# Identify Blood Cell Subtypes From Images

- Basophil vs Eosinophil vs Lymphocyte vs Monocyte vs Neutrophil
- Mononuclear (Basophil + Lymphocyte vs Monocyte) vs Polynuclear (Neutrophil + Eosinophil)

## Import Libraries

In [1]:
```
import numpy as np
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D, Lambda, MaxPool2D, BatchNormalization
from keras.utils import np_utils
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import RMSprop
from sklearn.preprocessing import LabelEncoder
from sklearn.cross_validation import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
import xml.etree.ElementTree as ET
import sklearn
import itertools
import cv2
import scipy
import os
import csv
import matplotlib.pyplot as plt
%matplotlib inline
```

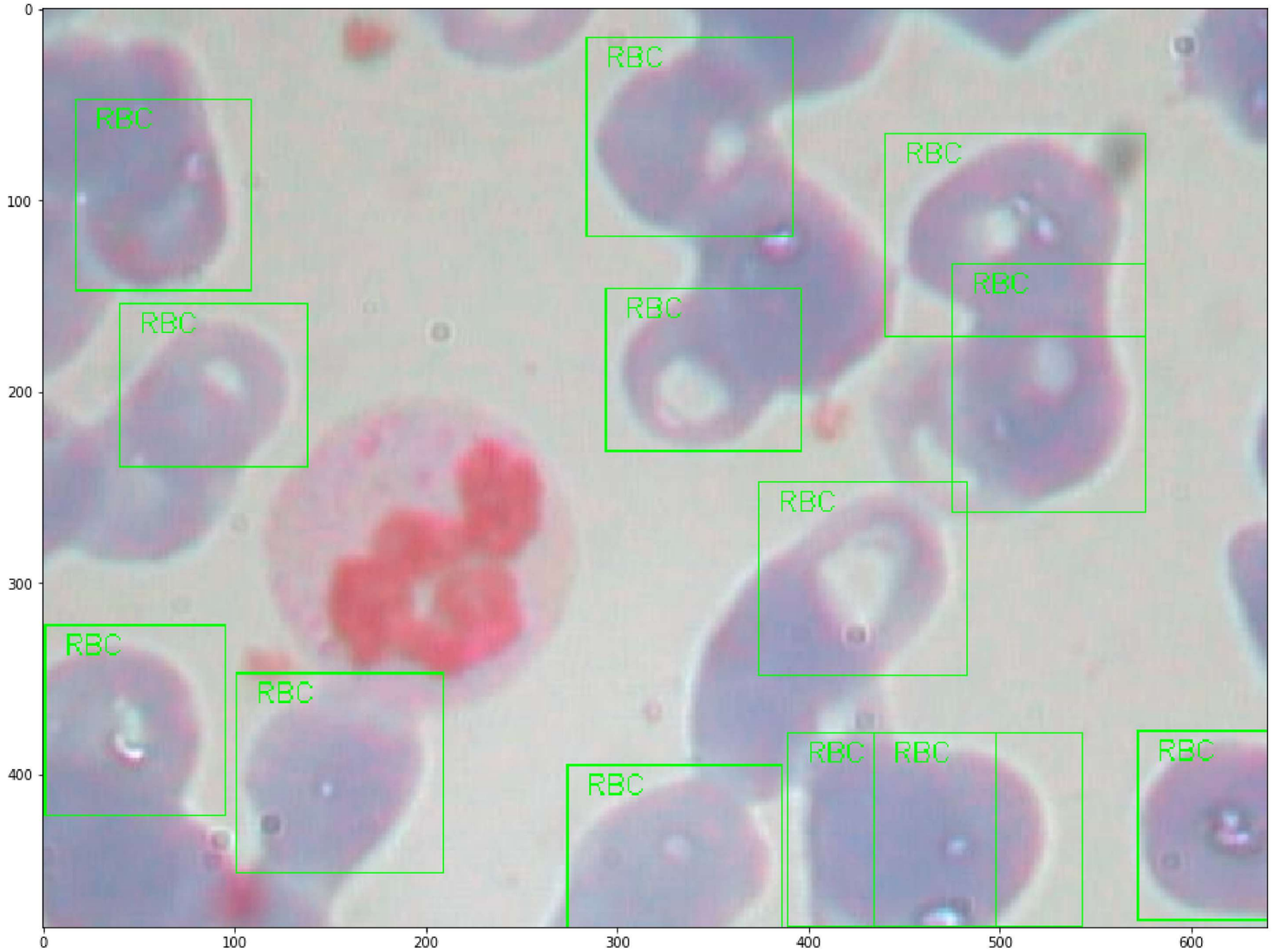
```
/opt/conda/lib/python3.6/site-packages/h5py/__init__.py:36: FutureWarning: Conversion of the second argument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
/opt/conda/lib/python3.6/site-packages/sklearn/cross_validation.py:41: DeprecationWarning: This module was deprecated in version 0.18 in favor of the model_selection module into which all the refactored classes and functions are moved. Also note that the interface of the new CV iterators are different from that of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

## Plot Data

In [2]:
```
dict_characters = {1:'NEUTROPHIL',2:'EOSINOPHIL',3:'MONOCYTE',4:'LYMPHOCYTE'}
dict_characters2 = {0:'Mononuclear',1:'Polynuclear'}
```

In [3]:
```
image = cv2.imread("../input/dataset-master/dataset-master/JPEGImages/BloodImage_00022.jpg")
tree = ET.parse("../input/dataset-master/dataset-master/Annotations/BloodImage_00022.xml")
for elem in tree.iter():
    if 'object' in elem.tag or 'part' in elem.tag:
        for attr in list(elem):
            if 'name' in attr.tag:
                name = attr.text
            if 'bndbox' in attr.tag:
                for dim in list(attr):
                    if 'xmin' in dim.tag:
                        xmin = int(round(float(dim.text)))
```

```
                    if 'ymin' in dim.tag:
                        ymin = int(round(float(dim.text)))
                    if 'xmax' in dim.tag:
                        xmax = int(round(float(dim.text)))
                    if 'ymax' in dim.tag:
                        ymax = int(round(float(dim.text)))
                if name[0] == "R":
                    cv2.rectangle(image, (xmin, ymin),
                                  (xmax, ymax), (0, 255, 0), 1)
                    cv2.putText(image, name, (xmin + 10, ymin + 15),
                            cv2.FONT_HERSHEY_SIMPLEX, 1e-3 * image.shape[0], (0, 255, 0), 1)
                if name[0] == "W":
                    cv2.rectangle(image, (xmin, ymin),
                                  (xmax, ymax), (0, 0, 255), 1)
                    cv2.putText(image, name, (xmin + 10, ymin + 15),
                            cv2.FONT_HERSHEY_SIMPLEX, 1e-3 * image.shape[0], (0, 0, 255), 1)
                if name[0] == "P":
                    cv2.rectangle(image, (xmin, ymin),
                                  (xmax, ymax), (255, 0, 0), 1)
                    cv2.putText(image, name, (xmin + 10, ymin + 15),
                            cv2.FONT_HERSHEY_SIMPLEX, 1e-3 * image.shape[0], (255, 0, 0), 1)
plt.figure(figsize=(16,16))
plt.imshow(image)
plt.show()
```



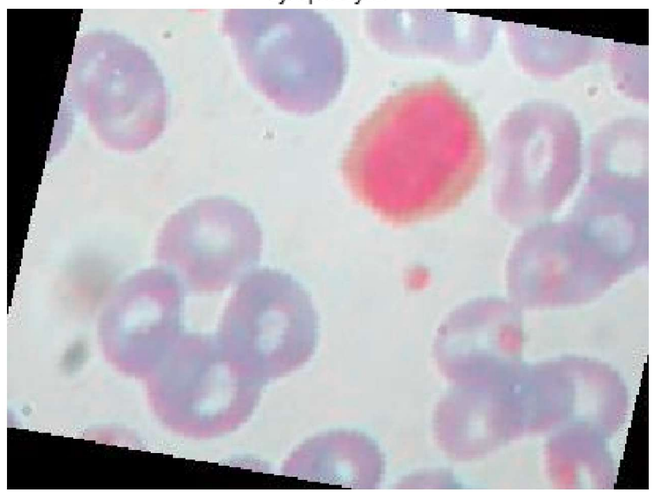

## Plot Image

In [4]:
```
def plotImage(image_location):
    image = cv2.imread(image_name)
    plt.imshow(image)
    return
image_name = '../input/dataset2-master/dataset2-master/images/TRAIN/EOSINOPHIL/_0_207.jpeg'
plt.figure(figsize=(16,16))
plt.subplot(221)
plt.title('Eosinophil')
plt.axis('off')
plotImage(image_name)
image_name = '../input/dataset2-master/dataset2-master/images/TRAIN/LYMPHOCYTE/_0_204.jpeg'
plt.subplot(222)
```

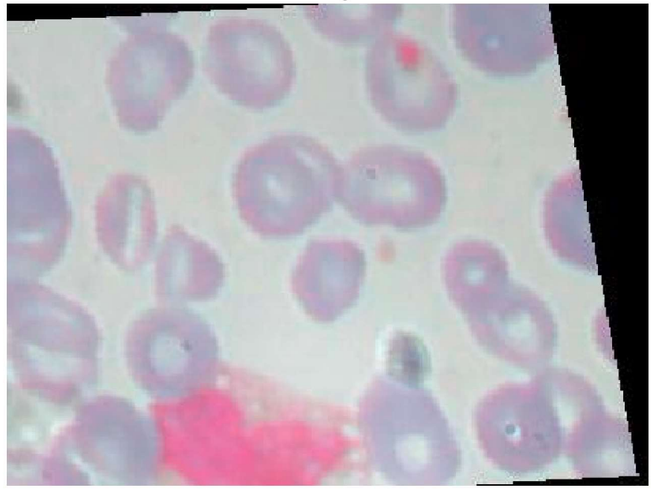
```
plt.title('Lymphocyte')
plt.axis('off')
plotImage(image_name)
image_name = '../input/dataset2-master/dataset2-master/images/TRAIN/MONOCYTE/_0_180.jpeg'
plt.subplot(223)
plt.title('Monocyte')
plt.axis('off')
plotImage(image_name)
plt.subplot(224)
image_name = '../input/dataset2-master/dataset2-master/images/TRAIN/NEUTROPHIL/_0_292.jpeg'
plt.title('Neutrophil')
plt.axis('off')
plotImage(image_name)
```

Eosinophil

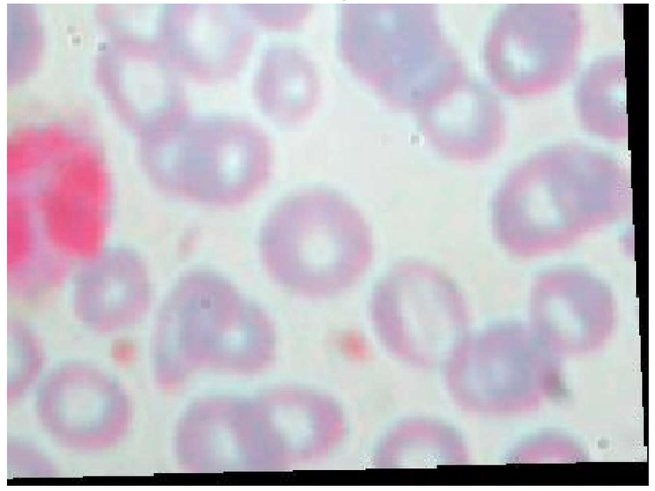
Lymphocyte



Monocyte



Neutrophil



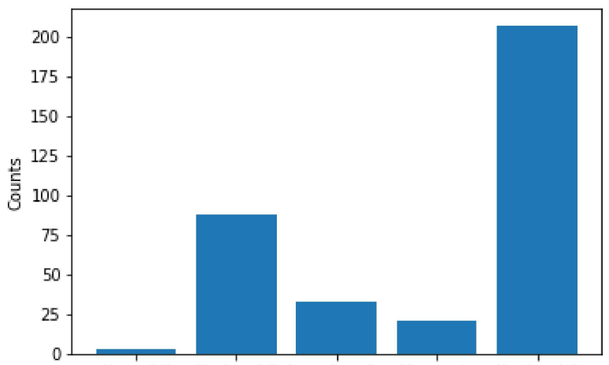
## Describe Data

In [5]:
```
reader = csv.reader(open('../input/dataset2-master/dataset2-master/labels.csv'))
# skip the header
next(reader)
X3 = []
y3 = []
```

```
for row in reader:
    label = row[2]
    if len(label) > 0 and label.find(',') == -1:
        y3.append(label)
y3 = np.asarray(y3)
encoder = LabelEncoder()
encoder.fit(y3)
encoded_y = encoder.transform(y3)
counts = np.bincount(encoded_y)
print(counts)
fig, ax = plt.subplots()
plt.bar(list(range(5)), counts)
ax.set_xticklabels(('', 'Basophil', 'Eosinophil', 'Lymphocyte', 'Monocyte', 'Neutrophil'))
ax.set_ylabel('Counts')
```

```
[  3  88  33  21 207]
```

Out[5]:
```
Text(0,0.5,'Counts')
```




You can see that the original images exhibit imbalanced class sizes. Instead, we will use the augmented images since they no longer have imbalanced class sizes due to oversampling.

## Load Augmented Dataset

In [6]:
```
from tqdm import tqdm
def get_data(folder):
    """
    Load the data and labels from the given folder.
    """
    X = []
    y = []
    z = []
    for wbc_type in os.listdir(folder):
        if not wbc_type.startswith('.'):
            if wbc_type in ['NEUTROPHIL']:
                label = 1
                label2 = 1
            elif wbc_type in ['EOSINOPHIL']:
                label = 2
                label2 = 1
            elif wbc_type in ['MONOCYTE']:
                label = 3
                label2 = 0
            elif wbc_type in ['LYMPHOCYTE']:
```

```
                label = 4
                label2 = 0
            else:
                label = 5
                label2 = 0
            for image_filename in tqdm(os.listdir(folder + wbc_type)):
                img_file = cv2.imread(folder + wbc_type + '/' + image_filename)
                if img_file is not None:
                    img_file = scipy.misc.imresize(arr=img_file, size=(60, 80, 3))
                    img_arr = np.asarray(img_file)
                    X.append(img_arr)
                    y.append(label)
                    z.append(label2)
    X = np.asarray(X)
    y = np.asarray(y)
    z = np.asarray(z)
    return X,y,z
X_train, y_train, z_train = get_data('../input/dataset2-master/dataset2-master/images/TRAIN/')
X_test, y_test, z_test = get_data('../input/dataset2-master/dataset2-master/images/TEST/')

# Encode labels to hot vectors (ex : 2 -> [0,0,1,0,0,0,0,0,0,0])
from keras.utils.np_utils import to_categorical
y_trainHot = to_categorical(y_train, num_classes = 5)
y_testHot = to_categorical(y_test, num_classes = 5)
z_trainHot = to_categorical(z_train, num_classes = 2)
z_testHot = to_categorical(z_test, num_classes = 2)
print(dict_characters)
print(dict_characters2)
```

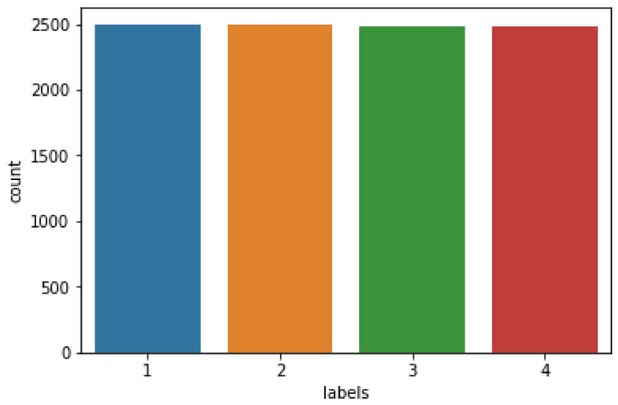
```
  0%|          | 0/2478 [00:00<?, ?it/s]/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:29: DeprecationWarning: `imresize` is deprecated!
`imresize` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``skimage.transform.resize`` instead.
100%|██████████| 2478/2478 [00:25<00:00, 97.32it/s]
100%|██████████| 2499/2499 [00:24<00:00, 100.37it/s]
100%|██████████| 2483/2483 [00:24<00:00, 101.33it/s]
100%|██████████| 2497/2497 [00:26<00:00, 92.83it/s]
100%|██████████| 620/620 [00:06<00:00, 102.91it/s]
100%|██████████| 624/624 [00:06<00:00, 96.16it/s]
100%|██████████| 620/620 [00:06<00:00, 100.09it/s]
100%|██████████| 623/623 [00:06<00:00, 100.04it/s]
{1: 'NEUTROPHIL', 2: 'EOSINOPHIL', 3: 'MONOCYTE', 4: 'LYMPHOCYTE'}
{0: 'Mononuclear', 1: 'Polynuclear'}
```

## Describe Augmented Dataset

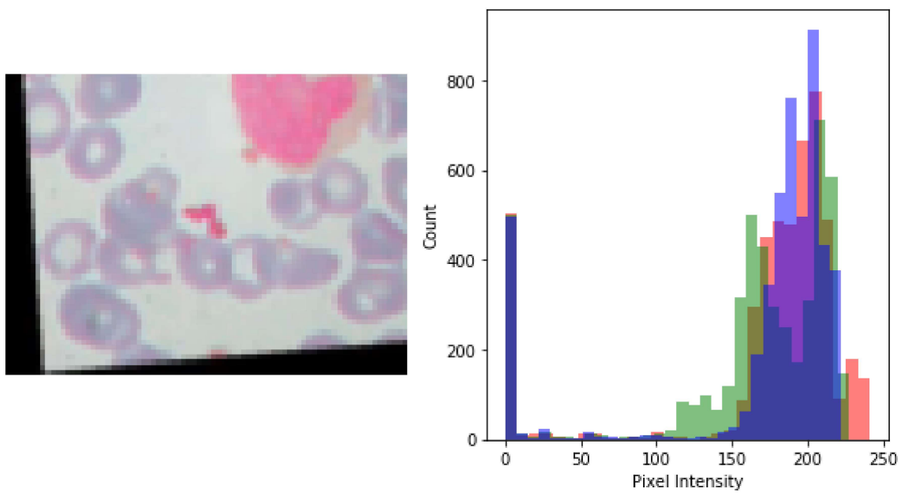
In [7]:

```
import seaborn as sns
df = pd.DataFrame()
df["labels"]=y_train
lab = df['labels']
dist = lab.value_counts()
sns.countplot(lab)
print(dict_characters)
```

```
{1: 'NEUTROPHIL', 2: 'EOSINOPHIL', 3: 'MONOCYTE', 4: 'LYMPHOCYTE'}
```




In [8]:
```
def plotHistogram(a):
    """
    Plot histogram of RGB Pixel Intensities
    """
    plt.figure(figsize=(10,5))
    plt.subplot(1,2,1)
    plt.imshow(a)
    plt.axis('off')
    histo = plt.subplot(1,2,2)
    histo.set_ylabel('Count')
    histo.set_xlabel('Pixel Intensity')
    n_bins = 30
    plt.hist(a[:,:,0].flatten(), bins= n_bins, lw = 0, color='r', alpha=0.5);
    plt.hist(a[:,:,1].flatten(), bins= n_bins, lw = 0, color='g', alpha=0.5);
    plt.hist(a[:,:,2].flatten(), bins= n_bins, lw = 0, color='b', alpha=0.5);
plotHistogram(X_train[1])
```

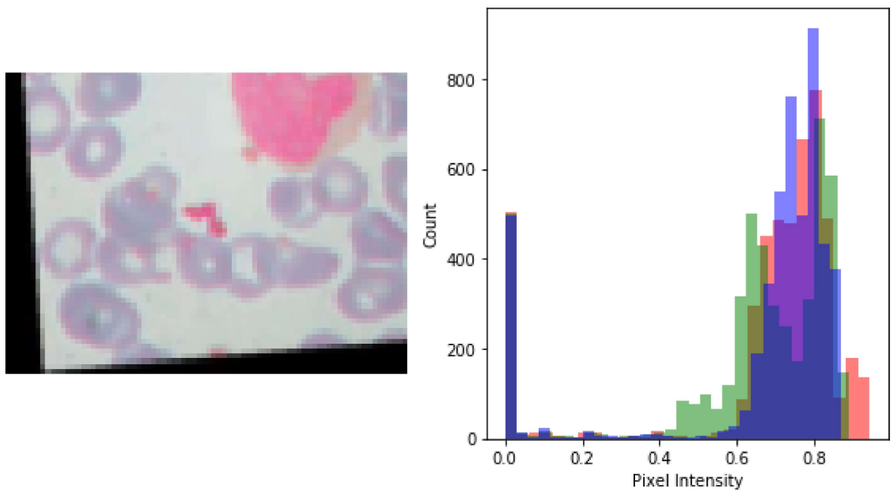



## Preprocess Data

In [9]:
```
X_train=np.array(X_train)
X_train=X_train/255.0
```

```
X_test=np.array(X_test)
X_test=X_test/255.0
```

In [10]:
```
plotHistogram(X_train[1])
```




## Define Helper Functions

In [11]:
```
# Helper Functions: Learning Curves and Confusion Matrix

from keras.callbacks import Callback, EarlyStopping, ReduceLROnPlateau, ModelCheckpoint

class MetricsCheckpoint(Callback):
    """Callback that saves metrics after each epoch"""
    def __init__(self, savepath):
        super(MetricsCheckpoint, self).__init__()
        self.savepath = savepath
        self.history = {}
    def on_epoch_end(self, epoch, logs=None):
        for k, v in logs.items():
            self.history.setdefault(k, []).append(v)
        np.save(self.savepath, self.history)

def plotKerasLearningCurve():
    plt.figure(figsize=(10,5))
    metrics = np.load('logs.npy')[()]
    filt = ['acc'] # try to add 'loss' to see the loss learning curve
    for k in filter(lambda x : np.any([kk in x for kk in filt]), metrics.keys()):
        l = np.array(metrics[k])
        plt.plot(l, c= 'r' if 'val' not in k else 'b', label='val' if 'val' in k else 'train')
        x = np.argmin(l) if 'loss' in k else np.argmax(l)
        y = l[x]
        plt.scatter(x,y, lw=0, alpha=0.25, s=100, c='r' if 'val' not in k else 'b')
        plt.text(x, y, '{} = {:.4f}'.format(x,y), size='15', color= 'r' if 'val' not in k else 'b')
    plt.legend(loc=4)
    plt.axis([0, None, None, None]);
    plt.grid()
    plt.xlabel('Number of epochs')
```

```
    plt.ylabel('Accuracy')

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.figure(figsize = (5,5))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90)
    plt.yticks(tick_marks, classes)
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

def plot_learning_curve(history):
    plt.figure(figsize=(8,8))
    plt.subplot(1,2,1)
    plt.plot(history.history['acc'])
    plt.plot(history.history['val_acc'])
    plt.title('model accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.savefig('./accuracy_curve.png')
    #plt.clf()
    # summarize history for loss
    plt.subplot(1,2,2)
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('model loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.savefig('./loss_curve.png')
```

## Evaluate Classification Models

In [12]:

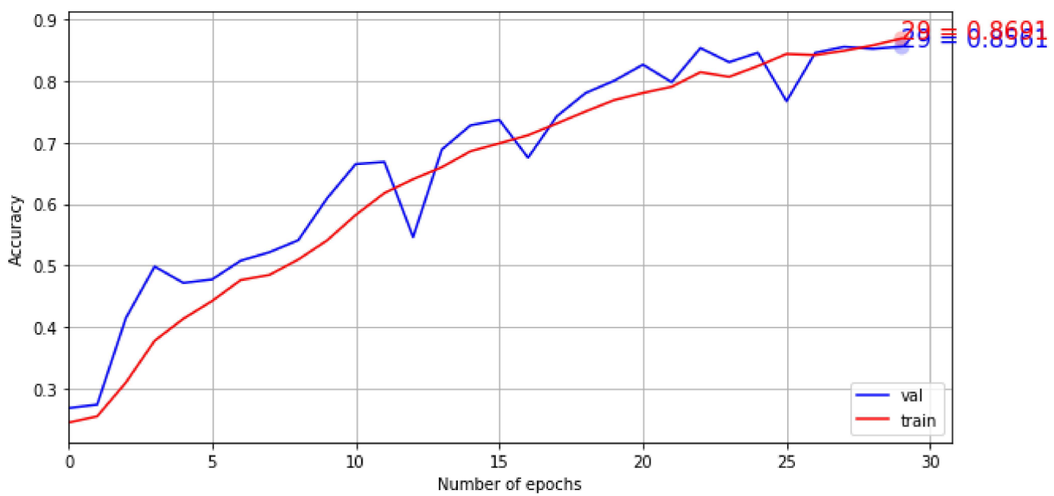
```
import keras
dict_characters = {1:'NEUTROPHIL',2:'EOSINOPHIL',3:'MONOCYTE',4:'LYMPHOCYTE'}
dict_characters2 = {0:'Mononuclear',1:'Polynuclear'}
def runKerasCNNAugment(a,b,c,d,e):
    batch_size = 128
    num_classes = len(b[0])
```

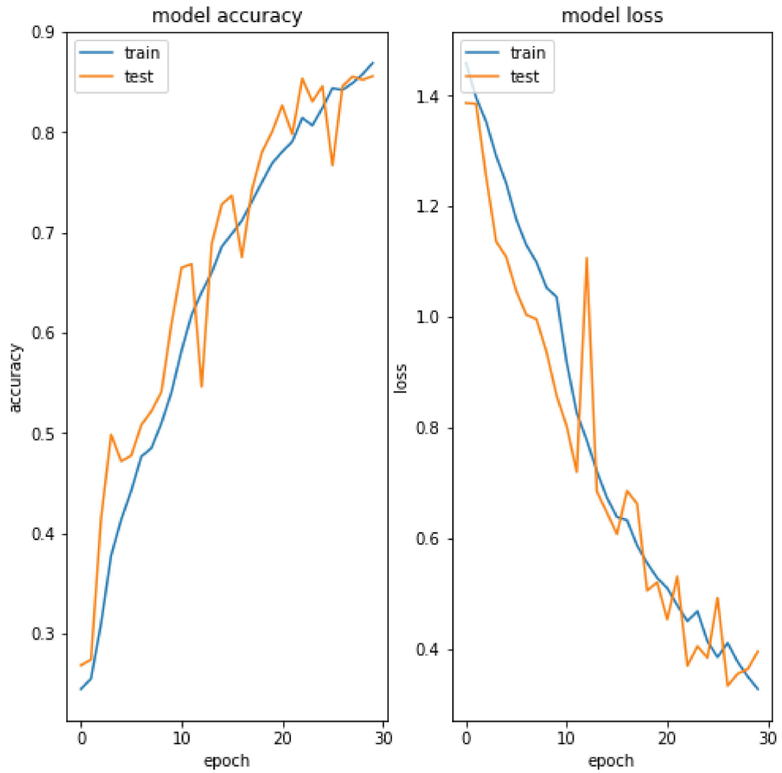
```
    epochs = 30
#     img_rows, img_cols = a.shape[1],a.shape[2]
    img_rows,img_cols=60,80
    input_shape = (img_rows, img_cols, 3)
    model = Sequential()
    model.add(Conv2D(32, kernel_size=(3, 3),
                     activation='relu',
                     input_shape=input_shape,strides=e))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes, activation='softmax'))
    model.compile(loss=keras.losses.categorical_crossentropy,
                  optimizer=keras.optimizers.Adadelta(),
                  metrics=['accuracy'])
    datagen = ImageDataGenerator(
        featurewise_center=False,  # set input mean to 0 over the dataset
        samplewise_center=False,  # set each sample mean to 0
        featurewise_std_normalization=False,  # divide inputs by std of the dataset
        samplewise_std_normalization=False,  # divide each input by its std
        zca_whitening=False,  # apply ZCA whitening
        rotation_range=10,  # randomly rotate images in the range (degrees, 0 to 180)
        width_shift_range=0.1,  # randomly shift images horizontally (fraction of total width)
        height_shift_range=0.1,  # randomly shift images vertically (fraction of total height)
        horizontal_flip=True,  # randomly flip images
        vertical_flip=False)  # randomly flip images
    history = model.fit_generator(datagen.flow(a,b, batch_size=32),
                        steps_per_epoch=len(a) / 32, epochs=epochs, validation_data = [c, d],callbacks = [MetricsCheckpoint('logs')])
    score = model.evaluate(c,d, verbose=0)
    print('\nKeras CNN #1C - accuracy:', score[1],'\n')
    y_pred = model.predict(c)
    map_characters = dict_characters
    print('\n', sklearn.metrics.classification_report(np.where(d > 0)[1], np.argmax(y_pred, axis=1), target_names=list(map_characters.values())), sep='')
    Y_pred_classes = np.argmax(y_pred,axis=1)
    Y_true = np.argmax(d,axis=1)
    plotKerasLearningCurve()
    plt.show()
    plot_learning_curve(history)
    plt.show()
    confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)
    plot_confusion_matrix(confusion_mtx, classes = list(dict_characters.values()))
    plt.show()
runKerasCNNAugment(X_train,y_trainHot,X_test,y_testHot,1)
```

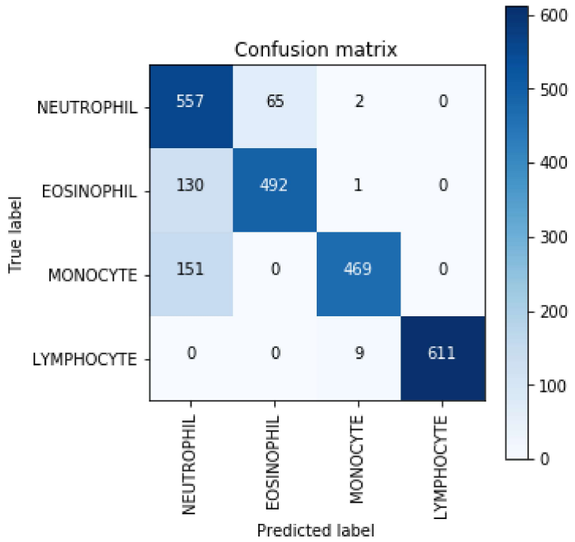
```
Epoch 1/30
312/311 [==============================] - 32s 101ms/step - loss: 1.4590 - acc: 0.2448 - val_loss: 1.3866 - val_acc: 0.2686
Epoch 2/30
312/311 [==============================] - 22s 71ms/step - loss: 1.3968 - acc: 0.2556 - val_loss: 1.3852 - val_acc: 0.2742
Epoch 3/30
312/311 [==============================] - 22s 71ms/step - loss: 1.3532 - acc: 0.3098 - val_loss: 1.2535 - val_acc: 0.4150
Epoch 4/30
312/311 [==============================] - 22s 70ms/step - loss: 1.2903 - acc: 0.3785 - val_loss: 1.1365 - val_acc: 0.4986
Epoch 5/30
312/311 [==============================] - 22s 71ms/step - loss: 1.2410 - acc: 0.4143 - val_loss: 1.1083 - val_acc: 0.4721
Epoch 6/30
312/311 [==============================] - 22s 70ms/step - loss: 1.1765 - acc: 0.4419 - val_loss: 1.0467 - val_acc: 0.4777
Epoch 7/30
312/311 [==============================] - 22s 71ms/step - loss: 1.1298 - acc: 0.4765 - val_loss: 1.0037 - val_acc: 0.5082
Epoch 8/30
312/311 [==============================] - 22s 70ms/step - loss: 1.0994 - acc: 0.4843 - val_loss: 0.9957 - val_acc: 0.5219
Epoch 9/30
312/311 [==============================] - 22s 70ms/step - loss: 1.0528 - acc: 0.5108 - val_loss: 0.9363 - val_acc: 0.5412
Epoch 10/30
312/311 [==============================] - 22s 70ms/step - loss: 1.0356 - acc: 0.5415 - val_loss: 0.8578 - val_acc: 0.6092
Epoch 11/30
312/311 [==============================] - 22s 70ms/step - loss: 0.9176 - acc: 0.5824 - val_loss: 0.8027 - val_acc: 0.6651
Epoch 12/30
312/311 [==============================] - 22s 71ms/step - loss: 0.8266 - acc: 0.6174 - val_loss: 0.7198 - val_acc: 0.6687
Epoch 13/30
312/311 [==============================] - 22s 71ms/step - loss: 0.7763 - acc: 0.6410 - val_loss: 1.1068 - val_acc: 0.5464
Epoch 14/30
312/311 [==============================] - 22s 70ms/step - loss: 0.7218 - acc: 0.6599 - val_loss: 0.6844 - val_acc: 0.6888
Epoch 15/30
312/311 [==============================] - 22s 71ms/step - loss: 0.6741 - acc: 0.6853 - val_loss: 0.6464 - val_acc: 0.7282
Epoch 16/30
312/311 [==============================] - 22s 71ms/step - loss: 0.6384 - acc: 0.6991 - val_loss: 0.6077 - val_acc: 0.7370
Epoch 17/30
312/311 [==============================] - 22s 71ms/step - loss: 0.6335 - acc: 0.7112 - val_loss: 0.6857 - val_acc: 0.6755
Epoch 18/30
312/311 [==============================] - 22s 70ms/step - loss: 0.5867 - acc: 0.7312 - val_loss: 0.6625 - val_acc: 0.7427
Epoch 19/30
312/311 [==============================] - 22s 72ms/step - loss: 0.5538 - acc: 0.7512 - val_loss: 0.5056 - val_acc: 0.7805
Epoch 20/30
312/311 [==============================] - 22s 71ms/step - loss: 0.5282 - acc: 0.7687 - val_loss: 0.5202 - val_acc: 0.8006
Epoch 21/30
312/311 [==============================] - 22s 71ms/step - loss: 0.5097 - acc: 0.7807 - val_loss: 0.4534 - val_acc: 0.8267
Epoch 22/30
312/311 [==============================] - 22s 72ms/step - loss: 0.4798 - acc: 0.7902 - val_loss: 0.5309 - val_acc: 0.7982
Epoch 23/30
312/311 [==============================] - 22s 71ms/step - loss: 0.4496 - acc: 0.8149 - val_loss: 0.3696 - val_acc: 0.8536
Epoch 24/30
312/311 [==============================] - 22s 71ms/step - loss: 0.4689 - acc: 0.8069 - val_loss: 0.4047 - val_acc: 0.8307
Epoch 25/30
312/311 [==============================] - 22s 71ms/step - loss: 0.4127 - acc: 0.8248 - val_loss: 0.3839 - val_acc: 0.8460
Epoch 26/30
312/311 [==============================] - 22s 71ms/step - loss: 0.3851 - acc: 0.8439 - val_loss: 0.4921 - val_acc: 0.7672
Epoch 27/30
312/311 [==============================] - 22s 70ms/step - loss: 0.4104 - acc: 0.8421 - val_loss: 0.3334 - val_acc: 0.8460
Epoch 28/30
312/311 [==============================] - 23s 73ms/step - loss: 0.3756 - acc: 0.8496 - val_loss: 0.3551 - val_acc: 0.8556
Epoch 29/30
312/311 [==============================] - 22s 72ms/step - loss: 0.3501 - acc: 0.8584 - val_loss: 0.3635 - val_acc: 0.8524
Epoch 30/30
312/311 [==============================] - 21s 67ms/step - loss: 0.3273 - acc: 0.8690 - val_loss: 0.3953 - val_acc: 0.8561
```

Keras CNN #1C - accuracy: 0.8560514675597853

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| NEUTROPHIL | 0.66 | 0.89 | 0.76 | 624 |
| EOSINOPHIL | 0.88 | 0.79 | 0.83 | 623 |
| MONOCYTE | 0.98 | 0.76 | 0.85 | 620 |
| LYMPHOCYTE | 1.00 | 0.99 | 0.99 | 620 |
| avg / total | 0.88 | 0.86 | 0.86 | 2487 |






85% accuracy is much better than random chance. Now, let's see if this model might work better for a binary classification task, such as determining whether the cell is polynuclear (e.g., neutrophil and eosinophil) or mononuclear (e.g., every other subtype).

In [13]:
```
dict_characters = dict_characters2
runKerasCNNAugment(X_train,z_trainHot,X_test,z_testHot,2)
```

```
dict_characters = {1:'NEUTROPHIL',2:'EOSINOPHIL',3:'MONOCYTE',4:'LYMPHOCYTE'}
dict_characters2 = {0:'Mononuclear',1:'Polynuclear'}
```

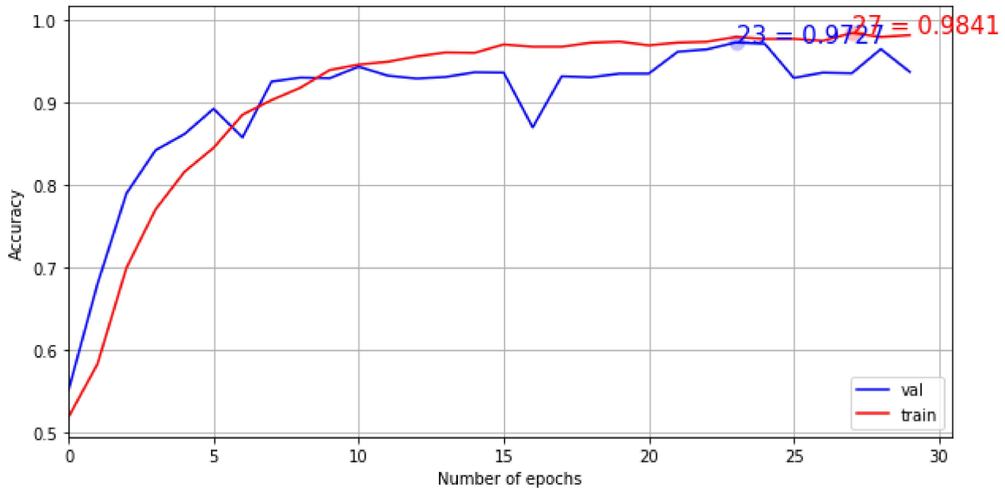
```
Epoch 1/30
312/311 [==============================] - 22s 69ms/step - loss: 0.7052 - acc: 0.5192 - val_loss: 0.6841 - val_acc: 0.5509
Epoch 2/30
312/311 [==============================] - 21s 68ms/step - loss: 0.6763 - acc: 0.5818 - val_loss: 0.6370 - val_acc: 0.6799
Epoch 3/30
312/311 [==============================] - 20s 65ms/step - loss: 0.5736 - acc: 0.6992 - val_loss: 0.4757 - val_acc: 0.7897
Epoch 4/30
312/311 [==============================] - 20s 65ms/step - loss: 0.4821 - acc: 0.7697 - val_loss: 0.3662 - val_acc: 0.8420
Epoch 5/30
312/311 [==============================] - 20s 66ms/step - loss: 0.4045 - acc: 0.8159 - val_loss: 0.3133 - val_acc: 0.8617
Epoch 6/30
312/311 [==============================] - 20s 66ms/step - loss: 0.3410 - acc: 0.8443 - val_loss: 0.2592 - val_acc: 0.8922
Epoch 7/30
312/311 [==============================] - 20s 65ms/step - loss: 0.2760 - acc: 0.8854 - val_loss: 0.3286 - val_acc: 0.8577
Epoch 8/30
312/311 [==============================] - 20s 65ms/step - loss: 0.2376 - acc: 0.9030 - val_loss: 0.1643 - val_acc: 0.9252
Epoch 9/30
312/311 [==============================] - 21s 66ms/step - loss: 0.1973 - acc: 0.9181 - val_loss: 0.1555 - val_acc: 0.9300
Epoch 10/30
312/311 [==============================] - 20s 65ms/step - loss: 0.1612 - acc: 0.9393 - val_loss: 0.1620 - val_acc: 0.9292
Epoch 11/30
312/311 [==============================] - 20s 63ms/step - loss: 0.1386 - acc: 0.9455 - val_loss: 0.1212 - val_acc: 0.9433
Epoch 12/30
312/311 [==============================] - 21s 67ms/step - loss: 0.1385 - acc: 0.9493 - val_loss: 0.1714 - val_acc: 0.9324
Epoch 13/30
312/311 [==============================] - 21s 68ms/step - loss: 0.1147 - acc: 0.9558 - val_loss: 0.2552 - val_acc: 0.9288
Epoch 14/30
312/311 [==============================] - 20s 65ms/step - loss: 0.1021 - acc: 0.9606 - val_loss: 0.1896 - val_acc: 0.9308
Epoch 15/30
312/311 [==============================] - 21s 66ms/step - loss: 0.1026 - acc: 0.9601 - val_loss: 0.1349 - val_acc: 0.9365
Epoch 16/30
312/311 [==============================] - 20s 64ms/step - loss: 0.0791 - acc: 0.9704 - val_loss: 0.1322 - val_acc: 0.9361
Epoch 17/30
312/311 [==============================] - 20s 64ms/step - loss: 0.0880 - acc: 0.9675 - val_loss: 0.5952 - val_acc: 0.8697
Epoch 18/30
312/311 [==============================] - 20s 65ms/step - loss: 0.0904 - acc: 0.9675 - val_loss: 0.1479 - val_acc: 0.9316
Epoch 19/30
312/311 [==============================] - 20s 64ms/step - loss: 0.0724 - acc: 0.9725 - val_loss: 0.2017 - val_acc: 0.9304
Epoch 20/30
312/311 [==============================] - 21s 67ms/step - loss: 0.0702 - acc: 0.9738 - val_loss: 0.1448 - val_acc: 0.9349
Epoch 21/30
312/311 [==============================] - 21s 66ms/step - loss: 0.0886 - acc: 0.9693 - val_loss: 0.1682 - val_acc: 0.9349
Epoch 22/30
312/311 [==============================] - 21s 67ms/step - loss: 0.0752 - acc: 0.9726 - val_loss: 0.0836 - val_acc: 0.9614
Epoch 23/30
312/311 [==============================] - 21s 66ms/step - loss: 0.0792 - acc: 0.9736 - val_loss: 0.0729 - val_acc: 0.9642
Epoch 24/30
312/311 [==============================] - 21s 68ms/step - loss: 0.0541 - acc: 0.9794 - val_loss: 0.0700 - val_acc: 0.9727
Epoch 25/30
312/311 [==============================] - 21s 69ms/step - loss: 0.0638 - acc: 0.9770 - val_loss: 0.0577 - val_acc: 0.9710
Epoch 26/30
312/311 [==============================] - 21s 68ms/step - loss: 0.0552 - acc: 0.9772 - val_loss: 0.3446 - val_acc: 0.9296
Epoch 27/30
312/311 [==============================] - 21s 68ms/step - loss: 0.0725 - acc: 0.9748 - val_loss: 0.3085 - val_acc: 0.9361
Epoch 28/30
312/311 [==============================] - 21s 67ms/step - loss: 0.0468 - acc: 0.9836 - val_loss: 0.1252 - val_acc: 0.9353
Epoch 29/30
312/311 [==============================] - 20s 64ms/step - loss: 0.0724 - acc: 0.9794 - val_loss: 0.0752 - val_acc: 0.9646
Epoch 30/30
312/311 [==============================] - 21s 67ms/step - loss: 0.0483 - acc: 0.9816 - val_loss: 0.3312 - val_acc: 0.9369
```

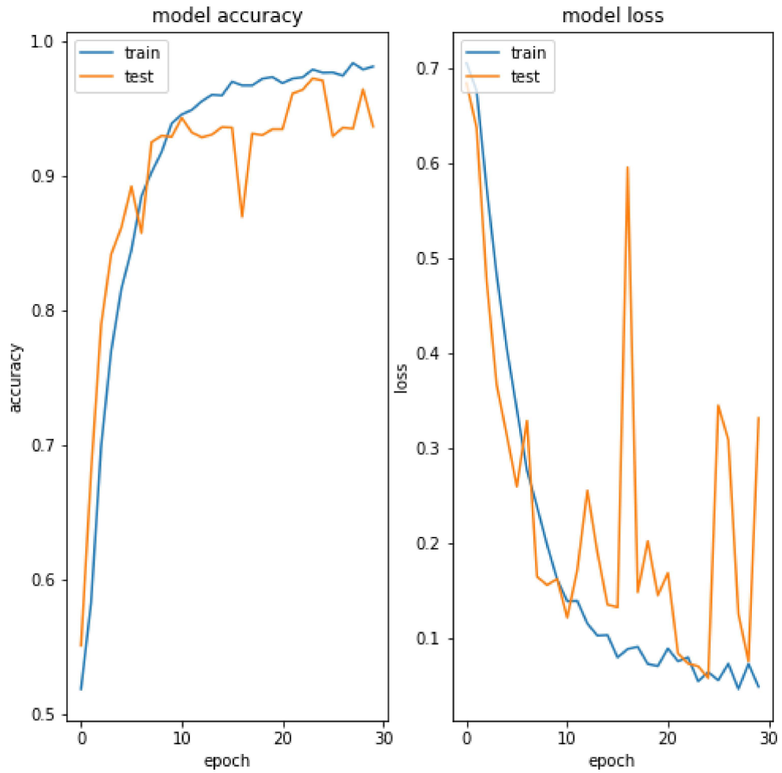
```
Keras CNN #1C - accuracy: 0.9368717330116606
```

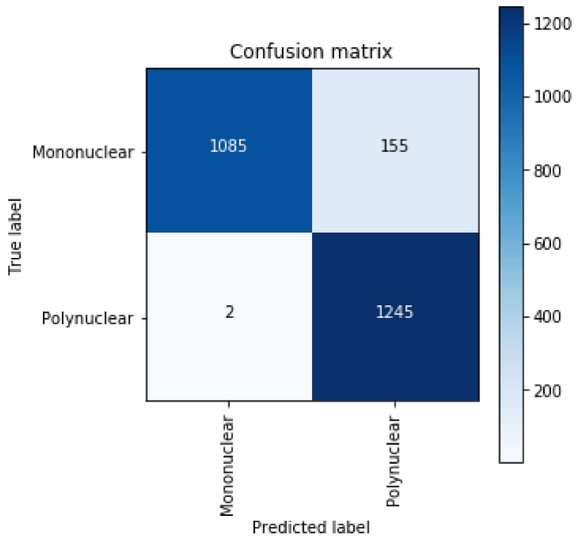
```
             precision    recall  f1-score   support

Mononuclear       1.00      0.88      0.93      1240
Polynuclear       0.89      1.00      0.94      1247

avg / total       0.94      0.94      0.94      2487
```






**That worked reasonably well, achieving 93% accuracy for two categories {1:'MONONUCLEAR', 2:'POLYNUCLEAR'}, compared to 85% accuracy for four categories {1:'NEUTROPHIL', 2:'EOSINOPHIL', 3:'MONOCYTE', 4:'LYMPHOCYTE'}.**

Automated methods such as this can be utilized to save time and enhance efficiency in clinical settings.