make predictions

Note, the y_pred value that is the output of the sigmoidal curve consist of a probability value between 0 and 1 that the image represents the positive or negative label. In this case, a value greater than 0.5 is indicative that the image is classified as a malignant cell. To verbalize this notion the probability vlaue has been articulated with a conidition statement to print out 'malignant' or 'normal' alongside the associated picture.

In [117…
```
# make predictions
def predict():
    import random
    idx = random.randint(0, len(Y_test))
    plt.imshow(X_test[idx, :], interpolation='nearest')
    plt.show()
    y_pred = model.predict(X_test[idx, :].reshape(1, 100, 100, 3))
    if y_pred >0.5:
        pred = 'malignant'
    else:
        pred = 'normal'
    print(pred)
```

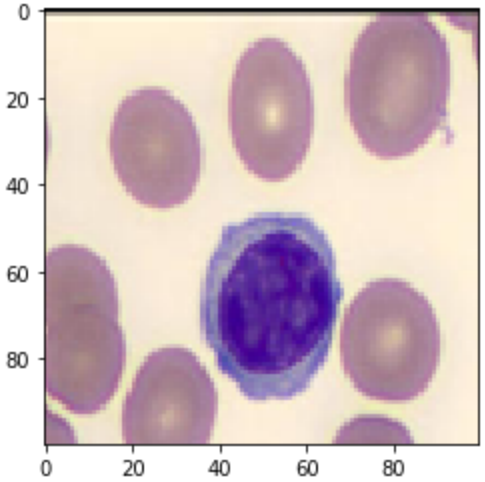
In [118…
```
predict()
```



malignant

In [119…

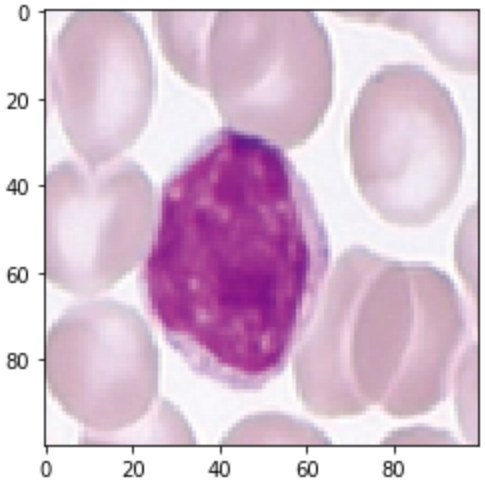
```
predict()
```



normal

In [120…

```
predict()
```




```
malignant
```

Just for demonstration we see that three randomly picked cells are actually classified correctly.

Summary

The convoluted neural network classifier for malignant lymphocytes proved to be fairly successful. The accuracy in the testing phase proved to be 90% while training accuracy approached 100%. The model performs excellent but secondary testing should be performed on a larger data set as well as possibly added in more cell types. This will test if the model has actually overfit in the training phase where generalization error may be present if the system is used in the 'real-world"

My next image classifier would include a multilabel classifier that is able to differentiate among the five major white blood cells (eosinophils, basophils, lymphocytes, neutrophils, and monocytes). Overall, the model proves to be fairly promising though even with slight overfitting. There is much promise that cell classification could serve as a great service to development of educational software for laboratory professionals as well as potential commerical usage if data sets may be obtain.

References:

- https://keras.io/
- https://pillow.readthedocs.io/en/stable/
- https://imagebank.hematology.org/