# Capstone Project

Venkata Ramarao

Machine Learning Engineer Nanodegree

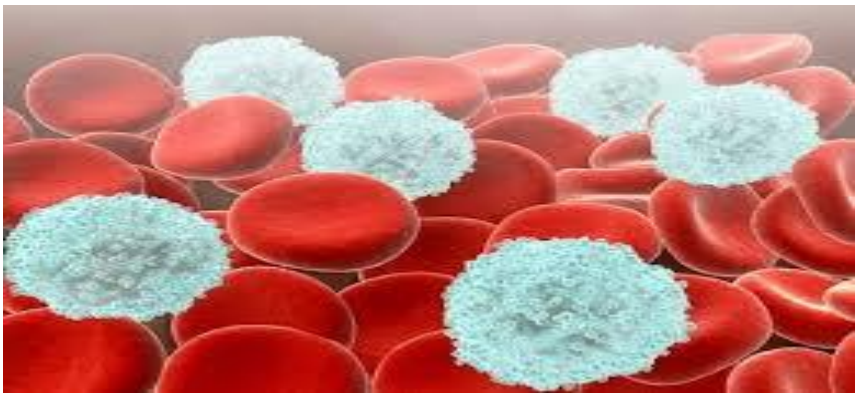
03/07/2017

## Definition

### Project Overview

White blood cells (also known as WBC or leukocytes) help our body fight infections by attacking bacteria, viruses and other germs that invade the body. A count of leukocytes can help reveal several hidden and undiagnosed diseases. In a manual microscopic review of blood samples, pathologists minutely examine the count and morphology (i.e. size and shape) of white blood cells.

Red blood cells (or RBC, or erythrocytes) are the most common type of blood cells, and they outnumber WBCs by about 600:1. So, in an image of a blood sample, you will see mostly RBCs, with a few WBCs thrown in here and there.



In manual process pathologists analyze the blood sample and count the WBC (white blood cells), but this is not accurately defined the correct count and related disease, they use pre-defined approach to determine the health of the person. But if we use Image segmentation using deep learning supervised algorithm model, it accurately demarcates the boundary of WBC even when they are touching each other and identify correct count. This will improve the accuracy and speed of testing and yield better results.

I hope by using state of the art deep learning model for this image segmentation task will improve the accuracy of results and correctly identify the health of a person.

### Problem Statement

The goal is to developing an efficient Deep Learning model using CNN (Convolutional Neural Networks) to accurately demarcate the boundary of white blood cells in microscopic images of blood

The final model can generalize to different segmentation tasks involving identifying cancel cells in human body images, segmenting heart pictures with right cells etc.

## Metrics

We use Dice coefficient or F1 score for evaluation of the model on predicted masks with ground truth values.

The Dice score is often used to quantify the performance of image segmentation methods. There you annotate some ground truth region in your image and then make an automated algorithm to do it. You validate the algorithm by calculating the Dice score, which is a measure of how similar the objects are. So it is the size of the overlap of the two segmentations divided by the total size of the two objects. Using the same terms as describing accuracy, the Dice score is:

***Dice score=number of true positives/number of positives + number of false positives***

So the number of true positives, is the number that your method finds, the number of positives is the total number of positives that can be found and the number of false positives is the number of points that are negative that your method classifies as positive.

If the dice score is high, the model is correctly segmenting given images with right classes.

# Analysis

## Data Exploration

The dataset for this project originates from the SigTuple AI Challenge. SigTuple released this data set to hire top AI talent for their engineering team. The data set is available at the hackathon competition.

I thought this is a good research problem and have a scope to select as a capstone for Machine Learning Nano degree.

Structure of the data set:

```
├── Test_Data (61 files)
|    ├── 017532875DDF.jpg
|    ├── 029E137BB177.jpg
|    ├── 029E137BB179.jpg
|    ├── ...
└── Train_Data (328 files)
     ├── train-0.jpg
```

├── train-0-mask.jpg

├── train-100.jpg

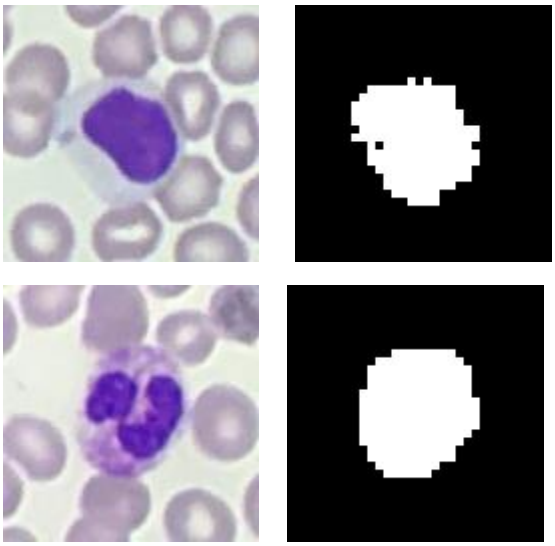
├── train-100-mask.jpg

├── ...

The training set consists of 164 (128X128) patches showing WBCs, and the area has been demarcated in a mask file. The cell at the center of these patches are WBCs, while those surrounding the WBC are RBCs. The files are named like train-0.jpg, train-1.jpg, .... The corresponding mask files are named train-0-mask.jpg, train-1-mask.jpg, ..., respectively. There are also around 5 larger images (and corresponding masks) of blood, showing one or more WBCs in the image.

The test set will consist of larger images of blood smears, from which we need to demarcate the WBC boundaries. My code has to both locate the WBCs and demarcate their boundary.

## Data Visualization



From the above pictures, the left side is train image with both WBCs (Blue region) and RBCs and right side is corresponding masked image with demarcated WBC boundaries. So, our model has to locate the WBCs and demarcate their boundary.

## Algorithms and Techniques

Convolutional neural networks will yield state of the art results on image processing like object detection, classification and segmentation.

For this problem I am using Convolutional neural networks for identifying WBCs boundary.

Convolutional neural network is a combination of convolutional map, filters, max pooling layers with activation and dropout functions.

For a given input image having dimensions (1,128,128,3) -> (number_of_samples, img_rows, img_cols, color_channels)

- First we need to define filters to run through entire image for capturing different features of image like edges, shapes etc.
- each filter slide through entire image with activation function like ReLU (rectified linear unit) produce a convolutional map
- Again we can apply same filters to better capture different features and produce an another convolutional map
- Although the role of the convolutional layer is to detect local conjunctions of features from the previous layer, the role of the pooling layer is to merge semantically similar features into one. Because the relative positions of the features forming a motif can vary somewhat, reliably detecting the motif can be done by coarse-graining the position of each feature. A typical pooling unit computes the maximum of a local patch of units in one feature map (or in a few feature maps). Neighboring pooling units take input from patches that are shifted by more than one row or column, thereby reducing the dimension of the representation and creating an invariance to small shifts and distortions. Two or three stages of convolution, non-linearity and pooling are stacked, followed by more convolutional and fully-connected layers.

If we apply convolutional and pooling operations sequentially on the given image, the image size will decrease deep down and it will not return correct masked area of WBCs. So in order to keep the image as equal to the input shape. We are going to apply de-convolution on convolutional feature maps to restore the image to original image resolution and correctly identify the demarcated area of white blood cells.

## Benchmark

I am using modified U-net model and training strategy that relies on the strong use of data augmentation to use the available annotated samples more efficiently. The architecture consists of a contracting path(Convolutions) to capture context and a symmetric expanding path(Deconvolution) that enables precise localization.

The modified model follows the same pattern of U-net but with adding drop-out and augmenting on rotation and flipping original images. So that the model able to see more data and well generalize and invariance to distortions in images.

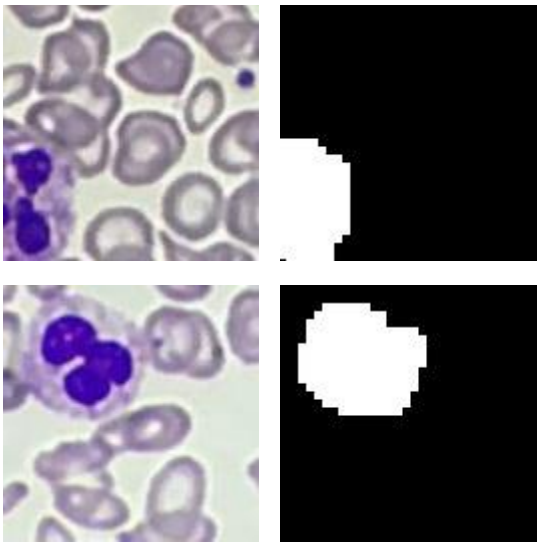
**The original U-net model have achieved 77.5% accuracy on DIC-HeLa data set from ISBI cell tracking challenge. I am taking this is as a benchmark and try to achieve more accuracy with modified U-net architecture.**

# Methodology

## Data Preprocessing

Before data can be used as input for machine learning algorithms, it often must be cleaned, formatted, and restructured — this is typically known as preprocessing. Fortunately, for this dataset, there are no invalid data we must deal with, however, there are 5 large images (corresponding masks) of blood. So we need to re-size these images to 128 * 128 without losing the pixel information from both train and mask data and store it in new folder called NEW_TRAIN_DATA. So that all the images (both train and mask) are in same dimension and pass to our deep learning algorithm for training.

Images in new folder after preprocessing large images



## Implementation

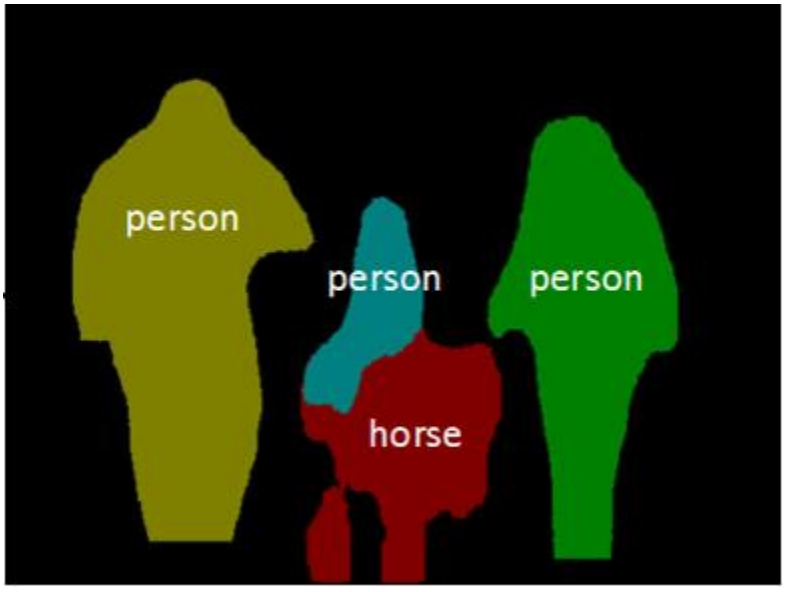
From the given training data set(SigTuple_data/Train_Data) the train images (train-0.jpg, train-1.jpg, train-2.jpg.. etc.) can act as a training set and mask images (train-0-mask.jpg, train-1-mask.jpg, train-2-mask.jpg.. etc.) as dependent to fit to our train images. We can create these two sets as X_train and Y_train and pass these for training to our model. Here we are using tensorflow as a back end for our Keras deep learning model.

When using TensorFlow as backend, Keras CNNs require a 4D array (which we'll also refer to as a 4D tensor) as input, with shape

(nbsamples,rows,columns,channels)

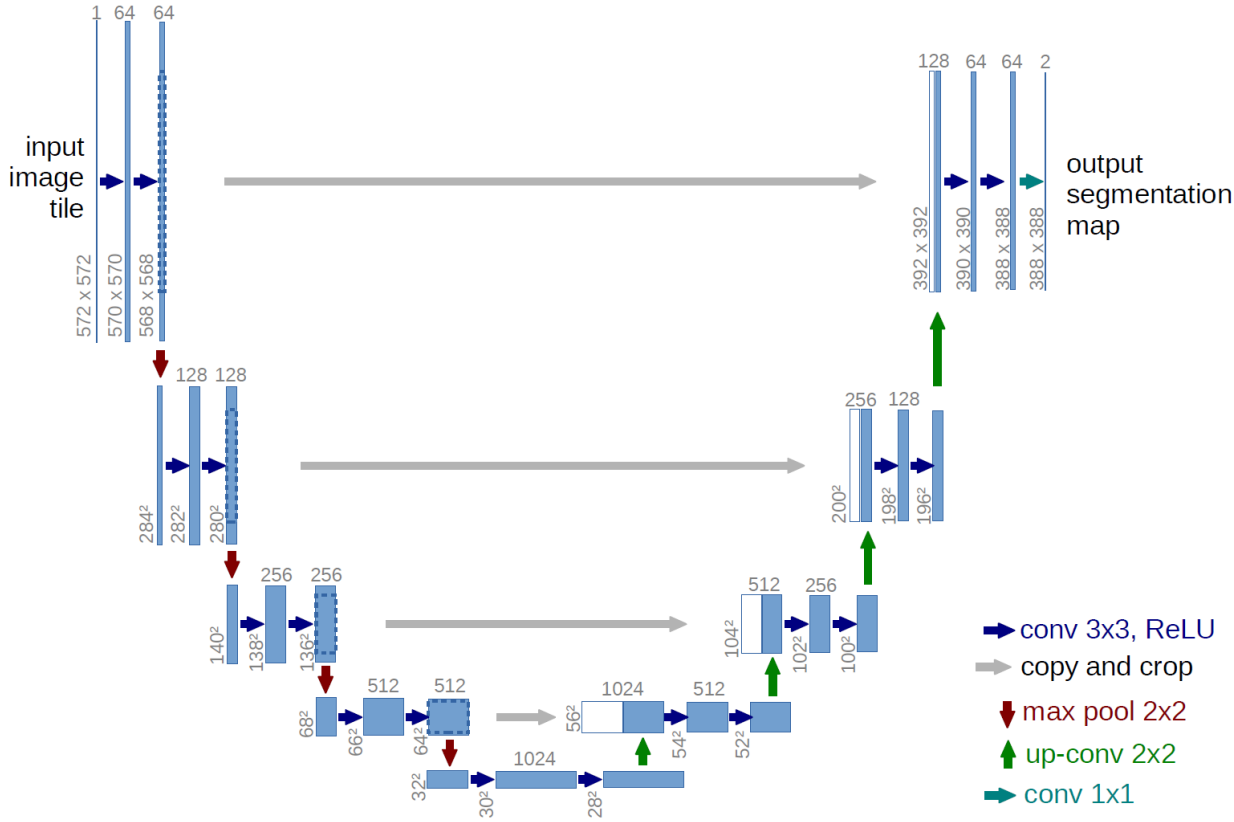
where nb_samples correspond to the total number of images (or samples), and rows, columns, and channels correspond to the number of rows, columns, and channels for each image, respectively.

For this problem who need to apply Instance based segmentation approach also called simultaneous detection and segmentation. If you use normal convolution approach we need more processing and training time for object detection and a separate method to segment each instance but if you use modified U-net model, you can run your input end to end at a time and there is no separate processing for identification and segmentation of instances.




From the above images the model is correctly segmenting two classes (person, horse) correctly using the instance based segmentation with accurate results. If we apply the same model to our problem, it will give accurate results and correctly identify the boundary of white blood cells in images even though the cells are touching with each other for counting purpose.

We are using the same approach using modified U-net model architecture for getting accurate results.




The contracting path follows the typical architecture of a convolutional network. It consists of the repeated application of two 3x3 convolutions (unpadded convolutions), each followed by a rectified linear unit (ReLU) and a 2x2 max pooling operation with stride 2 for down sampling. At each down sampling step, we double the number of feature channels.

Every step in the expansive path consists of an up sampling of the feature map followed by a 2x2 convolution (\up-convolution") that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU. The cropping is necessary due to the loss of border pixels in every convolution.

## Refinement

Initially I just used U-net architecture the accuracy is not great compared to the bench mark model. So I have used modified U-net with dropout.

Most of the time dropout is applied to fully connected layers compared to convolutional layers. since the convolutional layers don’t have a lot of parameters, overfitting is not a problem and

therefore dropout would not have much effect. However, the additional gain in performance obtained by adding dropout in the convolutional layers (3.02% to 2.55%) is worth noting. Dropout in the lower layers still helps because it provides noisy inputs for the higher fully connected layers which prevents them from overfitting.

The modified model follows the same pattern of U-net but with adding drop-out and augmenting on rotation and flipping original images. So that the model able to see more data and well generalize and invariance to distortions in images.

# Results

## Model Evaluation and Validation

We have used the given training set (both images and masks X_train, Y_train) to fit to our model and used dice coefficient as a metric to identify the accuracy of the model.

Finally, we applied trained model using dice score as metric to predict masks for X_train and compare those masks with Y_train. I mean with y_pred with y_true.

The model achieved state of the art results with accuracy of 94% and identified correct positions of white blood cells.

The model achieved better score compare to bench mark model I have chosen before and it can be applied to other segmentation tasks also.

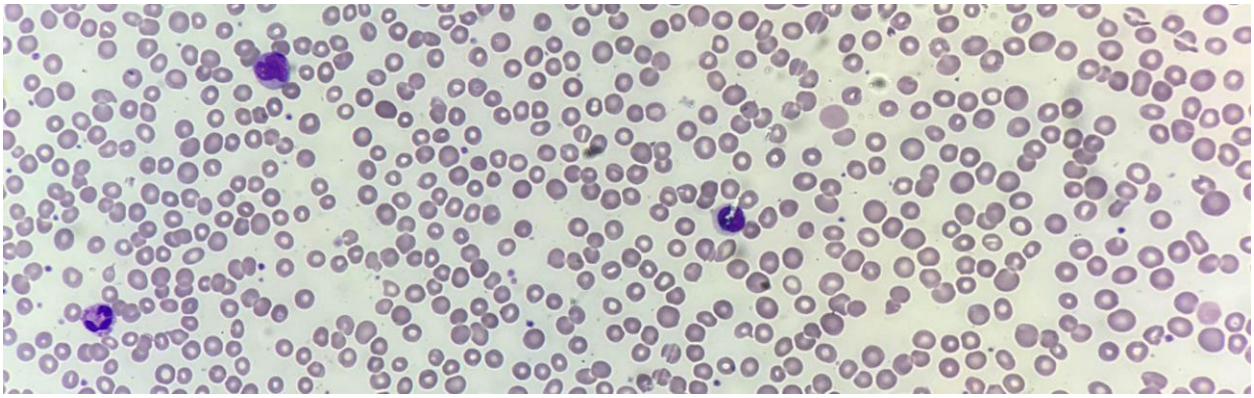
## Justification

The model correctly demarcates the boundary of white blood cells irrespective of its size and count.

All the images in test set are accurately demarcated with this model above 94%. And this accuracy is far better compared to the bench mark model.

# Conclusion

## Free-Form Visualization



## Reflection

The process used for this project can be summarized using the following steps:

1. An initial problem and relevant, public datasets were found
2. The data was downloaded and preprocessed
3. A benchmark was created for the classifier
4. The classifier was trained using the data
5. The classified predicted masks for test set
6. Validated the classifier with y_true and y_pred

I found steps 4 is very interesting. The original model(U-net) is not returning best accuracy compared to bench mark result. So I tweaked a bit by adding dropout and resultant model given good accuracy results.

I am so glad I found a good model(U-net) to solve this problem other it would take lot of time to come up with best model and needed a multiple gpu powered machines.

## Improvement

From the validation results and predicted masks, the model outperforms the bench mark model and yield good results.

On aws gpu machine each epoch of training time takes less than 5 sec and completed training in less than 5 min.

If we use multiple gpus and develop this model entirely on tensorflow back end, we will serve thousands of requests at a time and serve practical applications like identifying cancer or heart attack or retina defective cells for accurate results.

I would like to port this model on android and use tensor flow as back end for production.