An Integrated Neural Network with Edge Detection as a Fourth Channel for Similar and Overlapping Objects in Computer Vision

Kristina Mishra

A report

submitted in partial fulfillment of the

requirements of the degree of

Master of Business Analytics

Iowa State University

Year 2021

Project Committee:

Sree Nilakanta, Chair

# **Abstract**

One of the current most prominent topics in computer vision is medical image processing. This project used microscopic peripheral blood cell images from the Hospital Clinic of Barcelona to test if adding an edge as a fourth channel to an image would improve results. The dataset [1] was initially made publicly available with its own research for classification [15] to allow others researchers to add to their contribution. Similar research with malaria blood samples [3,11] shows that the need to have high accuracy in the classification of blood cells is still very prominent. Achieving high accuracy is not only important but also developing methods that improve accuracy as well as other evaluation metrics. Overlap in cells is problematic, and using the novel idea of adding an edge as a fourth channel can help address this ongoing issue in computer vision.

To add edge as a fourth channel to an image, the author of this paper built three models to leverage convolutional neural networks. The first model was SegNet and was used for baseline results. The second model was Holistically-Nested Edge Detection (HED) and was used to create the edge predictions for each image. The third model combined the edge predictions from HED as a fourth channel to images that were then processed with the SegNet model (referred to as SegNet + Edge).

The key aspects of this project included data cleansing, image manipulation and generation, and building three neural networks: two for comparison and one that was in tandem with one of the others. It also included standard classification techniques, including appropriate model splitting for training, testing, and validation sets and choosing appropriate metrics and visualizations to analyze and communicate results. On the non-technical side, but no less vital, additional skills used were project planning, problem evaluation, and communication with others for feedback.

The project results showed a significant difference in the Integrated SegNet + Edge model compared to the baseline model. IoU was chosen as the evaluation metric once it was determined that the accuracy metric was not a good fit. The overall results with most of the classes were very poor with the testing of different models. However, the final model did show some remarkable improvement between the base model and the integrated model with a mean IoU improvement of almost 6% - exceeding one of the aspirational goals (5%) of the project. The Neutrophil class stood out between the SegNet Model (13.8273%) and the SegNet + Edge Model (44.5503%). IoU differences between the classes on the final model for the test set ranged between 6.127 and 30.723 percentage points for the eight original cell classes, with some decreasing and some increasing.

Another aspirational goal of this project was to create a synthetic dataset. This goal became a high priority when it was discovered that the original base dataset produced disappointing results. Adding a synthetic set of 1800 images improved the previously low results, but it also drastically reduced previously excellent results to almost 0. This change was attributed to the class imbalances found through the dataset. Some results stood out with improvement using the new synthetic dataset merged with the original base dataset. For example, eosinophil changed from 0.0869% to 43.2296% in the SegNet test set and 0.0898% to 32.594% in the SegNet + Edge test set.

A key limitation of this project was that not enough time remained to create and test a larger synthetic dataset to achieve more definitive results. A larger dataset could have also addressed better the question of overlapping objects. Another synthetic set should be created that targets creating a balanced dataset with more variation and produces at least 5000 images. It is easy to create a much larger synthetic dataset, but the training time may be prohibitive; thus, 5000 is a reasonable number and comparable to a synthetic set created in similar research [15]. Training time for SegNet and SegNet + Edge usually ran between 2 and 3 hours but could be run parallel. (Any tuning to the HED model had to be run prior to running the SegNet + Edge model.)

Besides the initial low results, the biggest surprise revolved around image work (data preparation and cleansing). The initial images were not annotated as thought, and the process to do so was highly time-consuming. Additionally, preprocessing of images had to occur as well as creating ground truth images – all of which were not a part of the original project plan.

Another surprise was that models tested at the beginning of the project with another dataset did not work initially with this project dataset. This problem resulted in much debugging and even an attempt at building a different model. The problems were resolved, but all of the unexpected laborious tasks had the severe potential for putting the project at risk.

This paper details the process and findings of the proposed idea and is the culmination of the work done by the author in the Iowa State University Master of Business Analytics program.

# **Introduction**

Computer Vision is the field of study that trains computers to interpret the visual world. Recent advancements in computer vision offer opportunities to address problems that affect the global community: from medical to agriculture to improved transportation safety. Discoveries done in one area can be used across disciplines and as building blocks toward new ideas. Of particular interest is the problem of overlapping similar objects, as this can be difficult for computers to classify correctly.

This project proposes to design a framework that uses computer vision to identify and classify specific types of blood cells. The framework consists of three neural networks: a baseline neural network, an edge detection neural network, and an integrated neural network (baseline network combined with the edge network's output as a fourth channel). The main objective is to determine if this will improve semantic segmentation object detection that can be cross-applied to multiple fields for quickly identifying similar/overlapping objects.

A secondary goal of this project is to use lower-cost resources than is typically available for such projects. The purpose of this goal is to create a reproducible process at a low-cost entry point for organizations, non-profits, government organizations, researchers, and other stakeholders requiring minimal cost expenditures for developing similar projects or adapting any portion of this project. As such, the process is listed with end-to-end details, and a link to all associated code is in the index.

The beginning of this paper focuses on the problem and opportunity addressed by this project, identifying the target audience, the value proposition this research brings, and describing the goals. Following the introduction is a summary discussion of related research and this project's applicability to the Iowa State University's Master of Business Analytics program. The remainder of the paper is organized by illustrating the overall concept and tasks, a discussion of the project limitations, costs, the individual architectures, datasets used, and detailing the methodology used. The results are displayed with visualizations and discussed, closing with the conclusions and references cited. A small index is included for miscellaneous items, including key terms and definitions.

## Problem/Opportunity

In computer vision, objects with similar appearances, shapes, and overlap are problematic in object detection with neural networks. Many solutions have been proposed, with varying degrees of success, but there is still no silver bullet, and many current solutions are not practical to deliver efficiently. Efficient use of resources is of significant concern with global healthcare and global agriculture projects, where resources may be limited. This problem creates an opportunity to determine if a lower resource solution, using concepts with Edge Detection, can increase accuracy rates resulting in a potential benefit to the global community.

## Target Audience

Stakeholders that can benefit or may have interest from part or whole of this project include laboratory professionals, agricultural companies, healthcare informatics companies, researchers, health professionals, data scientists, students, small companies newly entering the computer vision space, and the Iowa State University Master of Business Analytics Capstone Committee.

This project can help stakeholders interested in identifying and classifying similar and potentially overlapping objects, from the beginning stages of annotation images to using an integrated network with edge detection as an image's fourth channel. Examples of how to use the results of this project include implementing a deep learning solution for identifying cell images, agricultural projects where images may have many of the same/similar objects and may overlap (e.g., rice, maize), and identification of coins in images filled with multiple denominations and origins. Furthermore, this project provides a lower-cost solution for stakeholders that may have barriers to using much-needed deep learning solutions. Having a staff that manually identifies problematic cells based on evaluating each image may be more error-prone and more expensive than loading thousands of images into a low-cost cloud account that flags images that meet a criterion for further evaluation: without needing lots of the expensive IT architectures often required to run multiple resource-intensive neural networks. This project can benefit anyone interested in the computer vision problem of identifying and classifying similar and potentially overlapping objects.

## Value Proposition

Creating an integrated neural network that uses edge detection as a fourth channel with a lower cost of resources may help stakeholders identify if this technology is an avenue to incorporate into their platform. Identifying if such an integrated network can help projects that range in size and scope and will allow stakeholders to choose what may benefit their individual needs without committing or using considerable resources. Agencies that may wish to incorporate similar technologies as part of the solution to a more extensive project can use this information as part of a whole or forego it all together (and not expend any resources). If the framework does not show an accuracy or improvement level needed, organizations can use resources elsewhere.

## Domain/Focus

### *Definition of target*

Develop, train, test, tune, and record and evaluate findings of neural networks (base, edge detection, and integrated) to detect and classify blood cells.

### *Goals*

This section defines the initial project goals for minimal, expected, and aspirational levels of success.

The minimum goals for this project relate to the development and testing of the proposed framework. The framework consists of creating three networks: a base neural network with SegNet, an edge detection neural network, and an integrated neural network (using edge as a 4th channel to images). Once these have been set up and tested, the goal is to see if adding an edge as a fourth channel to an image improves the model's accuracy. The model's accuracy is determined by comparing the loss and accuracy between the networks for each class.

In addition to setting up the framework, the expected goal is to test multiple splits of data to determine the best model. Also, given the density of two of the classes - mature red blood cells and platelets – it is expected that there will be an increase in accuracy for at least these classes from the baseline neural network.

Aspirational goals for this project are multifold and listed below:

* + Increase accuracy for all cells by 5%.
  + Use a more robust base network such as ResNet or U-Net.
  + Apply the model to other datasets using transfer learning.
  + Create a synthetic cell dataset to increase the number of images to train and test the model.
  + Implement a patch-based sliding window to increase the number of images and decrease image sizes for faster processing.

The evaluation metrics *overall accuracy* and *loss* determine the quality of each model. Additional accuracy metrics will be collected and reviewed for each object classified and the mean accuracy for all objects.

The quality of the project is defined by the successful execution of models and determining if adding edges as a fourth channel increases accuracy. Increases and decreases for accuracy for each class, total accuracy, loss, and mean accuracy will be evaluated.

# **Research**

## Similar Research

The Acute Lymphocytic Leukemia (ALL) Image Database for Image Processing (IDB) has similar research with blood cells (Scotti, 2005) in the goal that it seeks to find a way to automate the classification of specific blood cells as a low-cost, high accuracy solution. So similar is the idea that the ALL IDB was obtained and evaluated for this project (though later rejected). In that paper, Scotti presented a methodology to automate the classification of ALL from microscope blood-film images. The paper proposes a subsystem that performs feature extraction and classification and referencing a methodology for selecting sub-images of lymphocytes from a blood film image. However, the process described in the paper was particular to identifying Acute Lymphocytic Leukemia blood cells and may be challenging to adapt to other needs outside of blood cells. In our research, we want to identify every cell. The dataset was determined to not work for this project but potentially could be used to cross-test the results.

Malaria and parasite detection datasets and research provide another option for blood cell images. Poostchi *et al.* (2018) and Pattanaik, Swarnkar, and Sheet (2017) propose automated systems to detect and segment red blood cells (RBCs) and identify infected cells. In both papers, the systems first use detection and segmentation methods, followed by feature extraction and classification. Like the ALL IDB paper, these papers are directed towards specifically identifying if a condition exists or not, rather than identifying each cell. They incorporate a drill-down method (eliminating groups of cells that do not meet a qualification) to get to cells that need to be classified as having an infection or not.

In Marmanis *et al.* (2018), we see a pivot from traditional feature extraction methods to convolutional neural networks (CNN). Like this project, a simple model was used by combining SegNet with boundary (edge) detection. Marmanis goes further with including boundary detection in fully convolutional network (FCN) type models and a classifier ensemble. The method they described with FCN/ensemble models is computationally expensive and would not meet our criteria for having a low-cost, lower-computation solution. Additionally, their paper uses a two-branch architecture: a DSM channel (depth image) is trained separately from the original image, concatenated with the original image, and then fused with the edge network. The depth images are a crucial part of the base model that relies on aerial images and provides the network with additional important information. By combining many models and the depth images, high accuracy is achieved, but at the cost of higher computational requirements, complicated environments, and inability to determine the benefit of only adding an edge as a fourth channel.

In a paper related to this project's dataset (Acevedo *et al.*, 2019), the authors develop an end-to-end system for returning a high accuracy (96.2%) for eight of the blood cells targeted in the image. The accuracy of these eight classes in the best model had an accuracy between 91.8% and 99.61%. However, mature red blood cells are not counted as a class or considered with their method. Given that the majority of the overlapped cells were with mature red blood cells, the problem of overlapped objects is still left unknown. The paper briefly mentions creating an augmented dataset to enhance the existing one, but there is no indication that there is overlap within the augmented images, nor does the paper isolate the benefit of using synthetic datasets.

## Adapted systems

Marmanis provides a foundation for using a base network with edge detection. Given the conceptual similarity - though with much different object types, desired infrastructure, and goals - the base concept of SegNet (Badrinarayanan, Kendall and Cipolla, 2017) with a Holistically Edge Detection (HED) (Xie and Tu, 2015) network was adapted. Rather than using two branches, images are fused with a previously trained HED network (that used the same images), creating a matching 4th channel of edges for each image. This new image is then used to train the new integrated SegNet network.

An aspiring goal for this project was to create and use synthetic images. Bailo, Ham, and Shin (2019) conclude that combining synthetic data with real data can boost performance. Furthermore, Mayer *et al.* (2018) outline the importance of diversity, learning schedules, and the unimportance of realism when using synthetic datasets. As such, and using the process, code, and tutorial from Kelly (2019), synthetic datasets were created to augment the sparse set of annotated images available. In Kelly's work, the synthetic sets were created for instance segmentation, so code changes were needed to adapt to semantic segmentation.

## Value Proposition of This project Over Similar research

The previous section titled *Similar Research* reviewed some of the value propositions unique to this capstone project over the listed research. The value proposition includes the classification of multiple/all objects in an image [2, 3, 11], isolating the benefits of edge detection [5], isolating the benefits of using synthetic data sets, addressing overlapping objects [15], and designing an end-to-end architecture with low-cost resources. Unlike other similar projects, obtaining maximum accuracy is not a goal of this project but specifically addressing if adding a fourth channel of an edge will improve accuracy with any significance. An additional item of note is the encapsulating of independent processes should a stakeholder derive the value of any singular part of the process, particularly the annotation process that is not well documented and often confusing to those not familiar. Lastly, the accompanied code should add tremendous value to those that may be new to the computer vision space or may need to broaden their experience in any specific area of this project.

## Relationship to ISU MBAn

This capstone project draws from coursework from the Iowa State University Master of Business Analytics (MBAn) program. Starting with the initial MIS 547x course: Teams, Projects, and Business Analytics Leadership, project management skills were used to identify and segment the project into chunks and keep timelines on track. Courses MIS 536X (Business Analytics Foundation), and STAT 526x (Applied Statistical Modeling), were complementary courses that advanced knowledge in topics used in this project such as data preparation (learning common problems in data and how to resolve them), predictive analytics, strategies for model selection, and model averaging. These courses were building blocks to the more advanced coursework IE 583 (Knowledge Discovery and Data Mining) and MIS 546X (Advanced Business Analytics). These two courses were essential to this capstone project by delving into big data, classification, and neural networks.

Following along the lines of leadership skills, the MIS 551x course (IT Strategy & Execution) was fundamental in things such as planning, prioritizing, determining the value of tasks, and identifying meaningful metrics to determine success, all of which helped to organize the project, bring meaning to this report, and highlight important factors in the corresponding capstone presentation. The MKT 552x (Market Insights) course further honed consulting skills for this project to break the tasks into meaningful parts, determine the best course of action, and professionally communicate key takeaways. These skills are essential with the corresponding presentation. The knowledge gained from the course STAT 528x (Data Analytics and Visualization) prepared the author to give concise complex technical detail for a wide range of stakeholders with visualization techniques.

Prior to this capstone project, the final and most influential course to this capstone project was the IE 592 X course (Advanced Analytics Projects) from Fall semester 2020. This course involved a team of three graduate students working with an industry partner on a similar computer vision project. Details of similarities will not be listed here due to previous non-disclosure agreements to err on the side of caution. This capstone project is a continuation of technologies learned on that project, along with additional ideas for exploration.

# **Concept**

The proposed project uses object detection and classification. From a framework perspective, this must be divided into tasks that are categorized into groups. Figure 1 shows a high-level mapping of the tasks involved in each group. Groups are numbered as Step 1, Step 2, and Step 3, showing the required order. The gavel indicates potential decisions.

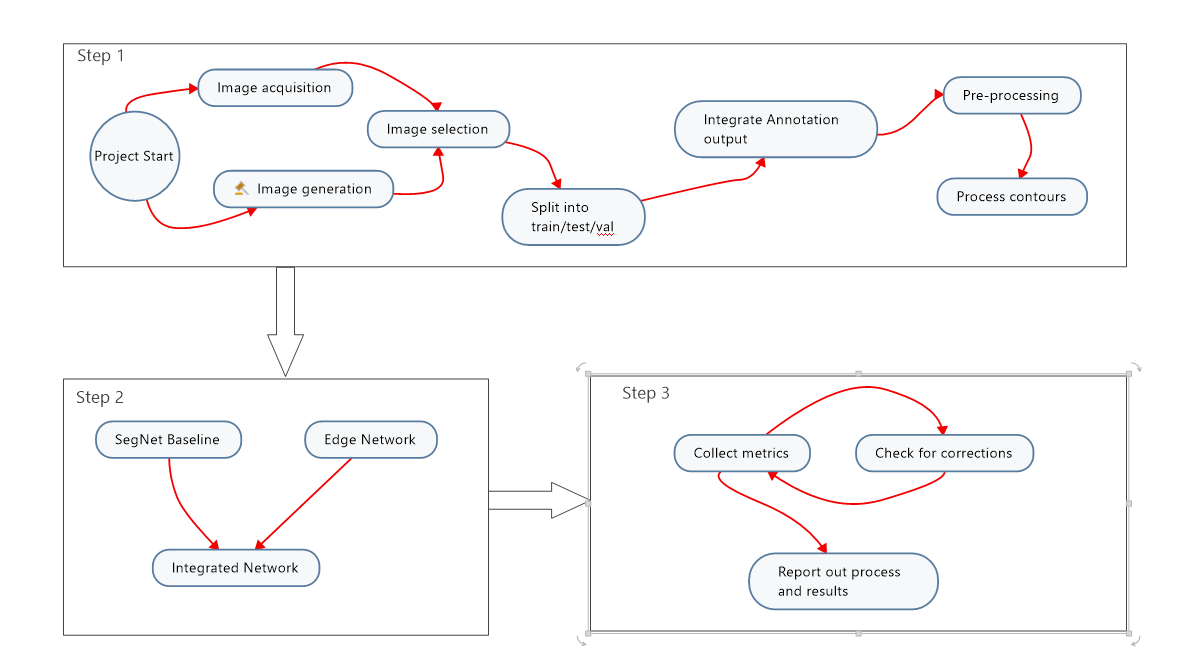


Figure : High-level concept diagram

# **Tasks**

## Value Effort Determination

Given the timeline constraints inherent to a semester, a review and prioritization of tasks were performed (Figure 2). Items marked with red arrows are required, gavels indicate decisions, and red flags mark risk. Note that the gavel icon indicates that a decision inherent to the task can change its effort and complexity. Green tasks with a checkmark note tasks that are chosen.

Items with low value initially were determined to have low or no priority. Data Tasks assessment was based on previous work and analysis of the dataset. Initial assessment of some of the Data Tasks (image processing and annotation tasks) was greatly underestimated for effort, causing project task selection to shift. The previous idea that a synthetic dataset was low value was changed.

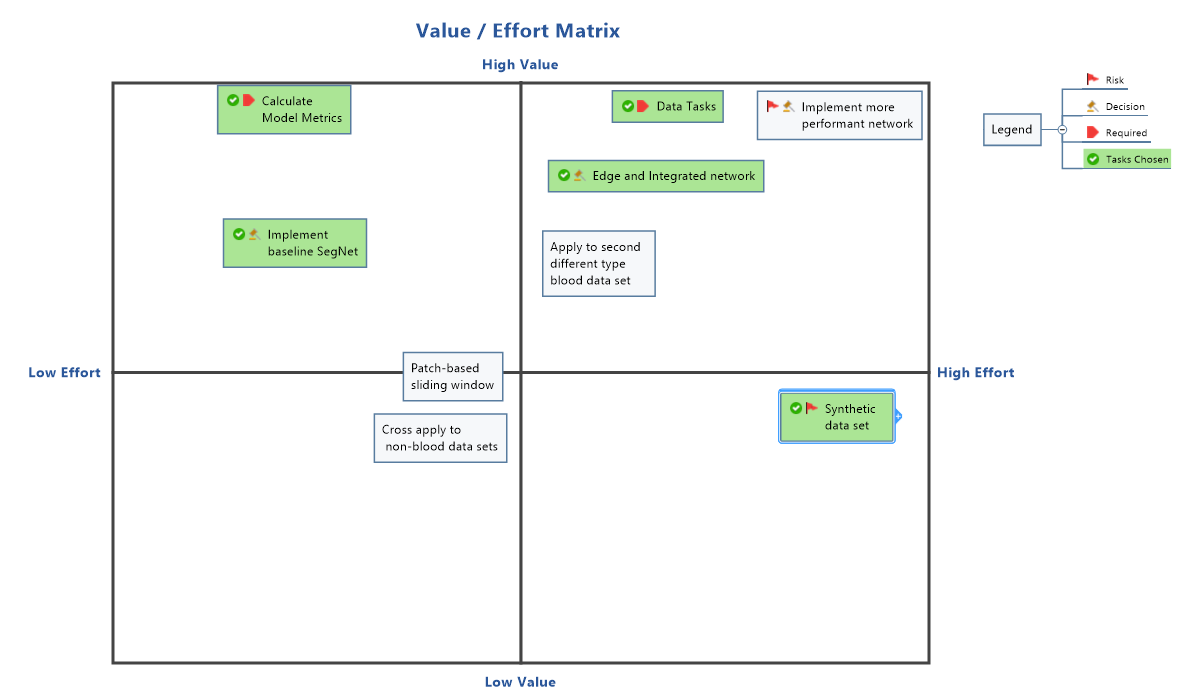
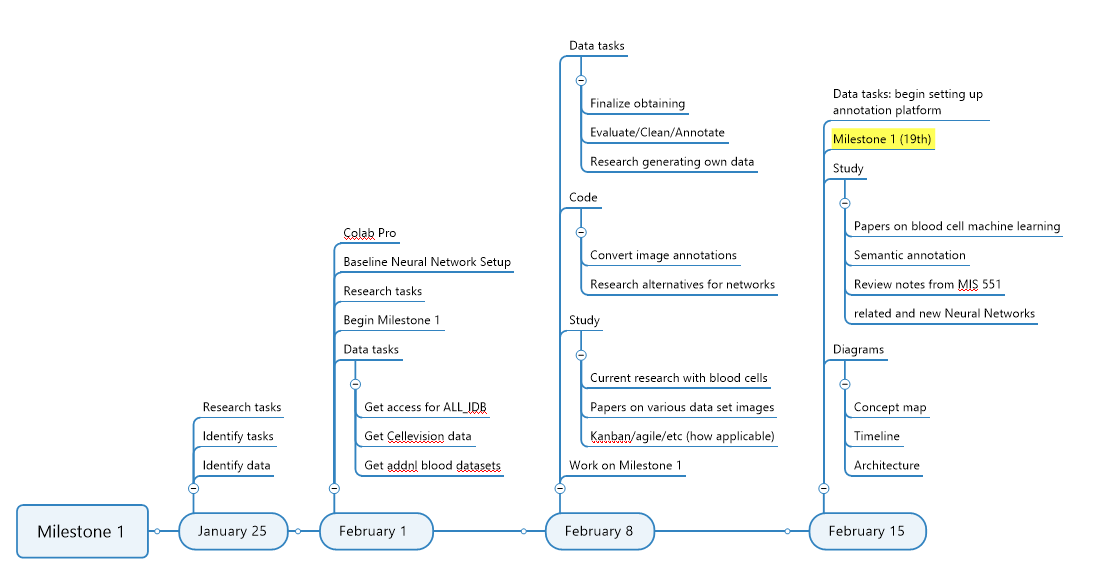


Figure : Value /Effort Matrix for task assessment

## Timeline

The initial project timeline for the summarized tasks is seen below in Figure 3. Items noted with the gavel were undecided at the time of project plan development. Items highlighted in yellow identify due dates for major deliverables.



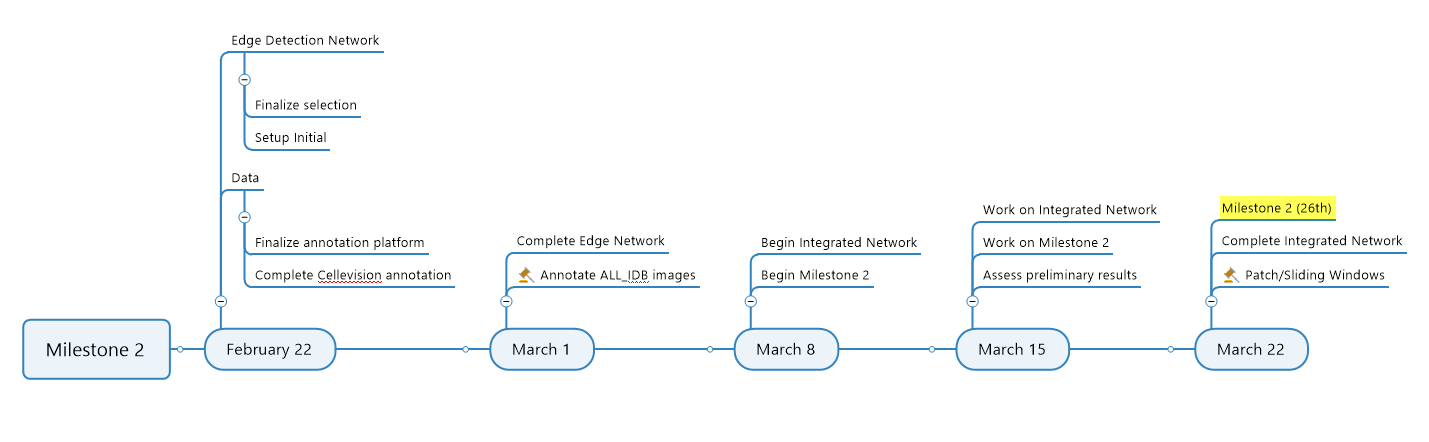






Figure : Initial Project timeline for milestones and final weeks

The initial timeline was considerably changed at the beginning of the project after identifying that the annotations provided with the dataset were not sufficient. This problem added an unexpected 1-2 weeks' worth of effort to find processes and software that could meet the project's requirements that had dependencies to the annotations. Hand annotation added another week. Unexpectedly, there was additional cleansing of the data tasks (creating code to label annotated images by class) and debugging the problem of image size.

After these tasks were completed and tested with the networks successfully, the metrics showed a significant problem resulting from not having enough images. A tutorial and course on creating a synthetic dataset (with corresponding annotations and ground truth images) and the adaption of the code added 1.5 weeks to the project.

# **Constraints, Resources, and Risks (Limitations)**

## Overall Project Limitations

A limitation of the model is the inability to recognize highly obscured cells. Overlapping cells cause problems with deep learning that may rely on feature attributes to determine the classification label. Particularly with regards to shape, but potentially other features as well. In the synthetic images created, there an observation that some images had a clustering of severely obscured/overlapping cells.

Another limitation of this project is that it does not address images with large sizes that could cause the current setup with Google Colab Pro to timeout. A patch-based sliding window with random cropped is suggested for projects with large image sizes. This method will also give the added benefit of increasing the size of the dataset [20].

## Key constraints

A key constraint of this project is the need for a GPU when training neural networks. For this constraint, Google Colab (Pro) was used with Google Drive to store images, code, input/output files, and results.

Additional constraints include annotations and images that were not in the needed format. Selection of software or coding for these tasks should be considered for any project using this or similar technology.

Lastly, for stakeholders wanting to adopt the architectures used in this project, image size considerations may be a constraint. Neural networks require specific image sizes that vary depending on multiple variables defined within the architecture (such as kernel or padding). Proposed image sizes should be tested prior to an entire run of the framework.

## Required Resources

This section outlines the required resources for this project:

* An image dataset with some format of annotation by experts.
* GPU for running the framework.
* Reasonable internet connection speeds (random testing of this project ranged from 12-19 Mbps for downloads and 3-6 Mbps for uploads).
* VGG Image Annotator (VIA) software (free and only requires web browser capabilities).
* Python IDE (Anaconda and Spyder, both free, was primary used in this project, but other IDE's may be used).
* Standard laptop with minimum 16GB of ram (see section *Costs and Equipment* below for specifications used in this project).
* Google account with Google Colab Pro and sufficient space available for storage of images files in Google Drive.
* GIMP software (free).
* MS office or compatible software with document, spreadsheet, and presentation capabilities. Google provides compatible options for free.
* Potentially needed: Additional annotation software to create ground truth images and contour JSON files.

## Risks

Identified risks for this project include time constraints, code issues, software incompatibility, internet problems, and potential timeouts with Google Colab. Time constraints are exceptionally high risk if the dataset is not annotated with files compatible with VIA or does not have ground truth images. In this case, a knowledgeable person with experience in annotating images should be consulted to determine the best course of action for the individual dataset or considerations for creating a synthetic dataset from the original dataset.

Frameworks are not plug-and-play when adapting code. Multiple variables (such as image size) can cause code to work with one dataset but not another. If attempting to use portions of this project, a person who is proficient in python is assumed.

Inherent to any IT project is the risk of software incompatibility. The python code in this project uses multiple libraries that often can run into issues based on the operating system, IDE(s), and other environmental variables. Best practices for setting up an environment are highly advised but outside the scope of this paper. Similar issues were encountered when various annotation software was tested for this project.

Lastly, internet stability and dependency on Google Colab Pro is an identified risk. While this project could be converted to use a standalone machine with a GPU or without Google Colab for stability, to do so is outside the scope of this project. Occasional timeouts with Colab were noted during the training of various networks if left unattended. Training with the largest dataset ran 2-4 hours. Temporary loss of internet can cause unforeseen timeouts, potentially losing the neural network training that may be occurring at the time.

# **Costs and Equipment**

Storage costs for Google for four months (two months for 200 GB, two months for 2 TB): $24.98

Google Colab Pro for four months: $39.96

Udemy course: Complete Guide to Creating COCO datasets: $32.09

Laptop specifications: Dell Inspiron 3780 – Intel® Core™ i5-8265U CPU @ 1.60GHz, 16 GB of RAM, 64-bit operating system, x64-based processor running Windows 10 Home Version 20H2, OS build 19042.928 (at the close of the project) purchased April 2019. Used for image annotation, python code related to images and debugging, running various annotation, mind mapping, Office, image manipulation, internet, and other common software products.

Potential costs: annotation software, developers, image annotators.

Variable costs: storage in google drive.

# **Architecture**

The project uses three network architectures: 1.) SegNet [8], 2.) Holistic-Nested Edge Detection (HED) [7], 3.) SegNet with an edge as a fourth channel (edge produced by HED). Figures 4-6 show models of each architecture.

SegNet (Figure 4) is the baseline neural network used for this project and the second-tier of the combined network architectures. SegNet was chosen due to its established performance with comparatively few tunable weights. The SegNet encoder-decoder architecture's simplicity provides useful results and allows the project to proceed faster, with less chance of running into problems related directly to the network. Additional networks ResNet and U-Net were considered a baseline network and for additional comparisons - as they generally provide better accuracy - but due to time constraints, were discontinued.

A downsample block was added in the input frame that uses a 3-channel (RGB) input, a 32-channel output, a kernel size of 3, a stride of 2, and padding set at (0,0) to speed up performance in our SegNet network [16]. Following the downsample block, there are 13 convolutional layers over 5 encoder stages, with max pooling at each stage. There are two convolutional layers with 64 output channels in the first block, with a kernel size of 3, a stride of 1, and padding of 1, followed by max pooling. While doing the 2x2 max pooling, the corresponding indices are stored. The next block again uses two convolutional layers with everything the same except the output channels is 128. The third, fourth, and fifth blocks have three convolutional layers with output channels 256, 256, and 512. The third layer of each of these uses a kernel size of 1, with a stride of 1, and 0 padding.

At the decoder, we flip the process: upsampling and convolutions are performed. During upsampling, the max pooling indices at the corresponding encoder layer are recalled to upsample, resulting in an output channel of 12 – representing each of the classes plus one for the 0 index we do not use. Lastly, the softmax classifier is used to predict the class for each pixel. The softmax activation function makes sure all the output values add up to exactly one. The idea is that each output is a value that represents the percent likelihood that a certain type of object was detected, and all values (1 for each class) should add up to 100 % or one.

A picture containing diagram

Description automatically generated

Figure 4: SegNet Neural Network Architecture

For the edge detection component of this project, the deep learning Holistically-Nested Edge Detection (HED) [7] model was used. HED is similar to SEGNET at its convolutional layers. The difference is that HED learns rich hierarchical representations using bilinear deconvolute for each layer and creates a prediction side output for each one. Each side-output layer is also associated with a classifier. At each level, there is an increase in the filter and stride, as seen in Figure 5. At the end, each prediction is concatenated with multi-scale weighted fusion, creating a final output image.

Diagram

Description automatically generated

Figure : HOLISTICALLY-NESTED EDGE DETECTION Network ARCHITECTURE

In the Integrated SegNet architecture, the HED network is incorporated at the beginning. The input image is feed to the HED model in inference mode to predict the edges. The predicted edges are then concatenated with the original image and then feed into SegNet (operating in train mode). After that, everything is similar to the basic SegNet except having an initial downsample input of 4 channels instead of 3. The additional channel is the newly fused edge layer. Figure 6 shows the integrated architecture. The HED and SegNet details are the same as those previously displayed in Figures 4 and 5 and are not duplicated in Figure 6.

Diagram

Description automatically generated

Figure 6: INTEGRATED NEURAL NETWORK ARCHITECTURE

# **Datasets**

## Primary Dataset

The primary dataset (PBC\_dataset\_normal\_DIB [1]) is a subset of the microscopic peripheral blood cell images made publicly available through the collaboration of The Clinic Hospital of Barcelona, The Technical University of Catalonia, and the University del Rosario, Bogota. (Acevedo *et al.*, 2020). Images were obtained using the analyzer CellaVision DM96 in the Core Laboratory at the Hospital Clinic of Barcelona and consist of red blood cells (RBC), white blood cells (WBC), and platelets. The dataset was annotated by expert clinical pathologists and organized into eight groups: neutrophils (WBC), eosinophils (WBC), basophils (WBC), lymphocytes (WBC), monocytes (WBC), immature granulocytes (promyelocytes, myelocytes, and metamyelocytes: all WBC), erythroblasts (RBC) and platelets (thrombocytes). Mature red blood cells are not separately grouped in the dataset but are easily identified, as noted in figure 7. Images were captured from individuals without infection, hematologic or oncologic disease, and free of any pharmacologic treatment at the moment of blood collection.

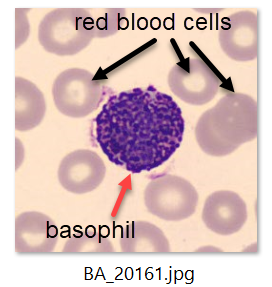


Figure : Image BA\_20161.jpg showing basophil and red blood cells

The full dataset consists of 17,092 images of the size 360 x 363 pixels, with a small subset of images having a size difference of fewer than 10 pixels on either side or both (example: 366 x 369 pixels). For the base dataset, a total 114 images were used from each of the eight groups to further annotate in a format usable by the neural networks.

## Synthetic data sets

Model results of the base dataset revealed a need for more images. Manual annotation of images is an extremely slow process: in this project, 1 single image averaged between 5 and 10 minutes using the fast-painting solution SuperAnnotate [17]. Other annotation solutions that were tried where the user draws each contour manually took between 20-40 minutes. For this reason, the author of this paper decided to try and find a way to create a synthetic dataset that would automatically generate corresponding annotation and ground truth files. Reviewing the work of Kelly [9,10], it was determined that his method would meet the requirements and produce the fastest set, as well as create an easily reproducible process that could be used with any dataset.

A synthetic dataset consisting of 1800 images was created to overcome the sparse base dataset. 219 new images were selected from the original full dataset: 196 for foreground images and 23 for background images (1 image having multiple foregrounds). Immersion software [9] provides free code in GitHub and a general outline of creating a synthetic data set. Kelly and Kelly [10] also have an accompanying Udemy course that explains each step of the process/code.

# **Methodology**

## Method Summary

A computer vision project comprises multiple steps: acquiring data, data preparation, organizing of data, supporting file generation, creating test use cases, creating metadata of the dataset(s), running of the models, recording and analyzing results, and constant debugging throughout. In this project, image/data work and debugging took approximately 60-70% of the total project time. This section highlights the order of steps to recreate this project with other datasets. Figure 23 in the index has a simplified flowchart for the steps below.

The first step is acquiring the data (section *Datasets*). Once this is complete, images from the original dataset must be selected to create a primary dataset (section *Initial Image Set*) or used to create a synthetic dataset (section *Synthetic Image Creation*). In this project, both were done, creating two datasets that were merged.

Synthetic images were created to increase the number of images available and to increase learning capabilities [16]. A subset of images from the original dataset was used as an initial base dataset of images (dataset 1) for the neural networks, and then synthetic images (dataset 2) were generated to supplement the low count of many object categories. The two sets were merged, creating a new image set (dataset 3) to use as a comparison against the baseline set of images.

All images must be of a size that will work with the specifications of the network for downsampling and upsampling to work correctly. The section *Image Size Requirements* details problems with the original image size and what was done to correct it. During this project, it was discovered that additional images that were generated had to be corrected for size, but this can be avoided if all the images that will be used are selected from the beginning and the size correction is applied to the new dataset created.

Additional image tasks need to occur following image selection and correction: annotation tasks and the generation of binary contour images. Annotation tasks are detailed in the section *Annotations* and section *Additional Annotation Tasks*. Since this project uses an edge detection model, binary contour images (section Contour Images) must be generated that correspond with each dataset image once the annotations are complete. The contour images are required for manually annotated datasets, as well as synthetic datasets.

Computer vision classification problems are like other classification problems in that the dataset is split into a train, test, and validation set. Organizing the images into different model splits can occur anytime after image selection and before training of the models, but the advantage of putting it after annotation and binary contour image generation is that if any image problems occur (that may need to be removed from the set), the model splits will not need to be recalculated.

The model splits are stored in the file image\_metadata.xlsx. After the previous tasks are done, a minimum of image names and the image assignment to train/test/validation must be completed and uploaded. Additional information included in this file includes image names, object information for images, model splits, metadata and statistics of the different classes, definitions and examples, statistics for annotated images and synthetic images, and foreground/dataset image class breakdowns. The models use only the main image tab.

A separate image\_metadata.xlsx was created for test use cases and replaced with the full set once all tests were completed. Each network model can range in time depending on the number of images, batch size (image size if changing), and the number of epochs – so test cases should be implemented. Tests ranged from 1 hour to around 6 hours (more in cases where there were internet problems).

The following model splits were tested for both dataset 1 and dataset 3: 80/10/10 (model 1) and 70/15/15 (model 2). A 60/20/20 split was rejected as the number of object images was low, and there was a noticeable drop in accuracy from model 1 to model 2. More information detailing train/test/validation splits is in the section *Training/Testing/Validation* under the section heading *Commonalities*. All model splits are stored on individual tabs in the excel file: image\_metadata.xlsx and are used by each model.

For each model split, SegNet and HED were run in parallel. Results were observed, recorded, and analyzed. After both were completed, the Integrated network was run, with results recorded and re-evaluated. Details of training parameters, epochs, and network implementation stages are defined in section *Commonalities*. Results for the model selected are listed in the section *Result.*

## Images

### *Initial Image Set*

Images in the primary dataset contain mature red blood cells (RBC) and one of the eight-cell types centered in the middle of the image (Figure 2). Some of the images in the dataset contain more than one of the same cell types, a different cell type, small platelets, or various small artifacts. Images with different cell types (excepting small platelets and mature RBCs) were not included in subsets used for this project as the additional cells were not separately annotated. Images were all prefixed according to category and are listed in Table 1.

|  |  |
| --- | --- |
| **Folder Name** | **Image name prefix** |
| basophil | BA\_ |
| eosinophil | EO\_ |
| erythroblast | ERB\_ |
| ig | IG\_, MMY\_, MY\_, PMY\_ |
| lymphocyte | LY\_ |
| monocyte | MO\_ |
| neutrophil | BNE\_, NEUTROPHIL\_, SNE\_ |
| platelet | PLATELET\_ |

Table : Annotation FOlders for DATASET

The base dataset images contain an average of 16 objects per image, not including background. Figure 8 shows the disbursement of the cells. Mature RBCs are not separately annotated in the original dataset but have been classified here in 1 of two groups: RBC (for non-overlapping or non-edge RBCs) and RBC\_ol (for overlapping or on the edge of an image RBCs). It is noted in Figure 8 that the dataset is heavily unbalanced by the cells in each image.

The dataset has an excess of 85% RBCs (Figure 9), but since there is a higher number of desired overlap of the objects in the images specifically within RBCs, and because these are real-world images, the unbalance set was still used. (Note: Figure 8 separates erythroblasts from the RBC and RBC\_ol categories as they are classified differently. Erythroblasts are included in the RBC count in Figure 9.) Platelets also presented unbalance of objects as frequently multiple small platelets were in images for the other seven cell categories. Since the categorized images had more prominent (and clearer) examples of platelets, those were included as well.

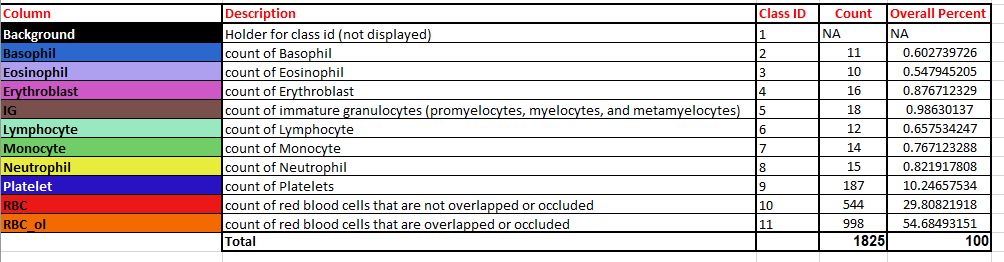


Figure : OBJECT CATEGORY BREAKDOWN FOR BASE DATASET

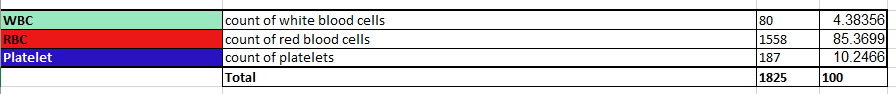


Figure : Type of CeLL breakdown for Base dataset

Images were pre-selected for annotation, producing ground truth images and a JSON file detailing each object's polygon contour. Each image was assigned to a train/test/validation group for the model splits of 80/10/10 (model 1), 70/15/15 (model 2), and 60/20/20 (model 3). The dispersion of base image subset counts is listed below in Table 2. An example of an image and corresponding ground truth image is provided in Figure 11.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **split** | **num images** | **act train** | **act test** | **act val** |
| 80/10/10 | 114 | 90 | 12 | 12 |
| 70/15/15 | 114 | 80 | 17 | 17 |
| 60/20/20 | 114 | 68 | 23 | 23 |

Table : Count of images in each set

### *Image Size Requirements*

Different Neural Networks and variables such as number/size of layers, padding, and kernel size, may change the requirements of the size of an image. It was determined in the testing of images with the neural networks that the image size of 360 W x 363 H would not work with our models' upsampling/downsampling processes. The upsample array sizes need to match the downsample array sizes. After testing different sizes with changes in variables, it was determined that 361 W x 363 H was the ideal size that would minimize changes to the original image and still process through the network. A helper python file was created to copy the pixels along one edge, resulting in the new desired size to extend the width in this manner. By doing this to the original images, all ground truth, annotations, and contour images end up being the correct length, and no information is lost.

### *Annotations*

The original format of annotations provided was in the form of organizing images into folder names of their respective categories, producing eight categories. Since this annotation method is not usable by a neural network, additional exploration of annotating the images was required. In a precursor computer vision project created for course IE 592x, images came pre-annotated by experts in the field. In Kaggle competitions, tutorials, and publicly available computer-visions datasets, this is also frequently the case. After reviewing multiple datasets and some of the corresponding annotations, it became clear that there is no standardized or universal format used for annotating images.

For this project, the requirement was to have images with every object in the image annotated in JSON format and a corresponding ground truth image (Figure 11). Some common annotation formats are JSON, XML, TXT, and CSV. To further complicate the process, the formats can be proprietary and can vary between applications. For example, in a given dataset, a JSON annotation may be in formats COCO, SuperAnnotate, Labelbox, VGG Image Annotator (VIA), or even generic one as an output created by code. Because of a process and knowledge obtained in a prior course (where the annotation JSON files create contour binary images), this project specifically required the format to be the JSON format provided as export by the open-source software VGG Image Annotator (VIA) [14]. However, VIA does not create ground truth images and is slow to use for annotation, so an additional solution was required.

Multiple software and coding options were explored and tested, with difficulty in finding the right solution for the models that had already been designed and tested with a known set of unrelated pre-annotated images. After much time spent annotating, coding, debugging, and testing the outcomes with the multiple architectures in this project, two products were decided upon: VGG Image Annotator (free) and SuperAnnotate (typically a paid-for product, but free for academic use).

VIA is a simple annotation program that only requires a web browser to use. For this project, separate batches of images were imported, and each batch was saved as a project. Annotating images often refers to identifying each object in an image and labeling it to its corresponding class. Figure 10 shows an example of the contours created and the selected object assigned to the RBC class. Once the annotations were complete, they were exported into the required JSON file used downstream in the process outlined in the section of this paper labeled: *Contour Images*.

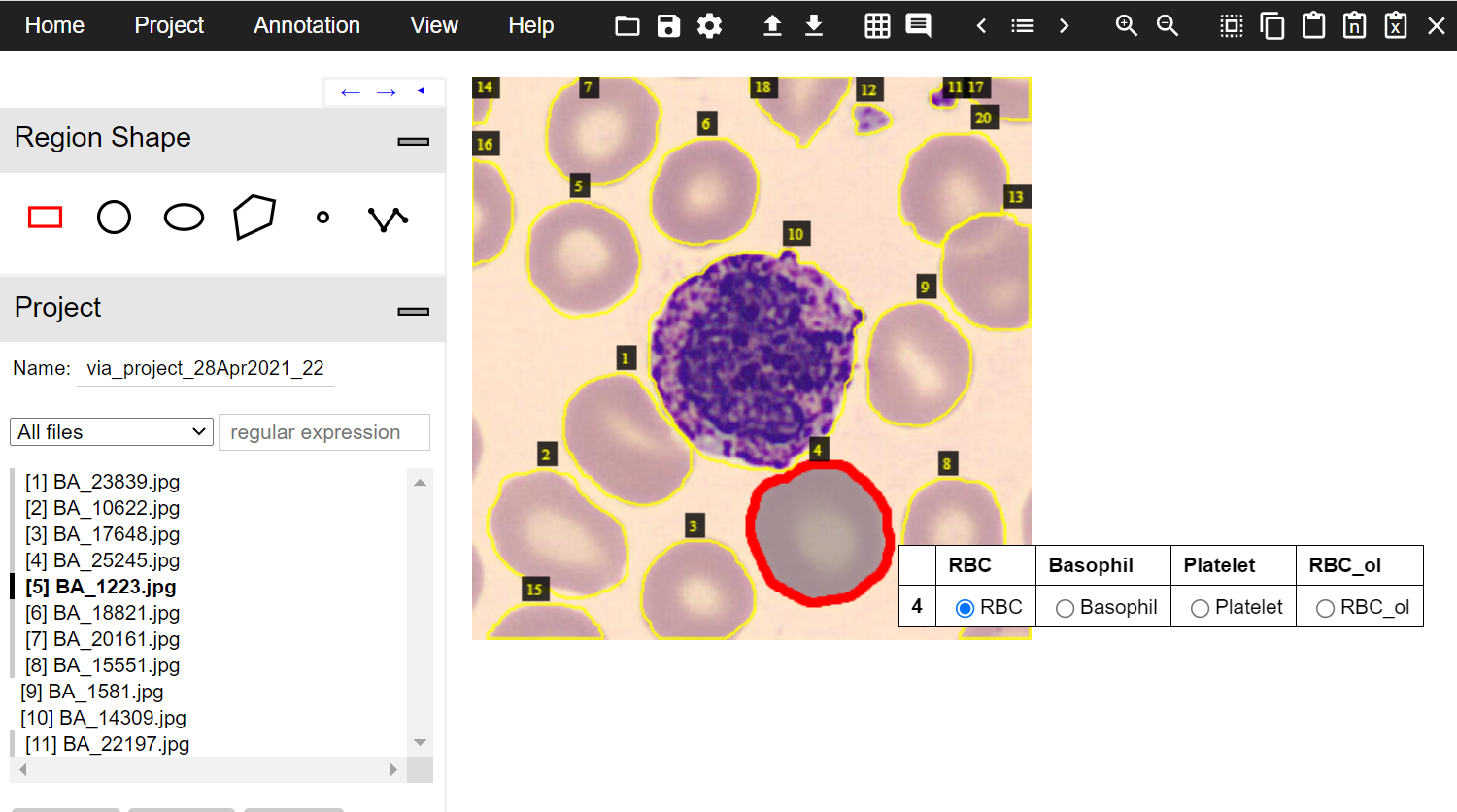


Figure : Example of annotation in VGG Image Annotator

The other part of the annotations process is creating a ground truth image that mimics the outlines detailed in the exported JSON file. A ground truth image is an image where each pixel has been labeled as a specific category. In the case of the ground truth image in Figure 11, there are five things labeled: 1.) background (purple), 2.) small platelet (tiny medium green), 3.) basophil (dark purple), 4.) mature red blood cells (light green), and 5.) mature red blood cells that either overlap or on the edge of the image (yellow).

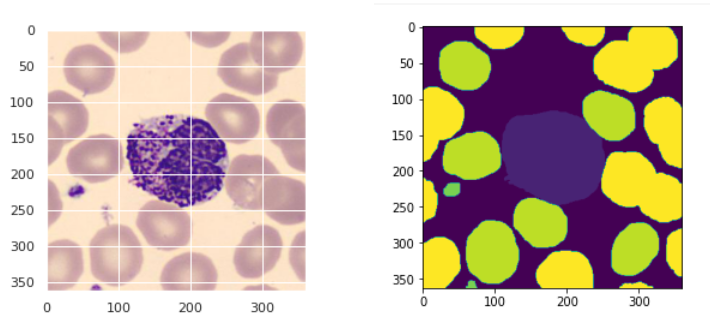


Figure : Example image with corresponding ground Truth image

VIA does not generate ground truth images, and the process to hand-annotate each object is very slow. To overcome these issues, SuperAnnotate was used. SuperAnnotate (web version) allowed this author to reduce the time for annotating an image to one-third (with painting an image instead of outlining), export the annotations in COCO JSON format, and generate ground truth images. Most products would not produce the latter, which was required and information on creating these in code was not readily available. Even so, it was later discovered that the ground truth images had a different format than expected, and additional coding was required to convert them. This process provided useful information later when the same problem was realized with the synthetic image generation. Once the images were annotated in SuperAnnotate, the exported COCO JSON file could be pulled into VIA, annotations checked, and then exported to the required VIA JSON format.

Many open-sournce products can be used to annotate images, but many of them have a large learning curve and complicated installation procedures. An inspection and conversion to the VIA JSON file format may be easier for users with annotation experience outside the steps listed in this paper. If using synthetic datasets only, SuperAnnotate is not needed as the ground truth images are created via code adapted from Kelly [9,10], available in the GitHub repository noted in the index.

### *Additional annotation tasks*

Ground truth images are not all in the same format. In the author's research and testing, PNG and TIFF were the two most often used image types (because they allow for transparency), but there are additional considerations for semantic segmentation ground truth images. In machine learning terms, an image is an array that has at least a height and width dimension but often has at least one more dimension, such as RGB color, in where a 360 W x 363 H image would be an array of (363, 360, 3). In this example, the 3 represents a color dimension that gives a numerical value for Red, Green, and Blue for the size of the height and width. For example, if a 5x5 image has a 3x3 square with a red value of 255, a green value of 5, and a blue value of 100, you would have three 5x5 blocks with each value for R, G, and B as seen in Figure 12. By flattening the three blocks to one block, the array has a singular number representing each block in the 3x3 square. In this process, the square's number is set to have a class label of 2 and the shape of the array changes to (363,360).

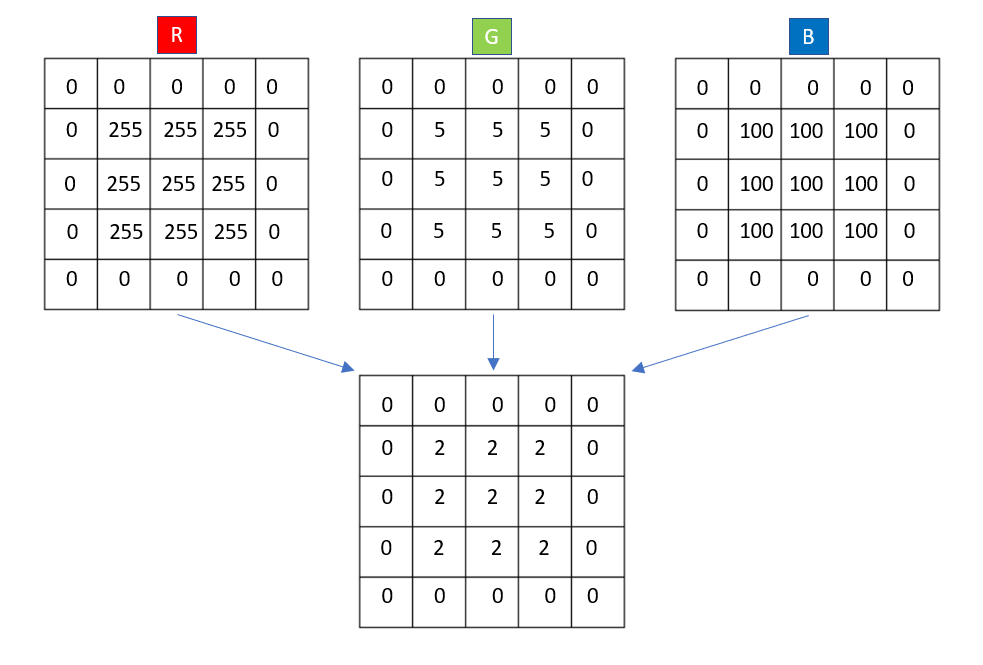


Figure : Example of RGB image array flattened to a labEL

In the case of the ground truth images created by SuperAnnotate, the array shape is (363,360,4) with all values on the 4th slice on axis 2 equaling 255 (black). By flattening the shape to (363,360) a singular value was given to denote the class of each object, and that was then converted to corresponding class label 1-11: 8 categories + 1 background + 2 types of red blood cells (see Figure 8 for class labels). The ground truth image is wholly preserved in this process and is consumed by the neural networks easier because it looks specifically for a class label. This process was done in python and took only a few minutes to run for all images in this project.

### *Synthetic Image Creation*

219 images were selected that had not been previously used in dataset 1 to create a synthetic dataset: 196 for foreground images and 23 for background images. Immersion software [9] provides free code in GitHub and a general outline of creating a synthetic data set. Kelly and Kelly [10] also have an accompanying Udemy course that explains each step of the process/code.

Initially, the first test set was produced quickly and examined for problems. An issue was discovered that the code was built for instance segmentation with Mask RCNN, instead of the project required semantic segmentation. This author built additional coding to adapt to the required semantic segmentation. This process was lengthier than expected but revealed many vital lessons on underlying python code and arrays. Additionally, Kelly's work helped this author to understand some interesting key information about images and creating masks, foregrounds, and backgrounds in GIMP.

The first step in creating a synthetic dataset is to create background and foreground images. The background images were created from sections of other images that had no cells present. To ensure bias was not introduced, foreground images were selected and organized into train/test/validation sets. No images were cross used between the different sets, and table 3 shows the foreground image count for each category and each train/test/validation set. Absent from these foregrounds were the RBC and RBC\_ol classes as the density of these in the original set was much higher. Using real images with synthetics images improves performance [4], thus no RBCs or RBC\_ols were used in the creation of synthetic data to ensure a more balanced representation of different cells between the datasets.

|  |  |  |  |
| --- | --- | --- | --- |
| **Cell type** | **train** | **test** | **val** |
| **Basophil** | 10 | 7 | 7 |
| **Eosinophil** | 10 | 7 | 7 |
| **Erythroblast** | 10 | 7 | 7 |
| **Immature Granulocytes** | 11 | 7 | 7 |
| **Lymphocyte** | 10 | 7 | 7 |
| **Monocyte** | 10 | 7 | 7 |
| **Neutrophil** | 10 | 7 | 7 |
| **Platelet** | 12 | 8 | 7 |

Table : Image Count for Foregrounds

After the images for backgrounds and foregrounds were selected and separated, each batch was opened in GIMP [18] to create the foreground images. Figure 14 displays the steps seen below.

1. Draw a polygon with a slight buffer around the object.
2. Fill in polygon with a mask that goes as close to the edges as possible with slight feathering.
3. Preview foreground selection and add or subject as needed.
4. Select the foreground that is a preview so that the selection contour is at the outer edges of the object (hard to see in the image).
5. Create a new layer, hide the one, and paste the foreground selection in the new layer.
6. Crop layer to object and save in PNG format.

|  |  |
| --- | --- |
|  |  |
| Step 1: Draw polygon | Step 2: Fill in polygon with mask |
|  |  |
| Step 3: Use foreground selection to get the edges | Step 4: Show refined selection |
|  |  |
| Step 5: Create layer with selection | Step 6: Crop to object |
|  |  |
| Completed foreground image (transparent background) |  |

Figure : Process of creating foreground image

Once the background and foreground images were created, the file image\_composition.py was run on each of the train/test/validation sets, with the necessary parameters. These parameters include input/output directories, the number of images to create, and the size of regular/ground truth images. The file image\_composition.py chooses a random foreground image, applies random transformations (rotate, scale, and brightness) to the foreground image, and then randomly places it on a background image to create a new image. Padding is added to avoid placement that could bleed into edges. The number of foregrounds used was between 1 and 6 (randomly chosen) per image. Figure 15 shows an example of synthetic images and corresponding ground truth images created, where the desired effect of overlapping objects can be seen in some of the images. The ground truth images show the labeling of the classes by color, note in the image 00001250.png, the purple shows two of the same type of cells with overlap.

A picture containing text

Description automatically generated

Graphical user interface, application

Description automatically generated

Figure : Examples of Synthetic images generated

Along with the images created, additional files were created - including necessary files for the next step of the process - and stored in the output directory. After this was completed, the coco\_json\_utils.py file was run to create the corresponding annotations COCO JSON file. Once that was complete, the images and COCO JSON file were brought into VIA to verify contours, and a new JSON file was exported.

### *Contour Images*

For the HED network, contour images must be created for each original image. Contour images are binary images that display only the outer edge of the objects in an image. Figure 15 shows two examples of contour images used in this project. Code was reused that the author wrote in the previous semester that uses the annotation JSON file created from VIA and creates a binary contour image to create these images. The edge detection network then uses the contour image as a ground truth image. The model can learn the edges of the objects and create a new learned edge image (figure 16) used in the integrated SegNet + Edge network.

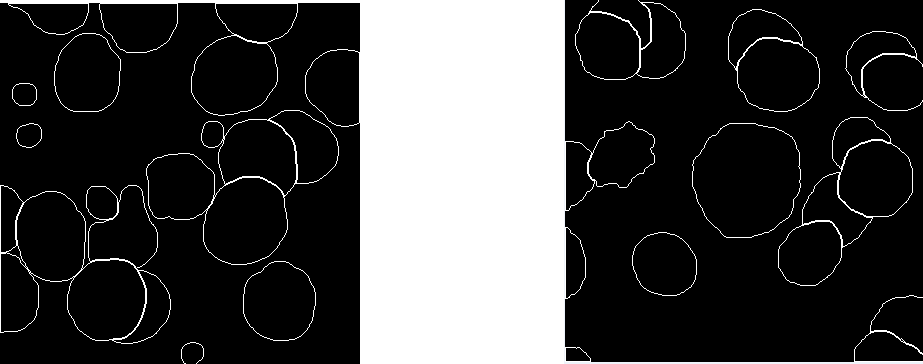


Figure : Example of Contour (Edge) Images

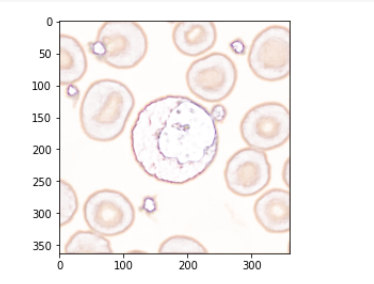


Figure : Example image produced from edge Network

## Commonalities

### *Initial Network Setup*

Initial steps for all networks involve reading the excel file *"image\_metadata.xlsx"* to retrieve image paths and create appropriate data splits for the train/test/validation sets. The class BloodCells is created to apply various manipulations to a blood cell dataset object.

### *Training/Testing/Validation*

The training and test set uses an image batch set (see Table 5 and 6 for batch sizes) from the datasets concatenated with the ground truth images. The validation set was not batched, and images were used with the same concatenation method as training/test. Train/Test/Validation sets were kept independent to ensure that bias was not introduced.

Additionally, synthetic image foregrounds were separated into train/test/validation sets prior to creating synthetic images to ensure no foreground image was used across train/test/validation sets or across models. In the case of synthetic images, 1800 total images were created, with 1200 reserved for the train set, 300 reserved for the test set, and 300 reserved for the validation set. 1500 images were used for each model, and no image was ever used in any other set. The train set had 1200 images to choose from, the test and validation set both had 300 to choose from. Table 4 shows the image count for each set in each model.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model split | total images | num images | train | test | val |
| 80/10/10 | 1800 | 1500 | 1200 | 150 | 150 |
| 70/15/15 | 1800 | 1500 | 1050 | 225 | 225 |
| 60/20/20 | 1800 | 1500 | 900 | 300 | 300 |

Table : Image count breakdown by model

### *Training Parameters*

Training parameters remained consistent across networks and image sets, except for batch size, as seen in Tables 5 and 6. Model split parameters remained the same across image sets and thus are not separated out.

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameters** | **Seg-Net** | **HED** | **Seg-Net + Edge** |
| **Batch Size** | 3 | 3 | 3 |
| **Learning Rate** | 0.0001 | 0.0001 | 0.0001 |
| **Epochs** | 155 | 155 | 155 |
| **Learning Rate Decay** | 0.1 after 100 epochs | 0.1 after 100 epochs | 0.1 after 100 epochs |
| **Optimizer** | Adam | Adam | Adam |
| **Momentum** | 0.9 & 0.999 | 0.9 & 0.999 | 0.9 & 0.999 |

Table : Tuning parameters for Base dataset images

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameters** | **Seg-Net** | **HED** | **Seg-Net + Edge** |
| **Batch Size** | 25 | 25 | 25 |
| **Learning Rate** | 0.0001 | 0.0001 | 0.0001 |
| **Epochs** | 155 | 155 | 155 |
| **Learning Rate Decay** | 0.1 after 100 epochs | 0.1 after 100 epochs | 0.1 after 100 epochs |
| **Optimizer** | Adam | Adam | Adam |
| **Momentum** | 0.9 & 0.999 | 0.9 & 0.999 | 0.9 & 0.999 |

Table : Tuning parameters for base dataset images plus synthetic image set

### *Epoch Verification*

Accuracy, loss, and IoU are checked at each epoch. After each epoch, the model and optimizer state (validation loss, accuracy, and IoUs) are saved to a corresponding directory at the root level. The minimum number of epochs was set at 150 to capture more data for analysis and visual displays. Selected epochs are later moved to an archive directory if they need to be reviewed again after other sets are run.

### *Final Checks*

Once training completes, the framework plots the validation loss, accuracy, and IoUs for each epoch. The final trained model is loaded from the root directory, the test set is evaluated, and checks for metrics are assessed. Random images are selected to visually compare prediction/ground truth/synthetic images.

### Network Implementation Stages

Initial network setup and training/testing/validation are performed for each model. The model is called with Xavier weights, and Adam optimizer applied. Cross-entropy loss is used. The training function is implemented with the values described in Tables 5 and 6. Accuracy of the model is performed throughout the training. Finally, *Epoch Verification* and *Final Checks* are performed, and results are stored for later comparison.

# **Results**

The initial main target metrics collected for this project were *loss* and *overall* *accuracy* for each network. However, it was determined that classes covering large areas of the image (such as background or a large cell) could skew the results. Intersection over Union (IoU) was implemented on a class basis as well as the IoU mean of the classes to have more detail about individual cell accuracy. IoU is used to measure how close two masks overlap (aka intersect). If the masks overlap exactly, then the IoU = 1; if they do not overlap at all, the IoU = 0. The mask used to determine the IoU is the prediction mask over the ground truth mask. This method not only detects the object and its shape but also compares the prediction of the class.

Chart, bubble chart

Description automatically generated

Figure : Simplified example of the calculation of Intersection over union

## Comparison of Metrics

Dataset 1 (base dataset of 114 annotated images) was first run through the three models with a model split of 80/10/10, followed by a model split of 70/15/15. A third model split of 60/20/20 was abandoned after seeing the individual results of the first two. Figure 18 shows the results of those splits with comparisons between SegNet and SegNet + Edge for validation and test sets. All results shown are in percentages, except for *loss*.

Classes that dominated the image (background, RBC, and RBC\_ol) had an IoU between 48.5019% and 93.4489% and contributed to overall higher accuracy. These high results were followed by the class platelets with a higher number of objects in more images, though many were very tiny. Further investigation of the remaining classes showed extremely low IoUs, thus a complete inability to identify most of the key classes to this project.

The low IoU classes (<21%) had limited examples for the networks to train on, and so the low results were not completely unexpected. Interestingly, in the 80/10/10 split where the models had more images to train with, there appears to be a more significant variation of results in the test set. The author considered that the splitting of the RBC's in manual annotations were likely contributing to errors, thus an adjusted IoU mean is added that removes those two classes. Due to the low IoUs for most classes, a synthetic set was created.

Table

Description automatically generated

Figure : FINAL RESULTS OF BASE IMAGE SET

The synthetic dataset was merged with the base dataset totaling 1616 images. The new dataset was tested with the same model splits as before, with the results of the 80/10/10 split shown in Table 8. The 80/10/10 split was chosen since the total image count was still lower than used in similar research, and this gives the models more data to learn from.

An immediate standout observation is that the cell classes that performed well before (RBC and RBC\_ol), dropped to less than 1.5%. Due to their density in the base dataset, they were not generated in the synthetic dataset. Platelets, another class that had a higher count in the base dataset, but had a tendency to be very small, also had a considerable drop. In the case of the SegNet test set, platelets dropped from 45.1855% to 0.7278%.

Conversely, the almost undetectable cells we saw previously have made incredible progress (though still low in most cases). Eosinophil stood out with a change from 0.0869% to 43.2296% in the SegNet test set and 0.0898% to 32.594% in the SegNet + Edge test set. The Neutrophil class also stood out between the SegNet Model (13.8273%) and the SegNet + Edge Model (44.5503%).

Given the IoUs of the different cells, accuracy does not appear to be a good metric. The synthetic dataset had a very high percentage of the image that was background, thus falsely elevating the accuracy results. As with the previous results, an adjusted mean was added to remove the classes RBC and RBC\_ol, though this had the opposite effect of raising the mean where previously it had lowered it.

Frequent significant differences between the SegNet results and the SegNet + Edge results are observed with a range of 6.127 and 30.723 percentage points for the test set for the eight original cell classes. Nothing remarkable for the loss presented across models.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **SegNet** | **SegNet + EDGE** | **SegNet** | **SegNet + EDGE** |
| **(Val Set)** | **(Val Set)** | **(Test Set)** | **(Test Set)** |
| **Loss** | 0.2742 | 0.3889 | 0.2607 | 0.3341 |
| **Accuracy** | 93.2704 | 91.9911 | 91.0738 | 91.8879 |
| **Background** | 97.8084 | 94.6816 | 95.7375 | 94.2119 |
| **Basophil** | 14.538 | 13.2722 | 8.9027 | 2.7757 |
| **Eosinophil** | 31.2126 | 26.8891 | 43.2296 | 32.594 |
| **Erythrophil** | 8.7142 | 16.1332 | 5.0236 | 14.6106 |
| **IG** | 9.4065 | 30.2469 | 9.3827 | 17.7288 |
| **Lymphocyte** | 14.7109 | 23.7224 | 9.3666 | 17.7288 |
| **Monocyte** | 25.7398 | 33.9071 | 23.3619 | 31.7164 |
| **Neutrophil** | 26.04875 | 71.4724 | 13.8273 | 44.5503 |
| **Platelet** | 19.5127 | 20.8977 | 0.7278 | 15.2304 |
| **RBC** | 0.3531 | 0.0955 | 0.4349 | 0.0475 |
| **RBC\_ol** | 0.5145 | 0.176 | 1.3301 | 0.0791 |
| **IoUm** | **22.5963** | **30.1358** | **18.4702** | **24.3367** |
| **Adjusted Mean** | 27.521317 | 36.802511 | 23.284411 | 30.127433 |

Table : final results of MERGED base image set and synthetic image set

## Comparison of Diagrams

Figure 19 shows the training over epochs for the model split 80/10/10 with both datasets for the SegNet and the SegNet + Edge models. In the base dataset on the left, there is much more variability as the model learns. All four diagrams show how the Background class influences the accuracy. As such, accuracy is an invalid metric to use with our current datasets (given the high number of pixels in the background class). The diagrams also show an inverse relationship for IoU percentage between the base dataset and the merged dataset for RBC and RBC\_ol. Lastly, the diagrams show a smoother line after around 100 epochs, indicating this to be the target epoch.

Graphical user interface, chart

Description automatically generated

Figure : OVERALL ACCURACY AND IOU (BACKGROUND, PLATELETS, RBC, RBC\_OL, AND MEAN) DIAGRAMS with Base dataset and merged dataset for Segnet and segnet + edge models.

Figure 20 shows the same learning rate over epochs (using the model split of 80/10/10) with both datasets for the SegNet model and the SegNet + Edge model. As before, we start to see a smoothing around the 100th epoch, though around the 120th epoch for the merged dataset, we begin to see some noise again and a downturn of several IoUs. This noise indicates that we may be beginning to overfit.

All the classes show negligible results for the base dataset, though Immature Granulocytes has some remarkable variation in the SegNet + Edge model. The merged dataset shows an increase in IoU across the board, with Neutrophil having a significant increase in the SegNet + Edge model. Eosinophil in the merged dataset has a marked improvement from the base dataset but produces similar results between SegNet and SegNet + Edge. As with Figure 19, we see a smoothing out around epoch 100, with epoch 120 beginning to show variation again.

Chart, histogram

Description automatically generated

Figure : OVERALL ACCURACY AND IOU (Basophil, Eosinophil, Erythroblast, Immature granulates, lymphocyte, Monocyte, and Neutrophil) DIAGRAMS with BASE DATASET AND MERGED DATASET FOR SEGNET AND INTEGRATED NETWORK (SEGNET + EDGE).

Figure 21 compares the loss over epoch for the four different model runs. The learning rate curve looks good in all of them, though at epoch 40 there is a spike in all but the merged dataset in the SegNet model. In the merged dataset for the SegNet + Edge model the curve looks more shallow, but inspection of the loss values shows that it is only due to the spikes around epoch 20 and 40. Except for the spikes, the merged dataset for both models shows the train and validation loss much closer and thus more reliable. As with the diagrams in Figures 19 and 20, the learning rate smooths out around epoch 100, and the slow expansion between the training loss line and the validation loss line further shows that the models may be beginning to overfit.

Graphical user interface, application

Description automatically generated

Figure : Loss by epoch diagrams with BASE DATASET AND MERGED DATASET FOR SEGNET AND INTEGRATED NETWORK (SEGNET + EDGE).

## Visual Comparisons

Even though the IoU results are lower than expected, a visual display of the predictions reveals information not easily noticed by the numbers and charts alone. The merged dataset was created twice in this project, once with the synthetic images added after the base dataset, and then a second time with the order rearranged, so the synthetic dataset images were ordered before the base dataset images (since they were less in complexity).

Figure 22 and 23 shows an example prediction of the merged dataset for both models with the original base dataset first in the learning order. In both models, borders are mostly defined, and objects appear in a transitional stage of learning. Red blood cells (peach) and red blood cells that are considered overlapped or on the edge of the image (beige) are predicted quite well – which makes sense given their higher numbers that the neural networks were able to learn from. Small platelets also did fairly well as they had a medium number of instances to learn from. The SegNet + Edge image shows a slightly better prediction at the pixel level than the SegNet alone.

Chart, bubble chart

Description automatically generated

Figure : IMAGE PREDICTION RESULTS OF SEGNET

A screenshot of a computer

Description automatically generated with medium confidence

Figure : IMAGE PREDICTION RESULTS OF Segnet + Edge

The base dataset was then put at the end of the learning order in the merged dataset and the evolution of training was examined (Table 8). After the first epoch, the SegNet + Edge image has nothing but background, but by epoch 5 we have a very clear picture of most of the objects compared by a still very noisy image for SegNet. By epoch 100, the SegNet model is doing a better job with the overlapped objects and with classifying neighboring pixels of an object together, giving it a much higher IoU mean of 62.169% versus 42.168%. Table 8 highlights that the two models learn very differently.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Epoch** | **SegNet** | **SegNet + Edge** | **Epoch** | | **SegNet** | **SegNet + Edge** |
| 1 |  |  | 25 |  | |  |
| 2 |  |  | 50 |  | |  |
| 5 |  |  | 75 |  | |  |
| 10 |  |  | 100 |  | |  |

Table : Evolution of MODEL IMAGES FOr TEST SET by Epoch

For comparison with Table 8, the ground truth image and corresponding original/concatenated images are shown in Figure 24.

Chart, bubble chart

Description automatically generated

Figure : Original Image, Ground truth, and Concatenated image used in evolution of images in

# **Conclusions**

After examining all the results, the author concludes that adding an edge as a fourth channel has high potential for increasing object classification. Given the results from various models run for splits, epochs, and HED epochs used (model results available in the GitHub link listed in the index), a pattern was observed of a distinct variation between the results from the SegNet model and the SegNet + Edge model.

However, the idea that adding edge as a fourth channel will increase accuracy cannot be fully determined without using a much larger dataset. The easiest way to accomplish this would be using the synthetic generation method with more variations of foreground images and an increased count of foreground images that can be used in an image. Furthermore, considerations for how much an object can be occluded should be added into the generation of a synthetic dataset.

With the addition of the synthetic dataset, the division of RBC and RBC\_ol becomes obsolete. A synthetic dataset should be created with mature red blood cells with consideration for balancing the dataset. Balance of objects needs to be more dispersed over the image sets – thus, balance needs to occur not only over the entire dataset but also throughout the dataset.

On the process side, stakeholders must realize the costs, time, and complications of annotating a dataset. After examining some ground truth images, the author questioned if one of the object annotations in an image was incorrect. Since this is possible with any dataset, it is crucial to realize the risk of small datasets, potentially causing results to be unreliable. When possible, synthetic datasets should be used along with expertly annotated datasets or in place of them (though the synthetic dataset still will need experts to determine classes with foregrounds).

Lastly, the author determined that using a low-cost solution is possible, but with the current outline of steps may not always be reliable. Alternative faster neural networks or methods for stopping Google Colab from timing out during long runs need to be explored; as well as alternatives to low internet connection speeds. Colab timeouts were not frequent, but stakeholders running this project to train with a dataset will need to determine if this risk is appropriate for their project.

## Lessons Learned/Problems Encountered

Computer vision projects require many images to learn—more than can be reasonably hand-annotated by 1 person in a short period. Many neural network projects will have specific annotation requirements that can vary widely and produce a very wide range of results. There are many free annotation software products, but these often are very resource-intensive for hand annotating and may not produce ground truth images or may produce the result in a format that has to be further converted. Paid products often have ways to annotate faster, but the process is still slow and often produces a result that will need further conversion.

Comparative work done in a previous course did not expose a realistic time estimation for the full cycle needed to annotate/label images. Most online examples, tutorials, and previous experience used already annotated images, and there is a lack of information available on the process. Annotating a dataset from scratch is extremely resource-intensive. Without the experience of the different products that assist with this task, additional time for selecting and researching the right product for a use case can quickly derail a timeline.

Based on these observations, synthetic datasets are a much better option. The time is cut down tremendously, and there are free resources (as well as low-cost paid lessons on how to use the code available) to quickly get an extremely large dataset generated. The larger dataset increases learning for a neural network.

That said, creating a synthetic dataset introduced additional challenges. Available code all showed methods specific to instance segmentation. Since this project was designed for semantic segmentation, code had to be reworked and tested to adjust the process to semantic segmentation. This problem provided many learning opportunities for creating synthetic datasets and how it works under the covers for the different types of segmentation. Creating and using a synthetic dataset also brought to light how object class density needs to be dispersed more evenly throughout an image set, even when it is dispersed similarly throughout train/test/validation sets.

In addition to considerations for object class density within a dataset, careful thought should be applied to how balanced class pixels are within the individual images. In a computer vision project, the standard accuracy metric can be completely skewed by a class with a high number of pixels within an image and that the network knows well (potentially from learning on an unbalanced dataset). IoU corrects the issue of too many pixels belonging to a single class, but there may still be issues from an unbalanced dataset. As such, accuracy should not be used as a primary metric for datasets that fit this criterion.

Lastly, neural networks are not plug-and-play, particularly with computer vision projects. Preliminary testing was done on the networks used with a specific dataset, but they did not work with the new dataset. Additional coding and debugging for the SegNet and SegNet + Edge models took more time than expected. Knowing the image requirements of size, shape, and ground truth image availability at the beginning of a project is extremely important to avoid problems down the road.

## Future Considerations

Objects that overlap with the edge of images can cause problems with object detection. Precautions were taken in the placement process to avoid this from occurring to overcome this in the synthetic dataset. In the SegNet layers, padding was implemented, but if any images are used that are not from the synthetic dataset, then small padding to the edges should be implemented to remove the variable that may cause problems.

While analyzing the results, it was discovered that the evaluation metric accuracy was not a good fit for the datasets used in this project. IoU was used instead, but additional metrics should be added. The Dice Coefficient (F1 Score) is used for computer vision projects, but it positively correlates with IoU and so adding it would not be of much benefit. A confusion matrix that gives a breakdown of correct predictions and incorrect predictions made on a class level would help identify any potential problems with the dataset or annotations. With the confusion matrix, the mean average precision (mAP) could be calculated. mAP is a metric that compares the ground truth to the prediction and returns a score. The higher the score, the more accurate the model is in detecting objects by class.

One of the aspirational goals for this project was to use additional architectures such as ResNet or U-Net instead of SegNet. Initial attempts were made for U-Net after the SegNet architecture was complete, but the problems with images de-prioritized this goal. Alternatively, replacing the HED network with a more modern edge detection solution may provide additional information.

Similar to changing the HED network, an idea that came during the execution of this project to use the contour binary images as a fourth channel to see if having only outer edges would improve results. By testing this method, it could be quickly observed if using outer edges only shows any benefit since edge detection networks capture a lot of unwanted artifacts and edges inside objects. The additional edges inside an object may be of benefit, so doing a run-through with the already available contour images would identify if having the outer edges only helps or destroys import information.

Multiple types of tuning to the current model should be considered. Besides parameter tuning, alternatives to IoU, and the architecture considerations mentioned above, fine-tuning the HED output networks may produce better results.

Lastly, transfer learning should be considered, particularly if IoU results improved. If IoUs do not reach an acceptable level, then changing the classes to red blood cells, white blood cells, and platelets may produce a higher benefit that can then be used for transfer learning. The main idea behind this project was the ability to apply to other datasets; thus, incorporating testing and results with transfer learning would benefit others interested in seeing if the results worked elsewhere.

## Next Steps

The easiest first next step is to increase the size of the dataset. This task has already been started by the author and could be reasonably done over a weekend, with the bulk of the time creating additional foregrounds to add to the mix. By doing this, the transformations that occur in the synthetic dataset generation process would create a multiplicative increase in variation for the newly generated images. With a new large synthetic dataset, it would be interesting to test it without the base images or shuffle the base images within the learning order instead of putting them in the front or end. Too late for this project, the author learned that this could be of potential benefit.

Along the same lines, increasing the number of foreground images that can be randomly selected for a synthetic image would give a better idea of how effective this project is for different levels of overlapping objects. The current number was set at 6 as it was noted in some images that some of the foregrounds were 100% obscured when the number was set too high, but that could be corrected by adding a minimum percent of required visibility for an object -relative to the size of the object - inside an image.

Because a synthetic dataset is being used, the RBC classes should be condensed into 1 class and included in the synthetic dataset. This merging could easily be done even if including the base dataset images, as the ground truth images could have the RBC\_ol class value in the array changed to RBC. In addition, an automated process to easily count the classes on a per-image basis would be a big benefit to quickly discover any bias in the dataset and make decisions accordingly.

Finally, immediate next steps would involve doing the steps mentioned above, a round of final testing, and posting all updates and discoveries to GitHub for easy access. One of the goals for this project was to help others, and by making the information and code available to all, as well as invitations to discuss, then that will be the most important next step of all.

# **References**

1. Acevedo, Andrea; Merino, Anna; Alferez, Santiago; Molina, Ángel; Boldú, Laura; Rodellar, José (2020), "A dataset for microscopic peripheral blood cell images for development of automatic recognition systems", Mendeley Data, V1, doi: 10.17632/snkd93bnjr.1
2. F. Scotti, "Automatic morphological analysis for acute leukemia identification in peripheral blood microscope images," CIMSA. 2005 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications, 2005., 2005, pp. 96-101, doi: 10.1109/CIMSA.2005.1522835.
3. Pattanaik, P. A., Swarnkar, T. and Sheet, D. (2017) 'Object detection technique for malaria parasite in thin blood smear images', *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. doi: [10.1109/BIBM.2017.8217986](https://doi.org/10.1109/BIBM.2017.8217986).
4. Bailo, O., Ham, D. and Shin, Y. M. (2019) 'Red blood cell image generation for data augmentation using Conditional Generative Adversarial Networks', *arXiv:1901.06219 [cs]*. Available at: <http://arxiv.org/abs/1901.06219> (Accessed: 9 February 2021).
5. Marmanis, D. *et al.* (2018) "Classification with an edge: Improving semantic image segmentation with boundary detection", *ISPRS Journal of Photogrammetry and Remote Sensing*, 135, pp. 158–172. doi: [10.1016/j.isprsjprs.2017.11.009](https://doi.org/10.1016/j.isprsjprs.2017.11.009).
6. Bhavnani, L. A., Jaliya, U. K. and Joshi, M. (2016) "Segmentation and Counting of WBCs and RBCs from Microscopic Blood Sample Images". doi: [10.5815/IJIGSP.2015.11.05](https://doi.org/10.5815/IJIGSP.2015.11.05).
7. Xie, S. and Tu, Z. (2015) "Holistically-Nested Edge Detection", *arXiv:1504.06375 [cs]*. Available at: <http://arxiv.org/abs/1504.06375> (Accessed: 4 October 2020).
8. Badrinarayanan, V., Kendall, A. and Cipolla, R. (2017) "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12), pp. 2481–2495. doi: [10.1109/TPAMI.2016.2644615](https://doi.org/10.1109/TPAMI.2016.2644615).
9. Kelly, Adam (2019) *Create COCO Annotations From Scratch*, *Immersive Limit*. Available at: <https://www.immersivelimit.com/tutorials/create-coco-annotations-from-scratch> (Accessed: 3 April 2021).
10. Kelly, A. and Kelly, K. (2020) Complete Guide to Creating COCO Datasets Course, Immersive Limit. Available at: https://www.immersivelimit.com/course/creating-coco-datasets (Accessed: 3 April 2021).
11. Poostchi, M. et al. (2018) 'Malaria parasite detection and cell counting for human and mouse using thin blood smear microscopy', Journal of Medical Imaging, 5(4), p. 044506. doi: 10.1117/1.JMI.5.4.044506.
12. Mayer, N. et al. (2018) 'What Makes Good Synthetic Training Data for Learning Disparity and Optical Flow Estimation?', International Journal of Computer Vision, 126(9), pp. 942–960. doi: 10.1007/s11263-018-1082-6.
13. Raza, S. E. A. *et al.* (2019) 'Micro-Net: A unified model for segmentation of various objects in microscopy images', *Medical Image Analysis*, 52, pp. 160–173. doi: [10.1016/j.media.2018.12.003](https://doi.org/10.1016/j.media.2018.12.003)
14. Dutta, A. and Zisserman, A. (2019) 'The VIA Annotation Software for Images, Audio and Video', in *Proceedings of the 27th ACM International Conference on Multimedia*. *MM '19: The 27th ACM International Conference on Multimedia*, Nice France: ACM, pp. 2276–2279. doi: [10.1145/3343031.3350535](https://doi.org/10.1145/3343031.3350535).
15. Acevedo, A. et al. (2019) 'Recognition of peripheral blood cell images using convolutional neural networks', Computer Methods and Programs in Biomedicine, 180, p. 105020. doi: 10.1016/j.cmpb.2019.105020.
16. Marin, D. et al. (2019) 'Efficient Segmentation: Learning Downsampling Near Semantic Boundaries', arXiv:1907.07156 [cs]. Available at: http://arxiv.org/abs/1907.07156 (Accessed: 30 April 2021).
17. *SuperAnnotate | The fastest annotation platform for training AI* (2021) *SuperAnnotate*. Available at: <https://superannotate.com/> (Accessed: 1 May 2021).
18. *GIMP* (2021) *GIMP*. Available at: <https://www.gimp.org/> (Accessed: 14 April 2021).
19. Bernal, J. et al. (2019) 'Quantitative Analysis of Patch-Based Fully Convolutional Neural Networks for Tissue Segmentation on Brain Magnetic Resonance Imaging', IEEE Access, 7, pp. 89986–90002. doi: 10.1109/ACCESS.2019.2926697.

# **Index**

Code and additional instructions to recreate this project: <https://github.com/kdmishraISU/Classwork/>CapstoneFinal

While mentioned in the references, an additional note is warranted for the course to create a synthetic dataset as listed in reference Kelly (2020). The Udemy course that corresponds with the tutorial for creating synthetic datasets is particularly exceptional and adaptable for a range of annotation types.

## Key Terms and Definitions

Given the wide audience that this project is intended for, some key terms and definitions are listed here that are used throughout this document. Some terms used have a variety of definitions; thus, the definitions below provide context to how the terms are used here.

**Annotation** – a system or process that labels an object in an image to ensure an object belongs to a particular class. The system can be software, code, by manually labeling images in photo editing software, or by developing code that recognizes object edges and saves the shapes to file.

**Accuracy** – an evaluation metrics that returns percentage of pixels in the image that are classified correctly.

**Computer vision** – a field that focuses on how computers can gain high-level understanding from digital images. Computer Vision is frequently used to understand and automate tasks that normally require the human visual system.

**Contour** – the outside edge of a given object that produces a closed shape.

**Downsampling** -reduces an image using max pooling.

**Edge detection** – the process of detecting edges in an object. In the context of a neural network, it is a neural network that is given an image and produces a corresponding image that displays only edges.

**Epoch** – a pass of all the training set images through the neural network that is then used to tune the weights. Each number of an epoch represents the number of times the set has been passed through the network.

**Ground truth** (also referred to as a "mask") - a corresponding labeled image that is the basis from which a system will learn information specific to an object class.

**Intersection over Union (IoU)** (also known as the Jaccard index) - an evaluation metric used to measure the accuracy of an object detector on a particular dataset.

**Loss** - a summation of the errors made for each example in training or validation sets.

**Neural network** – an adaptive system that uses algorithms to look for patterns in data and to give a prediction of the likelihood that an object belongs to a particular class.

**Object detection** – the process of detecting objects in an image.

**SegNet** – a convolutional neural network (CNN) architecture used for semantic pixel-wise segmentation.

**Semantic segmentation** – the process where objects in an image are labeled by class.

**Upsampling** - enlarges an image based on original pixel values.

## Methodology Flowchart

Diagram

Description automatically generated

Figure : Flowchart for end-to-end project steps