LeukosCognosis: A Machine Learning Algorithm to Diagnose Acute Lymphoblastic Leukemia

By: Sirihaasa Nallamothu
University High School
601 Gregory St, Normal, IL 61790
sirihaasanallamothu28@gmail.com

LeukosCognosis: In-depth Video
LeukosCognosis: Brief Overview Video
Github Repository

Introduction

Every 9 minutes, a person in the United States dies from Leukemia, the cancer of blood-forming tissues in the bone marrow and blood cells [1]. Leukemia is often the result of an unusual amount of white blood cells in the body. The rapid production of abnormal white blood cells impairs the bone marrow's ability to produce red blood cells and platelets, a necessary response to fight infections. The inability to fight infection along with a weakened immune system makes the body susceptible to other diseases. The general symptoms include the development of lymph nodes in the groin and neck area, petechiae, [red spots on the skin], extreme fatigue, and bone tenderness [2]. The symptoms increasingly worsen as the Leukemia progresses through the stages. There are many classifications of Leukemia including Acute Lymphoblastic Leukemia, Acute Myeloid Leukemia, and Chronic Myeloid Leukemia. Acute Lymphoblastic Leukemia[ALL] is the most common type of childhood cancer and is typically found in the 2-7 year age group, however, it can occur in adults aged 55 and up. This specific subcategory of Leukemia is caused by DNA errors in bone marrow cells [3]. The word "acute"denotes the fact that the cancer spreads extremely quickly throughout the body, and creates immature and irregular white blood cells. Treatment options are available, including medications, chemotherapy, supportive care, and allotransplantation[transferring/replacing live organs and tissues], still, these methods do not guarantee the termination of cancer. The survival rates after 5 years of treatment are around 50% to 64% [4]. The diagnosis of Acute Lymphoblastic Leukemia can be completed through a lab test, which entails a blood smear test confirming where abnormal lymphocytes are located, along with a white blood cell count. However these tests, performed by the human eye and a microscope, are only 30-40% Accurate. Once the presence of Acute Lymphoblastic Leukemia is noted in the white blood cell count, and smear test, a bone marrow test is conducted to authenticate and verify the diagnosis. Treatment must begin immediately. However, the test results for the white blood cell smear can take a minimum of 3-4 weeks to complete [4]. Acute Lymphoblastic Leukemia [ALL] can rapidly spread throughout the body and unbeknownst to the patient, can progress and worsen in a matter of 2-3 weeks. Oftentimes, the lab results take a little over a month to fully analyze and return to the patient [5]. This drastically decreases a patient's ability to survive and reduces the overall positive impact of treatment methods. The need for an instantaneous and efficient way to accurately diagnose Acute Lymphoblastic Leukemia in the medical field is apparent. An error-free and accessible method to determine the existence of Acute Lymphoblastic Leukemia can replace the need for a month-long lab diagnosis, quickly determine the necessity for treatment, and save a patient's life. In order to effectively solve this problem, the author, Sirihaasa Nallamothu, created LeukosCognosis, a machine learning algorithm that analyzes suspected microscopic white blood cell smears, and diagnoses Acute Lymphoblastic Leukemia on the spot. LeukosCognosis is essentially an image classifier created through a Convolutional Neural Network. In short, a Convolutional Neural Network is an algorithm, with many connected layers, that can take any input image, and utilize image recognition and processing to identify and classify said image [6]. Convolutional Neural Networks, also referred to as CNNs, utilize complex deep learning algorithms, and neurons to process pixel and image data. A CNN has the ability to "learn", by applying information accessed through training images and comparing it with the input image. A machine learning algorithm's ability to learn is similar to a human brain's cerebrum, which controls memory, speech, and cognitive thinking [7]. The model was coded using the Tensorflow environment, the Python programming language, and Jupyter Notebook. The LeukosCognosis algorithm is around 97% accurate and returns diagnosis results almost immediately. LeukosCognosis eliminates the need for a lab analysis of a blood smear; all the Oncologist[doctor who treats cancer] would have to do is take a microscopic image of the white blood cell smear, and upload it to the model. Within seconds, an accurate diagnosis will appear, along with the percentage pertaining to the probability of Leukemia. With an accurate confirmation of Acute Lymphoblastic Leukemia, bone marrow testing can proceed, and life-saving treatment can begin immediately. However, it is important to note that the LuekosCognosis algorithm had to undergo a great deal of experimentation, testing, prototyping, model revisions, and reconstruction of the Convolutional Neural Network. The evaluations and experiments were conducted in order to perfect the overall medical precision of the model. The training data set for the model was also regulated throughout the coding process. Continuous research on Acute Lymphoblastic Leukemia was also necessary, to ensure the utmost medical validity. The methods and materials that were necessary to create and evaluate the LeukeosCognosis Machine learning algorithm, along with the results of the conducted experiments will be heavily examined and analyzed in both a medical and programming perspective throughout this paper.

Note: All figures and images that are referenced throughout this paper are located in the 'Figures Section'

Methods and Materials

The LeukosCognosis algorithm and Convolutional Neural Network code underwent a variety of alterations and revisions in order to increase efficiency, along with multiple evaluations to determine the exact medical capabilities of the model. The model's validation accuracy/loss and learning rate were tested through a surplus of drafts and "reconstruction" of the code itself. The medical accuracy, along with the diagnosis of test images was also experimented on. In addition, the author [Sirihaasa Nallamothu] conducted experimental trials to determine the accuracy of the model's predictions, the benefits of utilizing this model over a lab blood smear test, and the precise amount of time the model takes to provide a prediction. The methods and materials, along with the procedure applied to perform these tests will be heavily highlighted and explored throughout the entirety of this section. Along with this, the code for LeukosCognosis will be thoroughly explained.

Constructing and Perfecting the Model: Procedure and Explanation

Constructing the machine learning model, along with the modifications and edits made to the code through sufficient testing, was a vital part of creating the perfect LeukosCognosis Algorithm. When programming a solution of importance, coders and developers follow the "Software Development Design Process," in order to ensure the final product is of optimal performance (Fig. 1). The "Software Development Design Process," involves a repeating cycle of "Designing, Prototyping, and Reviewing/Refining" the solution. The first overarching step in the "Software Design Process" is to complete a "Requirements Analysis". This means developers must assess the overall problem, gather information relevant to the problem, and determine any prevalent needs. Developers often revisit the "Requirements Analysis" stage in order to seek out new information or determine accurate solutions. In the "Requirements Analysis" process, the author gathered sufficient information on Acute Lymphoblastic Leukemia, including the symptoms/overall effects on the body, treatment methods,and the survival rate of those affected. During this process, the author also researched and identified inconsistencies in the diagnosis procedure. After gathering information in the "Requirements Analysis" stage, developers often segway to the "Design and Prototyping phase". These phases closely relate to each other and intertwine in their processes. Typically, in the Design phase, developers use in-depth information gathered on the problem to come up with a viable solution. In the design process, developers create a general solution to the problem and plan out the next steps. Prototyping is also included in the design process, as the problems identified in prototype solutions can be improved upon and utilized in the official solution.

The Author found many intriguing statistics on the imprecisions of the Leukemia testing procedures, specifically pertaining to the lab tests performed on white blood cell smears. As mentioned in the introduction, White Blood Cell Lab Smear results often take 3-4 weeks to complete and return to the patient. To add on to the horribly delayed time period of lab tests, human diagnosis of white blood cell smears are often only 30%-40% accurate[8]. After conducting in-depth research on these inaccuracies of the diagnosis process, the author came up with a simple solution to solve this problem, in the design process. When diagnosing Leukemia, doctors and oncologists look for Complete Blood Count or CBC, to determine the amount of abnormal white blood cells and red blood cells that are present. An arbitrarily high or low amount of white blood cells can indicate Leukemia. Additionally, oncologists look for abnormally shaped lymphocytes to determine Leukemia. Oftentimes, the human eye is unable to detect and pre-diagnose Acute Lymphoblastic Leukemia based on miniscule signs and traces found in the blood smear test. The Author considered replacing the inaccurate human eye with a machine. If a computer diagnosed Acute Lymphoblastic Leukemia, the results would be much more reliable and trustworthy. A computer could cut out the 'middle-man' of sending white blood cell smears to the lab: effectively saving time and money. The time saved by the accurate diagnosis could be used by the oncologist to proceed in the next steps of a patient's treatment. The computer's way of diagnosing Acute Lymphoblastic Leukemia would have to analyze the white blood cell smear and classify or diagnose the blood smear. Additionally, the computer would also have to give the oncologist the percentage of the probability of Leukemia. Put simply, the solution to diagnose Acute Lymphoblastic Leukemia would have to be similar to an image classifier that performs diagnosis via microscopic white blood cell smear images.

The Author decided to prototype the image classifier model, one of the key aspects of the "Designing and Prototyping" stage, by utilizing Wolfram Alpha. Wolfram Alpha is a computational intelligence platform that allows users to complete a variety of versatile and unique projects. Essentially, Wolfram Alpha is a computer science platform for developers to prototype their technological ideas in an easy and simple way. Wolfram Alpha is also compatible with a variety of coding languages, which makes it easier to integrate and relate back to the overall solution.Wolfram Alpha combines aspects of the "Wolfram Language", a programming language specific for Wolfram Alpha, and "Pseudocode". Pseudocode, also known as "Natural Language" is used in Computer Science as a way for humans to understand operational principles and logic behind a certain piece of code [9]. Pseudocode can also be used to plan out the logic behind certain conditionals.

Using Wolfram Alpha, the author was able to build a simple image classifier prototype that diagnosed Acute Lymphoblastic Leukemia, in a matter of days. The Image Classifier included two categories, "LeukemiaWhiteBloodCells" and "HealthyBloodCells". Because the Image classifier was a prototype, overarching labels pertaining to the two distinct categories was sufficient. To make a successful image classifier, a versatile image dataset is necessary. Image Datasets, allow the classifier to individually train off of each category, and compare certain similarities in the image to the test image. Staying true to the simplicity of the prototype, the author decided to utilize Wolfram Alpha's "Import Function" for an image dataset. The "Import Function" allows for large amounts of images to be pulled from certain websites, by specifying the url. The URL used for images was found by entering keywords such as "Leukemia White Blood Cells", "Leukemia White Blood Cell Smears", and "White Blood Cell Microscopic Image Smears'(Fig.2). Once the desired URL Image Dataset was determined for both the

"LeukemiaWhiteBloodCell" and "HealthyBloodCell" categories, the author proceeded with constructing the prototype. In order to ensure maximum accuracy, in terms of the correct classification and optimal probability of Leukemia, the author adhered to the following steps:

1. Allocate Testing Images [5-9 images of each category]. Ensure that these Images are not present in Image URL Datasets
2. Configure Keywords to type into Google Images to find new Image URL databases
3. Import Image URL Dataset into Wolfram Alpha Prototype code [Ensure that the images are microscopic white blood cell smears]
4. Make any necessary changes to the model in terms of pseudocode
5. Test model with allocated test images, Healthy White Blood Cells and abnormal Lymphocytes
6. Record resulting Accuracy of each image in a table. Record the classification of each image as well, and whether it was correct or not.
7. Complete steps 2-6 until the model is at least 80% accurate on both the probabilities of the Leukemia and Healthy White blood cell classifications.

The Steps were repeated around 6 times, until an optimal accuracy was reached. Following these steps allowed for the creation of an extremely accurate prototype model, which, in turn, helped in the process of building LeukosCognosis. The Author was able to utilize the knowledge she gained from building the prototype model and apply it to the final LeukosCognosis Model. The steps resulted in conscious pseudocode (Fig.3). The prototype code starts out with defining the Function "LeukemiaDetect" as a Wolfram Alpha command called "Classify".Then the categories are specified and an Image URL is allocated to each category. Finally, the URL is specified as an "Image" datatype. A Wolfram Alpha "Cloud Deploy Command" is also integrated into the code. This command allows the coder to deploy their image classifier to the web in a seamless fashion. It also allows for easy deployment and UI[user interface] interaction. [Note:The results of the Wolfram Alpha Image Classifier, including the "probability statistics", the accuracy, and more, will be discussed in the "Results" Section.]

After successfully creating an Image classifier prototype, the author decided to proceed in the "Software Development Design Process"(Recall Fig.1). Typically, after prototyping a plausible solution, and determining that the essence of the solution solves the problem,the development of a plausible solution begins. The steps of "Design" and "Review/Refine", closely relate to each other. When taking steps toward designing the final solution, coders and developers explore all viable platforms and methods in which they can code their classifier. This allows the solution to be built on a software that can provide efficient and accurate results. The author completed the following "Platform Thought Process" when determining a platform:

1. Research, in depth, different Image Classifier platforms
2. Does the platform allow for Outside of the box thinking and programming?
3. Is the platform compatible with other devices and services?
4. If utilized, will the platform be capable of providing productive results?

After thoroughly completing this through process, the author decided on what notion to utilize when developing LeukosCognosis. In order to expand on the Acute Lymphoblastic Leukemia Image classifier, and increase the overall accuracy, the author decided to take advantage of Machine Learning along with Data Science and integrate it into the medical tech solution. **Machine Learning Is** the application of Artificial Intelligence through Neural Networks, Image Classifiers, and more for the greater development of technology for the human kind [10]. Python is the typical programming language used when coding machine learning applications. Machine Learning has a multitude of capabilities and can be utilized in the intersection between the medical and technological field. The author decided to build the final LeukosCognosis Solution in the form of a **Convolutional Neural Network**. A Convolutional Neural Network is one of the many subcategories of Neural Networks in Machine Learning. Neural Networks, in general, are composed of several layers and 'nodes'. These layers and nodes, each with varying purposes, come together to complete the function of classification, and contribute to the overall algorithm as well. A Convolutional Neural Network is essentially a neural network that has extremely accurate image classification and recognition capabilities. These neural networks are utilized in a variety of everyday applications, from facial recognition to powering autonomous vehicles. These neural networks have the capability of taking in an 'input image' and providing a classification, along with a percentage of probability of the suspected classification(Fig.4). The Image[Located in the figures section]displays the means of a Convolutional Neural Network in simple terms. It is important to note that this is a simplified version of this neural network, and not all aspects are identified. A Convolutional Network takes in an input image, which is then pixelated and analyzed through the Convolutional Layer. The Convolutional Layer essentially finds notable features in the image to help with classification, including edges, curves and shapes. This layer works closely with matrices and arrays in order to pixelate and create a feature map of the input image. The pixelated image then goes through the pooling layer, which involves max pooling. This means that the pixelated image is broken down into groups and thoroughly analyzed for key features. Oftentimes, the convolutional layer may Sharpen or use edge detection, and blur on the image. The goal of this layer is to compile the key features and keep them on hand for classification. The Neurons in the Dense layer, essentially filters the key features, and gets it 'ready' for the final categorization of the image. The output layer provides a
"Yes" or "No" answer to the classification, along with the percentage probability. Throughout the process different optimizers, or algorithms that help with learning rates, are utilized to increase the accuracy and efficiency of the model itself. Neurons in a convolutional neural network are especially important as they provide input and output information capabilities and have the ability to transfer matrix values,which is useful in mapping the key features of an image. Neurons are also useful in constructing distinct convolution layers in the algorithm. The image below depicts the role of neurons in a convolutional neural network (Fig.5). When an

image is processed through a neural network, its features are broken down through the ‘filters’, or the use of matrices. This pixelated image is then stored into neurons when it goes to the pooling layer. After this layer, the notable features are found, which are then flattened and stored into 1D matrices, for easier classification and use. Neurons truly play a vital role in convolutional neural networks. There are many different methods and ways to build a convolutional neural network. Using the “Platform Thought Process”, the author determined that LeukosCognosis would be coded using Python and the Tensorflow/Keras Libraries. Tensorflow and Keras and high level machine learning libraries that provide coders an easy way to build neural networks without focusing on the math behind it. Tensorflow and Keras are used by companies and data scientists to develop deep learning applications. Utilizing these libraries in the LeukosCognosis Solution increased the capabilities of the neural network and the matrices and helped improve the accuracy of classification.

In order for a Convolutional Neural Network to function, a training and testing dataset is necessary. These training and testing datasets must contain high resolution images and must be compiled and stored. A separate comprehensive training dataset is necessary for each category. It is important to note that training and testing datasets must be kept separately to avoid bias in the model, and to prevent accuracy errors. For the LeukosCognosis Convolutional Neural Network, there are two categories of possible classification, “Acute Lymphoblastic Leukemia” and “Healthy White Blood Cells' '. This means that the author must find at least 200 images of abnormal Acute Lymphoblastic lymphocytes in the form of a white blood cell smear and at least 250 healthy white blood cell images. Each image must be a high resolution microscopic image. In order to acquire high quality images, the author applied for a dataset from esteemed Leukemia researcher Fabio Scotti. Combined, the datasets contained over 500 images of Acute Lymphoblastic Leukemia and Healthy white blood cell sears taken with a Canon PowerShot G5 camera with an optical laboratory microscope. These images not only focused on abnormal lymphocytes, but also the white blood cell count: combining these images would result in the perfect testing and training dataset for LeukosCognosis (Fig.6). [Note: In addition to the dataset by Fabio Scotti, the author added in other images of Acute Lymphoblastic Leukemia and Healthy white blood cells]. When coding neural networks, it is important that each image is of the same size, so notable features, such as edges and curves, can be recorded at the same proportion. The author utilized a software called IrfanView, to condense and compress these images to the same size, in terms of pixels. Once the author acquired and combined the comprehensive datasets, creating a training and testing dataset, the programming process began.

The author utilized Jupyter Notebooks to code the Convolutional Neural Network. Jupyter Notebook is an application that hosts interactive ‘ipython notebooks’, which allow coders to break down their code step-by-step. As the first step, the comprehensive datasets were imported into the code, and a path for each category was defined. The author decided to utilize color in the images, instead of converting it to Grayscale, which makes the images gray. It is necessary to leave color in the images, as cancer can be identified by the overgrowth of the brightly colored purple nuclei (Fig. 7). The code in the image[located in the ‘Figures section’], essentially defines the categories of images and displays in image with the primary “color map” of red, green, yellow and blue. The image is then displayed through the matplotlib python module, which allows for graphs and images with alterations to be displayed. The categories of “Acute Lymphoblastic Leukemia” and “Healthy White bLood Cells' ' were then put into a “Python List”. The category“Acute Lymphoblastic Leukemia” denoted as Leukemia in the LuekosCognosis code is ‘1’ and “Healthy White Blood Cells” is ‘0’. For the sake of simplicity, the ‘0’ and ‘1’ will be referred to as “Healthy” and “Leukemia” throughout the paper. After importing the images and defining the categories, the training data was created. The author manually created the training data in the Jupyter notebook, as in order for the image dataset to be manipulated in terms of size, images, and the images can be shuffled in any random order. Many parameters can be specified when creating the training data, including color, size, and category labels(Fig. 8). The training data code, featured in the ‘Figures’ Section, used python “For Loops” and utilized Numpy, or Numerical Python. The Numpy module helps the matrices and arrays in the convolutional neural network. When creating the training data, the images, which were assigned a category of ‘0’[Healthy] and ‘1’ [Leukemia], were put into a Numpy array, or matrix. This allows for ease when creating the training data; putting the image dataset into an array/matrix can help programmers when debugging errors. After creating the training data, the image dataset was then randomly shuffled, to prevent model bias (Fig.9). The code also illustrates that the length of the training data was 563 images. When the training data was shuffled a sample of the categorization labels of 10 images were printed out. In the sample, an equal amount of “Healthy White Blood cell’ and “Leukemia” image labels were printed out. [Denoted ast ‘1’s and ‘0’s]. Once the training data was created, the author created empty numpy matrices, also referred to as arrays, that denoted the features of the images and the labels, or the categories (Fig. 10). By creating these numpy arrays, the author ensured that the classification process in the convolutional neural network, ensues with ease, specifically the storage of pixelated and flattened images in the neurons of the network. After ensuring that the shape, summary and dimensions of the “Feature” and “Label” Numpy arrays were completed, drafting and constructing of the first layers of the neural network began.It is important to note that the author started by creating a rough draft of the model, specifically the layers of the neural network. Then, testing and experimental procedures ensued to perfect the overall model. The author started by importing the necessary Tensorflow modules it takes to create the layers, neurons and density of a convolutional neural network (Fig. 11). The Tensorflow and Numpy models were imported along with the module ‘Cifar10’. Cifar10 is a popular module used in machine learning when training neural networks based on image classification features. The author imported this module just in case it needed to be utilized in the code. The sequential and model was then imported, in order to better handle the matrices and arrays being created through numpy. Lastly, features such as Batch Normalization, Flatten, Max Pooling and Conv2d were imported in order to better assist the capabilities of each layer and each neuron. These modules especially helped the author better improve the convolution abilities. Once the correct modules were imported, the author embarked on the construction of each of the individual neural network layers, including the Convolutional Layers, the pooling layers, and the dense layers. These layers are vital for the function of an image classifier, and are necessary when building a neural network. Each layer, with its own purpose, helps to store data and notable features about the pixelated and convoluted image in arrays and matrices, and compare them against the test image. These layers also helped to train the neural network off of the aforementioned training data (Fig. 12). The code depicts a very basic convolutional neural network. It is important to note, that before the basic layers of the convolutional neural network were implemented, the author 'pickled in' the empty X and Y NumPy arrays that were to contain the notable features and labels of the training data images. In python there is a module called 'pickle' which allows for certain arrays to be saved across multiple files. The author started out by creating the variable model and giving it a Sequential Python function.The Sequential function allows the convolutional neural network to have multiple functions layers with matrices[1D and 3D] as image filters. Next, the author created a basic convolution layer by utilizing Conv2D. Using the code, "model.add(Conv2D(256, (3, 3), input_shape=X.shape[1:]))", the author essentially called on the shape of theNumPy training data feature array and assigned it to be the first convolutional layer's matrix shape. This specific overall layer has a 3 by 3 matrix input and has 256 smaller layers, each with their own image filter. Recall that the convolutional layer breaks down the image by enhancing and sharpening it. The next line of code contains the model activation, or 'Relu' also known as Rectified linear unit, which essentially synthesizes the math behind machine learning, and implements logistic regression in the code. A Max Pooling layer is added after the activation. Recall that Max Pooling entails grouping up the pixels in 'pools' and further breaking them down and storing notable features. This specific max pooling layer breaks down the image in a 2 by 2 dimension matrix. These steps are then repeated in the code. A dense layer, also called a fully connected layer and the final layer that the image filters through, is added containing 64 neurons. The dense layer is often considered the most raw layer in the neural network. A sigmoid activation, similar to 'Relu' is also included to help with the structure of the neurons in the dense layer. Lastly, in the 'model.compile', a metrics line of code is added which ensures that the accuracy of the network is relayed to the programmer. The above was an explanation of the basic and necessary layers of a convolutional neural network. In order to actually run the neural network model, and in order to train the machine learning algorithm, one last line of code is needed (Fig. 13). The code specifies the parameters and details of how the algorithm will run and train off of the training data set. This line of code specifies the usage of the "Features and Labels" NumPy arrays through 'X' and 'y'. The code 'batch_size' entails the amount of images the model will train off of at a time. Batch Sizes must always be even numbers, and must be low amount of images. The author specified that the model will train off of 32 images at a time. The amount of epochs was also specified in this line of code. In machine learning, an epoch is one complete iteration of the model training off of the dataset and the specified layers with neurons, by running the code. In short, the author specified 10 epochs, meaning the code would run 10 times. Typically, there is a correlation between the amount of epochs and the accuracy of the model itself. Lastly the author specified the 'validation split' in the code as 30%. This means that 30% of the image dataset would be put aside when running and training the model to test. Adding a validation split is beneficial as the model has its own testing images in the training dataset to compare against. When running this model, the author will receive output metrics of 4 percentages in correlation to the amount of epochs including: validation accuracy and loss[a comparison of the model against the validation data] and model accuracy and loss(Fig.14). In machine learning, loss can be considered the amount of time the model gets a classification or categorization incorrect.The amount of time, in seconds, it took to train the model is also included in the epoch metric reading.

After building the basic code for the convolutional neural network, the author decided to perfect and improve the accuracy and efficiency of the model. The author started out with the basic code for a convolutional neural network, and performed her own experimental trials to determine what optimizers, algorithms, neurons, layers and 'weights' to add to the model. These experimental trials were performed based on the validation loss and accuracy along with the model loss and accuracy. The author adhered to the "LeukosCognosis: Final drafting" procedure when perfecting the model:

1. Create a table or spreadsheet to record the results and progress of perfection the model
2. Run the LeukosCognosis convolutional neural network code in the Jupyter Notebook
3. Record the amount of epochs the model iterated and completed; Include the amount of individual times the code ran.
4. Record the Validation Loss and Accuracy percentage for the first and last epoch
5. Record the model Accuracy and model loss percentage for the first and last epoch
6. Based on the performance of LeukosCognosis, add layers,optimizers, 'class weights' and other supplemental lines of code to improve the accuracy of LeukosCognosis.
7. Record the supplemental line of code in the table, and explain the impact of the additional lines of code added to the model, or what changes were implemented and integrated into the code
8. Repeat steps 2-7 with the new and updated code until the validation accuracy is at least 90%
9. Create Matplotlib graphical representations of the epochs in correlation to the validation loss and accuracy.

After performing this procedure for a substantial amount of time, the author was able to perfect the model, and improve the overall validation accuracy of the model itself. Many tweaks and adjustments were made to the code based on substantial observations made by the author. Please note, that the data collected from perfecting the overall model will be featured in the results section. All steps, additions and addons to the model, including explanations behind the accuracy and epoch, will be thoroughly explained in the results section. However, the final draft of the model code will be thoroughly explained and dissected in the next paragraph.

The "LeukosCognosis: Final Drafting" procedure was highly effective when it came to perfecting the overall model. As an end result, the author was able to make LeukosCognosis around 98% accurate in both the validation accuracy and model accuracy. The Validation loss and model loss was under 1%, making the model's 'turn-over' and 'inaccuracy' rates extremely miniscule. In short, the author was able to perfect the classification abilities of LeukosCognosis, and make the medical-tech device extremely precise. The code for the final draft of the LeukosCognosis convolutional network included additions such as the algorithm optimizer 'Adam', the Tensorflow function 'Flatten', Batch Normalization, padding, class weights, binary crossentrophy and the implementation of advanced convolutional layers(Fig.15). In the code, located in the 'Figures Section', the author imported the generic modules necessary for a convolutional neural network. Modules such as tensorflow, Sequential, Conv2D, Max Pooling and Dense layers were imported. Next, the NumPy arrays of the training data features and labels were 'pickled in'. This allowed for these arrays to be referenced later in the code. The author added in the following line of code "tf.compat.v1.random.set_random_seed(1234)", in order to initialize the convolutional neural network, and ensure that results are reproducible throughout significant model 'reruns'. The number '1234' is simply a random number that the author selected in order to initialize the model. After adding the sequential function in the code, to ensure that multiple layers are viable, the author created an overarching convolutional layer that had 32 overall layers. The code was similar to the basic convolutional network code, however it implanted padding. The code 'Padding= same' ensures that the size output of the feature extraction maps and the notable features are around the same size. This helps with the processing of a convolutional neural network. The code 'Batch Normalization' line of code is a technique used by the author in order to improve the speed and efficiency of the convolutional neural network itself. By implementing this line of code, the author ensured that the convolutional neural network was able to learn faster through a small amount of training data. Another Convoluional layer, in which the image is sharpened, and notable features are extracted and stored into neurons, along with another batch normalization layer is utilized. Max Pooling, also found in the basic convolutional neural network code, is implemented in order to break down the feature map and extract features. The overarching Convolutional Layer is then implemented, twice, each time with an increasing amount of layers, along with this the max pooling layer is also implemented. The author then used the code 'model.add(Flatten())'. This feature allows the multidimensional array in which the extracted and notable features are stored are turned and transformed into a 1D or one dimensional array (Fig.16). This feature allows the data and neurons to be read and accessed more easily. Lastly, a handful of dense layers are added in, along with the sigmoid activation. In the 'model.compile' the author included a loss function called 'Binary Crossentrophy'. Binary Crossentrophy, essentially tells the coder how good or bad the model's predictions are. This loss function also aids in the final categorization of the image. The optimizer ‘Adam’ is also utilized in the code. ‘Adam’ is an algorithm that allows for maximum accuracy in training, and helps to increase the learning rate. When implementing this algorithm developers must specify certain parameters, including the ‘amsgrad’, which helps with the math behind the deep learning, and much more. In the code that specifies how the model ‘learns’ [The history=model.fit code], the author included ‘Class_weights’. Class Weights is a way of balancing out the bias and influence of unbalanced datasets. When creating the image dataset, the author had less Acute Lymphoblastic Leukemia images than Healthy blast cell images. The author fixed the unbalanced dataset by adding the slightest bit more ‘preference’ through class weights. The author also reduced the batch size to 16, and the validation split to 15% of the data. The amount of epochs were increased to 15, in order to increase the model’s accuracy (Fig. 17). The side by side comparison shows that the low model accuracy at 72% in the 1st epoch and at around 100% in the 15th epoch. In order to provide a visual representation of the layers in the model, the author utilized the code ‘model.summary’(Fig. 18). The model summary depicts the amount of layers that were implemented in the convolutional neural network. This shows what the image will filter through when actually testing the model.

The author perfected the model by adding more layers to the model inorder to increase the accuracy and the neural network’s ability to extract notable features. These additions made the Convolutional Neural network 98% accurate. In order to view the step-by-step results of the procedure that was undertaken when perfecting this model, please refer to the “Results” section.

LeukosCognosis Accuracy Testing: Procedure and Explanation

Once the LeukosCognosis Convolutional neural network accuracy was around 98%, the author decided to proceed in the “Software Development Design Process,” and into the “Testing” stage. The author performed several experimental trials in order to improve the accuracy of LeukosCognosis, however, it was necessary to test the model utilizing a plethora of ulterior testing images. The author determined that experimental trials will be held to determine the viability of the following: Amount of time it takes to test an image and Accuracy/Correctness of the classification. A Convolutional Neural Network must be tested with images outside of the training dataset, in order to determine the true accuracy of the model. The author compiled 19 images in order to test the model. These images consisted of 7 Healthy blast cells in the form of microscopic blood smear images and 7 Acute Lymphoblastic Leukemia blast cells in the form of a microscopic blood smear: these images were separated from Fabio Scotti’s image dataset. An additional 5 images were taken off of multiple Acute Lymphoblastic Leukemia and health blood cell websites. The author completed the following procedure in order to efficiently test the classification abilities of the model:

1. Create a table or spreadsheet to record the results of Testing the overall Classification abilities of the model.
2. Start by testing the Acute Lymphoblastic Leukemia blast cells first, and proceed with testing the Healthy White Blood cell smears.

3. Record the classification of each image, and the probable percentage of detected Leukemia in the testing image
4. When testing the image is running, record the amount of time it takes to receive the classification.
5. Repeat steps 3-4 until all images are tested.

The author followed this procedure in order to accurately test the model. In these specific experimental trials, the independent variable, or what the author controls, is the testing image that is provided, the dependent variable is the diagnosis and the amount of seconds it takes to receive a diagnosis. The control, or baseline, would be the actual diagnosis of the test image [Determined through Fabio Scotti's classification]. Please note that the results of each individual image will be adequately analyzed and explained in the "Results" section. The author utilized detailed code in order to test the LeukosCognosis Algorithm(Fig. 20).This code was essentially to adequately test the model through a testing image input. The author started off by importing the 'CV2' module, which allows for advanced deep learning and computer vision. The file path, as to where to find these testing images is also specified, aligned with the image size [128 pixels]. The categories are also respecified in this code. Next, the function 'prepare' is defined, with the argument of the file path passing through. The author then created two distinct arrays specifying the color and size of the image. Afterwards, the prediction of the image is printed, along with a short informational statement regarding the nature of the diagnosis. A line of code was included in order to format the percentage of probability of Leukemia. Because the model had two options in which to classify the microscopic blood smear image, Acute Lymphoblastic Leukemia, or Healthy Blood cells, the author made sure the model predicted the 'probability of Leukemia'. If the probability of Leukemia was High[around 98%],this meant that the patient had Acute Lymphoblastic Leukemia. If the probability of Leukemia was low[around 10%-0.3%], the patient had a healthy amount of white blood cells, and did not have any blast cells. The author utilized this code throughout the testing process in order to confirm the accuracy of LeukosCognosis.

The LeukosCognosis accuracy testing procedure will be explored in detail in the "Results" Section.

Results

The Results section will explore many aspects of the LeukosCognosis Convolutional Neural Network code in a logical and statistically inclined perspective and viewpoint. Along with an in-depth analysis of the results of the LeukosCognosis Algorithm, many supplemental sources such as tables, charts, graphs and more will be perused carefully. The Results section will explore and analyze the accuracy behind the prototype LeukosCognosis Algorithm. This section will also probe upon the accuracy of the LeukosCognosis model in different stages, along with what the author did to increase the accuracy. Lastly, the accuracy of classification and the amount of time it takes to classify a test image will be investigated. The procedures that were utilized to produce these impeccable results are located in the 'Method and Materials' section. An in-depth investigation of the implications of the Results, will be featured in the "Discussion" section.

LeukosCognosis: Prototype Accuracy and Results

The author started out by prototyping the LuekosCognosis model through Wolfram Alpha, as a part of the "Software Development Design Process". This ensured that the final LeukosCognosis Algorithm was extremely accurate. In order to improve the overall prototype model, the author decided to perform experimental trials. The author performed a procedure to test the accuracy of diagnosis of the prototype model when given a test image; specifically testing for the accuracy of the probability of the diagnosis. Tweaks were made to the overall testing 'URL Image dataset', by adding in different keywords, based on the percentage of accuracy. The procedure for this evaluation can be found in the methods and materials section. For coherency, the author recorded Key word changes to each Image URL dataset,the image that was used to test the prototype model, and the diagnosis probabilities in correlation with the model draft in a table(Fig. 21). In Drafts 1-6 in the 'Healthy Blood Cell Prototype Testing' table, as the key word searches get narrower and narrower, adding in words such as 'White Blood Cells' and 'Healthy Microscopic', the dataset improves, meaning that the accuracy of the diagnosis of the image as healthy improves. There are some outliers to this notion, including Draft 4, where the prototype classifier diagnoses the image incorrectly, as 'Leukemia' instead of Healthy. The accuracy for this specific microscopic blood smear image is recorded as 55% probability of Leukemia and 45% probability of Healthy white blood cells. However, at Draft 6, the model's accuracy is extremely high. This is partially due to the authors choice of narrowed down 'URL image dataset' including the terms 'Leukemia, Microscopic, Blood Smear Image'. The prototype diagnosed the image as 'Healthy', the correct classification, and gave a 9% probability/chance of Leukemia and a 91% probability/chance of Healthy White blood cells. These percentages indicate an extremely accurate prototype model in terms of the accuracy, classification and diagnosis of Healthy White Blood cells. Drafts 1-6, in pertains to the prototype model's ability to diagnose 'Leukemia', the overarching term used when creating the model, exhibit a similar trend in probability accuracy when specifying the key words of the 'URL Image dataset'. Throughout the progression of the drafts, when given a Leukemia white blood cell smear testing image, the accuracy of the classification and the probability percentages increased. However, there are some exceptions to this increase in accuracy, Including Draft 5, where the classification is incorrect, and the diagnosis percentages are extremely inaccurate. However, by Draft , with the change in key words, the prototype model's overall accuracy increases. In order to better visualize the prototype model's accuracy correlation to the accuracy of diagnosing a healthy white blood cell smear, the author graphed the accuracy percentages (Fig 22). This graph depicts the probability of Healthy white Blood cells in red, and the probability of Leukemia white blood cells in blue throughout each individual draft and in each respective classification. The author graphed this for the Healthy White Blood Cell dataset image drafts, in order to better visualize the complicated data. As seen as an overall trend in the graph, when the classification is correct, healthy white blood cells, the accuracy and the probability percentage is higher.

The prototype model, along with the accuracy testing, was an excellent gateway to constructing the final LeukosCognosis Model. This allowed the author to understand and identify key trends in accuracy and how improving the code and changing the dataset can affect the accuracy of diagnosis.

LeukosCognosis: Final Drafting Accuracy Testing

When drafting the final LeukosCognosis Convolutional Neural Network, the author decided to perform an experimental trial in order to better improve the structure and code behind the model. Starting out with the basic code for a convolutional neural network[thoroughly explained and discussed in the methods and materials section], the author recorded the epoch accuracy, including the model loss, model accuracy, validation accuracy and validation loss. Recall that an 'epoch' entails the amount of time the model iterates over the basic code in order to 'train' and 'learn'. Model accuracy and loss refer to how accurate or inaccurate the overall model is. Validation loss and accuracy refer to the machine's reading of how accurate the model is when compared to the ability to learn off of the training data. When building a convolutional neural typically high model accuracy and validation accuracy and low model loss and validation loss indicate that the model is performing fairly well. After running the code for the specified amount of epochs, the ending accuracy readings were recorded. Based on the accuracy of the model, different additions were made to the code in order to improve the performance. These additions were also recorded. A detailed procedure is located in the 'Methods and Materials' section. The author performed this process for a total of nine drafts, and stopped only after reaching an acceptable accuracy. The Changes and Additions, Description of the changes, Epochs, and Model readings were all recorded in Figure 23, for clarity. Draft 1, signifies the basic convolutional neural network code nec easy to build a model. The author ran this code for a total of 10 epochs, or 10 iterations, which resulted in a model accuracy of 52% and a model loss of

63%. The Validation accuracy for Draft 1 is extremely low, 28%, indicating that the model did not perform very well. It is important to note that the Validation Accuracy and Validation loss along with the Model Accuracy and model loss are on their own ‘percentage scales’. This means that these percentages do not equal 100 when concatenated.The author took these readings into account and added more to the overall code in Draft 2. In Draft 2, the author added in ‘Padding’, which essentially helps the images in the training dataset have the same amount of pixels when reformatting, storing information in the neurons and creating feature extracting maps. Implementing padding increased the overall validation accuracy indicating that a positive change had occurred. In Draft 3, additional convolutional layers were added into the model’s code in order to help with image filtration and feature extraction. Draft 4, includes a similar notion, more ‘Dense Layers’ were added to improve individual neuron filters. In Draft 5, tensorflow random seed was utilized, in order to prevent model bias and help with the model initiation process. Draft 6 utilized ‘Batch Normalization’ a tensorflow function which allows the learning rate of the model to improve and increase based on a smaller set of images. The next drafts utilize Adam, an algorithm/optimizer which increases in the model’s learning and mathematical abilities, and finally the ‘Flatten’ function which improves access to data and neurons in NumPy arrays. The last draft of the LeukosCognosis model, Draft 9, includes Class weights, which helps to balance out the slightly uneven datasets that the model is training off of. The amount of epochs, or iterations was also increased. The additions and improvements to the basic Convolutional Neural Network code improved the overall accuracy readings of the model itself. The final draft had 98% Model accuracy, 0.25% model loss, 98% validation accuracy and 2% Validation loss. This indicates that the model is extremely accurate and is performing fairly well in training. These values and readings were visually represented using tensorflow ‘matplotlib graphs’. The code for these graphs can be found in Figure 24. In the code, the Matplotlib modules are imported in order to create the graph. Next, the code specifies that the model accuracy and the validation accuracy are plotted in correlation to the epochs. These graphs essentially show the distinct correlation between Validation Loss and Model Loss and the Model Accuracy and Validation Accuracy.The result of this code is featured in Figure 25 and Figure 26. These graphs depict the epochs and progression of accuracy/loss for the final and official LeukosCognosis code. Figure 25 depicts the correlation between the Model Accuracy, featured in blue and called the ‘Test’, and Validation Accuracy featured in orange and called the ‘Train’. The accuracy typically increases throughout the progression of the epochs, with the exception of epoch 2, where the accuracy drastically drops. The fact that these separate and distinct lines closely follow the same trend, indicates that the model is extremely accurate. The opposite is true for Figure 26, which depicts Model Loss and Validation Loss. As the epochs and repetitions progress and increase, Through these figures, one can easily identify a correlation between the progression of the epochs and the increase in accuracy and decrease in loss. As the number of epochs increase, the validation accuracy and model accuracy increase, which is an indication that the model is extremely accurateThe supplements that the author made, including adding layers, tensorflow modules, optimizers and more, all contributed to the increase in accuracy of the model itself.

LeukosCognosis: Diagnosis Accuracy Testing

After creating the final draft of the LeukosCognosis Convolutional Neural Network, the author decided to test out the Classification and Diagnosis Abilities of the model through a round of experimental trials. This specific round of accuracy testing judged how well the model can classify a microscopic blood smear image as either 'Acute Lymphoblastic Leukemia' or 'Healthy White Blood Cells'. The author decided to test the model by using a series of 20 images that were outside of the training dataset, 10 Acute Lymphoblastic Leukemia images and 10 Healthy White Blood Cell images. The Amount of time it takes to diagnose the image, the classification of the test image, and the probability and percentage prediction of the image were all recorded in detail. In order to test out the model the author followed a detailed procedure, included in the 'Methods and Materials Section', and utilized a specific piece of code in order to process the image through the model. The results of this experiment, recorded in a total of two tables, are located in Figures 27 and 28. These tables, contain the test image itself , the actual classification of the image, the model's classification of the image, the probability of Leukemia, and the amount of time it took to diagnose the image.It is important to note that LeukosCognosis model's predictions are recorded on the scale of the probability of Leukemia in the image. If there is a 98% chance of Leukemia in the image, it is diagnosed as Leukemia. If there is a 0.25% chance of Leukemia in the image, it is diagnosed as Healthy White Blood Cells.Figure 27, contains the Accuracy testing of the model through Acute Lymphoblastic Leukemia blood smear images. When given 10 images that had Acute Lymphoblastic Blast cells, the model classified all of them as 'Leukemia', the correct classification and diagnosis. When predicting the probability of Acute Lymphoblastic Leukemia, the model gave out percentages that were in the 95%-99% ranges, indicating that the diagnosis is extremely accurate, as the percentages of Leukemia are extremely high. The amount of time it took to diagnose the Acute Lymphoblastic Leukemia image ranged between 3-12 seconds. Figure 28, which depicts the Diagnosis testing of healthy images, shows that the probability of Leukemia, throughout each individual image was extremely low. Test Image 7, was extremely accurate in terms of predicting the amount of Leukemia in the healthy white blood cell; 0.01%. This low percentage indicates that there is an extremely miniscule chance that the patient has Leukemia. Similar to the Acute Lymphoblastic Leukemia diagnosis testing, the amount of time it took to diagnose the actual test image was a matter of seconds. A graphical representation of the Diagnosis Accuracy testings, in terms of correct classification, percentage of probability, and time, can be found in Figures 29-32. Figure 29 shows the correlation between the Acute Lymphoblastic Leukemia Test Images and the model predicted probability of Leukemia [extremely high]. Figure 30 depicts the amount of time it took for the model to diagnose and classify each individual image, in a matter of seconds. Figure 31, depicts the probability of Leukemia in correlation to the healthy blood cell smear test images. Throughout the graph, depicted in a red line, there is a trend of a low probability and percentage, ranging from 0% to 7%. The amount of time it took to diagnose the healthy cell images was also extremely low. These accuracy testings in terms of the diagnosis provided a deeper insight into how the model can classify a microscopic image.

Discussion

LeukosCognosis, a convolutional neural network to diagnose Acute Lymphoblastic Leukemia, has been proven to be accurate, efficient and productive through countless experimental trials, procedures, and more. The data to these procedures can be found in the 'Results' Section. Performing multiple tests on the model's ability to 'learn', including validation and model accuracy/loss along with improving the model's code and structure throughout the process, showed just how accurate the model was, and its overall abilities. The experimental trials that conveyed this notion to the author were specifically the 'LeukosCognosis: Diagnosis Accuracy Testing, and the LeukosCognosis: Model Accuracy testing'. In the 'LuekosCognosis: Model Accuracy testing', the author started off with the basic convolutional neural network code and found the model readings of validation accuracy loss and model accuracy loss in correlation to the amount of epochs the model trained for. The author then added more to the overall structure and code of the convolutional neural network in order to improve the accuracy and decrease the loss. The author completed this process for a total of 9 drafts, each time adding in different Tensorflow amenities to help balance out the dataset, add more layers to the model, and reduce model bias towards a certain 'categorization'. By the end of the 9 drafts, the final LeukosCognosis solution was born. The LeukosCognoisis model had 98% Model Accuracy and Vladition accuracy along with a 0.25% model loss and 2% Validation loss. This indicates that the model is performing extremely well in regards to learning rate, ability to train off of the given data, and how accurate the model will be when diagnosing an actual test image. When graphing the LeukosCognosis model[final draft], in terms of how accuracy correlated with epochs, one can clearly perceive a positive trend, as the amount of epochs increased the model and validation accuracy increased. The opposite is true for the Model and validation loss. These specific tests, when training the model, gave the author an insight into how the training process worked and what changes needed to be made. In order to gain an in-depth understanding of how well the model diagnoses a test image, the author decided to perform the 'LeukosCognosis: Diagnosis Accuracy testing'. These tests gave the author an insight into how well the LeukosCognosis model could diagnose a microscopic blood smear image in relation to the correct classification and the probability of Leukemia. After giving the model 20 test images, 10 Acute Lymphoblastic Leukemia and 10 Healthy whtie blood cells smears, the model correctly diagnosed/classified each image. The probability of Leukemia in the Acute Lymphoblastic Leukemia blood smears was predicted to be around 98% for all test images, indicating there is an extremely high chance of Leukemia. When diagnosing the Healthy Blood Cell smears, the model classified all 10 of these images a Healthy, the correct classification, and gave extremely low percentages for the probability of Leukemia. This means that there is an extremely low chance, 0.25%-2% chance that there is Acute Lymphoblastic Leukemia in the blood smear.The meer fact that the model was able to properly diagnose all of these images, indicates the accuracy of LeukosCognosis. Throughout these tests, the amount of time it took to substantially diagnose the image ranged from 2 seconds- 14 seconds in total. This means that LeukosCognosis is extremely efficient in terms of the oncologist receiving a diagnosis immediately after a microscopic image of the blood smear is taken. LeukosCognosis's astounding results on these tests indicate the future of diagnosis testing of Acute Lymphoblastic Leukemia. When compared to diagnosis companies such as Labcorp or Quest, LeukosCognsois is far better in all aspects including the overall accuracy of the algorithm and the easy to use features. Labcorp and Quest employ human diagnosis, meaning that their ability to diagnose Leukemia based solely off of the Human eye is only around 30%-40% accurate. This means that if a patient sends in a blood smear test, they may receive a fault and incorrect result. LeukosCognosis is 98% accurate meaning that a patient can trust the results received, and immediately continue with the next steps in confirming the diagnosis of Acute Lymphoblastic Leukemia[bone marrow test], and continue with the treatment steps. LeukosCognosis is also extremely quick and efficient when compared to companies such as Labcorp and Quest. Labcorp and Quest can take a matter of weeks to diagnose the blood smear test and return it to the patient. This means that there is a high chance that a patient's window of treatment may pass, effectively decreasing their chance to recover from Acute Lymphoblastic Leukemia. LeukosCognosis cuts out the 'middle-man' of the labs. Instead, the oncologist can take a blood smear test, take a microscopic image of the test, upload it to the computer and receive a diagnosis in a matter of seconds. This efficiency allows for easier treatment for the patient. In the future, the author would like to deploy this model into the medical field to provide oncologists with access to this useful technology. In order to successfully deploy this model into the medical field, the author would like to create a User Interface to the model itself. Creating the user interface will make uploading the image to the model easier, and will prevent the oncologists from directly accessing or tampering with the code. The user interface would most likely be created with Flask, an easy web deployment software. By creating a user interface for the model and effectively deploying it into the medical field LeukosCognosis has the ability of positively impacting patients with Acute Lymphoblastic Leukemia.

LeukoCytosis can truly have an impact on the medical field, and improve the diagnosis rates of this fatal and fast acting condition. Deploying this machine learning algorithm into the medical field will cut the cost of lab tests, remove the 'midle-man', provide an accurate diagnosis and allow for easier access to testing.

Conclusion

Every 9 minutes a person in the United States dies from Leukemia, or cancer of the blood and bone marrow. Essentially, Leukemia is a cancer of the blood and bone marrow and occurs when one's body rapidly produces abnormal white blood cells, which decreases the body's ability to fight infections. If Leukemia is not immediately diagnosed, as symptoms are often lucrative, this disease can act quickly. According to the *Leukemia & Lymphoma Society*, diagnosis and lab blood smear sample tests can take weeks to return, decreasing a patient's ability and chances to survive. The need for a quick, effective, and proper diagnosis tool is urgent. In an effort to solve this problem, LeukosCognosis was created. LeukosCognosis is a Machine Learning Algorithm that can diagnose Acute Lymphoblastic Leukemia[ALL] on the spot with 98% accuracy. This algorithm can accurately diagnose Leukemia in seconds, a juxtaposition to the weeks that a lab Leukemia diagnosis may take. All the oncologist would have to do is take a microscopic image of the suspected white blood cell smear, and upload it to the model. Within seconds, the model will accurately diagnose the sample and predict the probability of Acute Lymphoblastic Leukemia. LeukosCognosis is unique as it provides a one-stop solution to long and often inaccurate lab tests and can be performed almost immediately. The model was coded using the Tensorflow environment, the Python programming language, and Jupyter Notebook. The LeukosCognosis algorithm is 98% accurate and returns diagnosis results almost immediately. The resources that were utilized in constructing the LeukosCognosis Machine Learning algorithm, consisted primarily of software and coding programs. The Machine Learning Software that was used to code the Convolutional Neural Network behind the LeukosCognosis Algorithm, was Tensorflow, through a Jupyter notebook interface. Keras and Python commands were also utilized to construct the model. Wolfram Alpha code was used to build a solid prototype of the model, for design and testing purposes. LeukosCognosis eliminates the need for a lab analysis of a blood smear; all the Oncologist[doctor who treats cancer] would have to do is take a microscopic image of the white blood cell smear, and upload it to the model. Within seconds, an accurate diagnosis will appear, along with the percentage pertaining to the probability of Leukemia. With an accurate confirmation of Acute Lymphoblastic Leukemia, bone marrow testing can proceed, and life-saving treatment can begin immediately. Companies such as LabCorp and Quest, which perform blood smear lab tests, are the LeukosCognosis Algorithms' biggest competitors. LabCorp and Quest use the Human eye to diagnose blood smears, rendering accuracy at a dismal 30-40%. However, LeukoCytosis is unique as it cuts out the 'middleman' of a costly Lab company and delivers results in under seconds. This Algorithm can be used anywhere, making it

accessible, cost-effective, and productive.It is important to note that the LuekosCognosis algorithm had to undergo a great deal of experimentation, testing, prototyping, model revisions, and reconstruction of the Convolutional Neural Network. The evaluations and experiments were conducted in order to perfect the overall medical precision of the model. The training data set for the model was also regulated throughout the coding process. Continuous research on Acute Lymphoblastic Leukemia was also necessary, to ensure the utmost medical validity. Once this model is deployed in the medical field, it will save time, money, and lives. This convolutional network-based image classifier is built on several 'layers', including dense layers, pooling layers, Max pooling layers, and conv3D layers. The author trained this model using hundreds of microscopic blood smear images of both health White blood Cells and Leukemia infected white blood cells. These microscopic images were acquired from Fabio Scotti, an esteemed Leukemia researcher. This machine learning convolutional neural network was built to provide accurate Leukemia diagnosis results, eliminate the waste of time, and provide a pre-lab analysis of the blood cells. The author hopes to one day deploy this tool as an API into the Medical field through Flask, to make it more accessible to healthcare workers. Deploying this to the medical field will allow for greater future implications of LeukosCognosis, and will create a greater impact. LeukosCognosis is cost-efficient and saves an abundance of time when diagnosing Acute Lymphoblastic Leukemia.This model will improve the diagnosis procedure and allow for greater positive medical impact on the patient.
LeukosCognosis will save time, money, and lives.

Figures

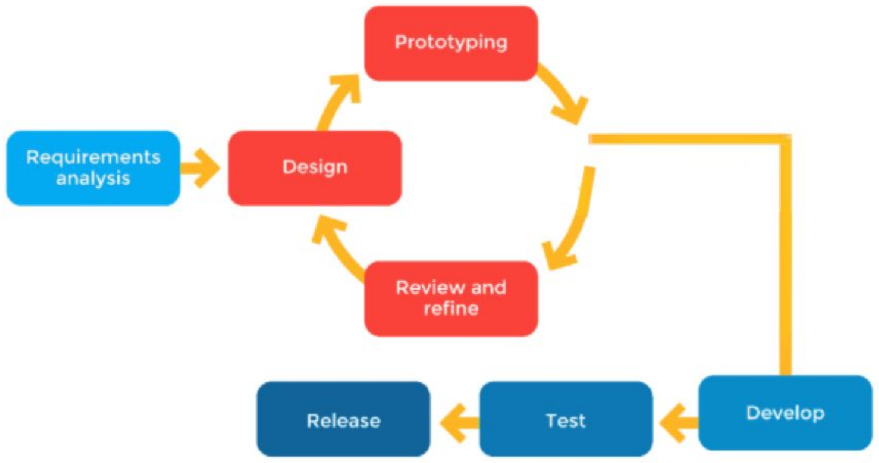


Figure 1-Software Development Design Process. Depicts the programming and software design process in relation to the progressing development of the solution.
Source: 2019. *Prototyping Design Process-Software Development.*[image] Available at: <https://www.plutora.com/blog/software-development-life-cycle-making-sense-of-the-different-methodologies>[Accessed 8 May 2020] Note: Author made edits to the image for clarity

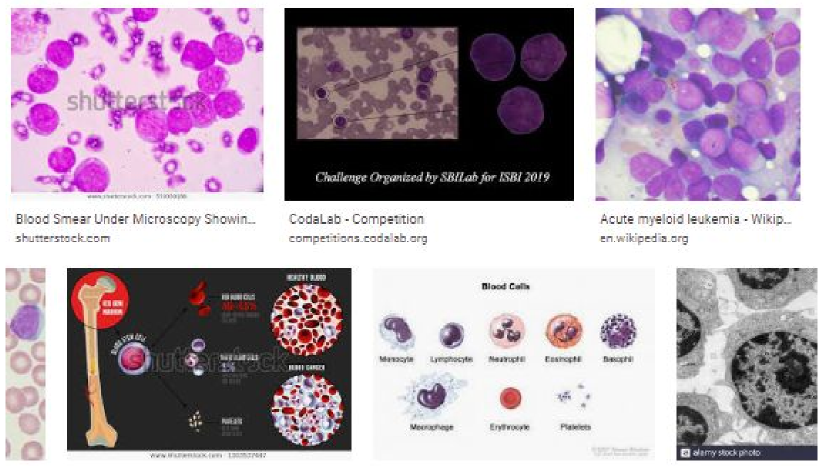
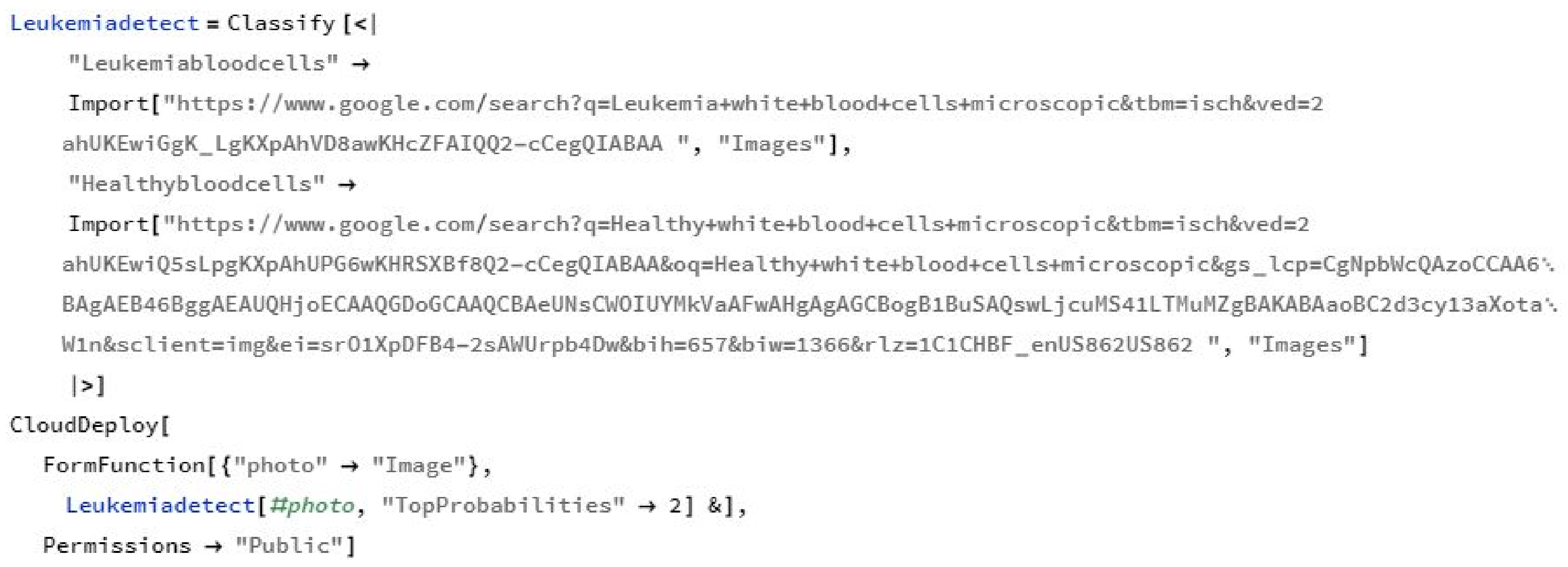


Figure 2-Sample Images in Web Search dataset. Displays a few images in the Websearch ‘dataset’ utilized in the Wolfram Alpha Prototype Model.
Source: Generated by Google Image Search Engine. Available at: <Leukemia White Blood Cells Microscopic Images> [Accessed 6 May 2020]



```
Leukemiadetect = Classify [<|
    "Leukemiabloodcells" →
    Import["https://www.google.com/search?q=Leukemia+white+blood+cells+microscopic&tbm=isch&ved=2
   ahUKEwiGgK_LgKXpAhVD8awKHcZFAIQQ2-cCegQIABAA ", "Images"],
    "Healthybloodcells" →
    Import["https://www.google.com/search?q=Healthy+white+blood+cells+microscopic&tbm=isch&ved=2
   ahUKEwiQ5sLpgKXpAhUPG6wKHRSXBf8Q2-cCegQIABAA&oq=Healthy+white+blood+cells+microscopic&gs_lcp=CgNpbWcQAzoCCAA6⋱
   BAgAEB46BggAEAUQHjoECAAQGDoGCAAQCBAeUNsCWOIUYMkVaAFwAHgAgAGCBogB1BuSAQswLjcuMS41LTMuMZgBAKABAaoBC2d3cy13aXota⋱
   W1n&sclient=img&ei=srO1XpDFB4-2sAWUrpb4Dw&bih=657&biw=1366&rlz=1C1CHBF_enUS862US862 ", "Images"]
    |>]
CloudDeploy[
  FormFunction[{"photo" → "Image"},
   Leukemiadetect[#photo, "TopProbabilities" → 2] &],
  Permissions → "Public"]
```

Figure 3-Image Classifier Prototype Code. Prototype Image Classifier Model generated by the author utilizing Wolfram Alpha, a popular programming tool.
Source: Generated by the Author. 2019. *Wolfram Alpha Image Classifier Prototype Code.* [Accessed 8 May 2020]

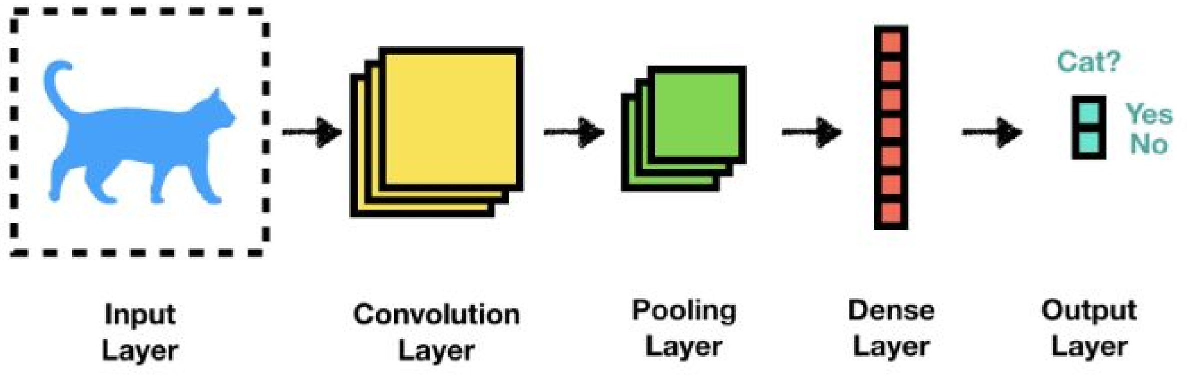


Figure 4-Convolutional Neural Network Basic Process. Depicts the basic outline and process of a CNN model, from an input image to an output layer answer, along with the layers in between.

Source: 2019. *Convolutional Neural Network-Layman's Terms*. [image] Available at: <https://towardsdatascience.com/convolutional-neural-network-a-step-by-step-guide-a8b4c88d6943 > [Accessed 4 May 2020].

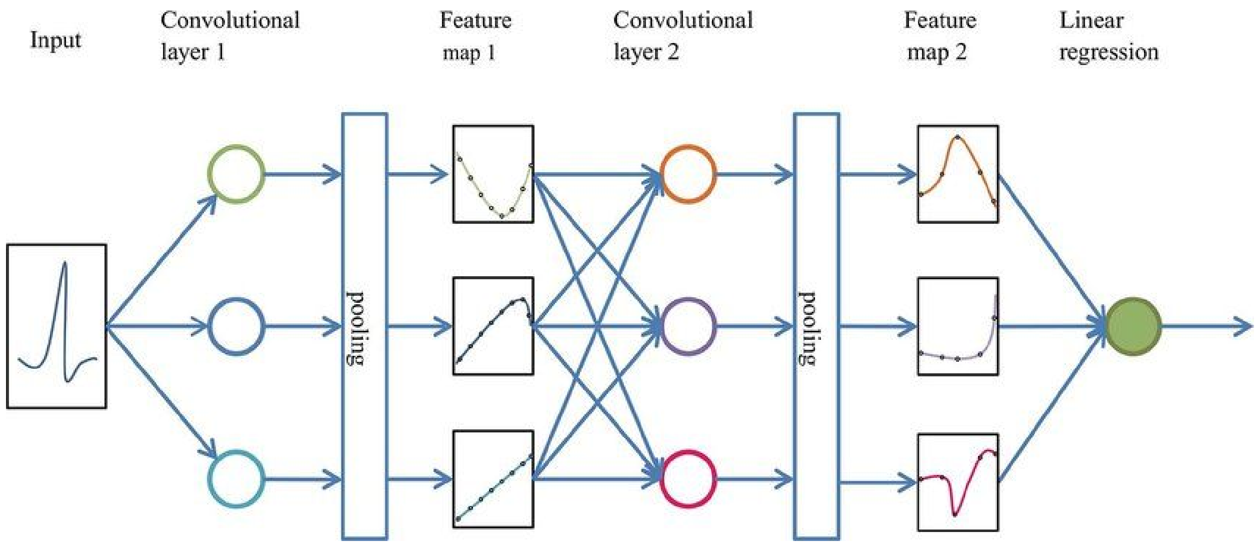


Figure 5- Neural Network Map. Depicts the flow of Neurons and utilization of neurons in a convolutional neural network.
Source: 2014.*Structure of Convolutional Neural Network-Neurons*. [image] Available at: <https://www.researchgate.net/figure/The-structure-of-a-convolutional-neural-network-adopted-in-this-paper-The-circles_fig2_286240187>[Acessed 10 May 2020]

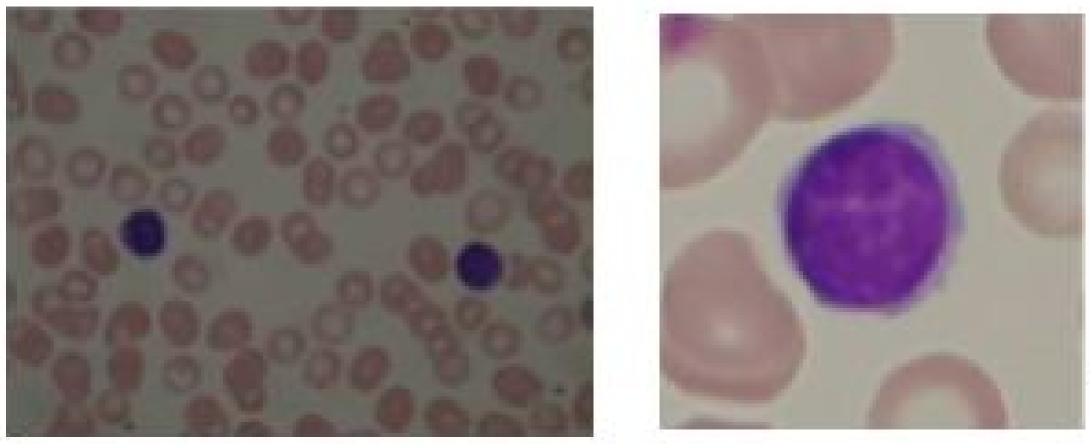
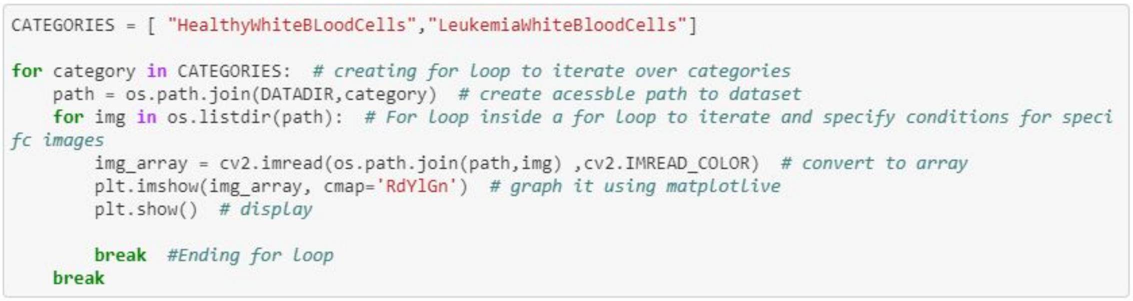


Figure 6-Dataset Images. The image to the left depicts a non-healthy cell from ALL patients, while the image to the right depicts a healthy blast cell. Note: This is part of the testing dataset in LeukosCognosis.
Source: 2010.*Dataset Sample.*[image] Available at:
<https://homes.di.unimi.it/scotti/all/>


```
CATEGORIES = [ "HealthyWhiteBLoodCells","LeukemiaWhiteBloodCells"]

for category in CATEGORIES:  # creating for loop to iterate over categories
    path = os.path.join(DATADIR,category)  # create acessble path to dataset
    for img in os.listdir(path):  # For loop inside a for loop to iterate and specify conditions for speci
fc images
        img_array = cv2.imread(os.path.join(path,img) ,cv2.IMREAD_COLOR)  # convert to array
        plt.imshow(img_array, cmap='RdYlGn')  # graph it using matplotlive
        plt.show()  # display

        break  #Ending for loop
    break
```

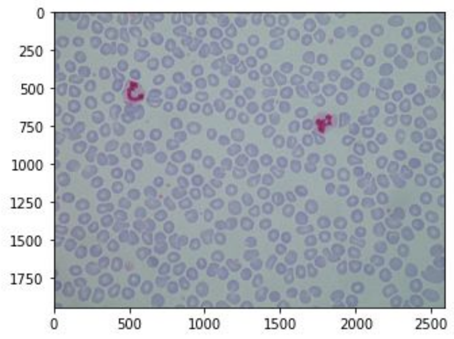


Figure 7- Depicts the code for the imported images, specifies categories, and Color scale of the image.
Source: 2020. *LeukosCognosis:Code.* [image] The code in this image was generated and created by the author [Sirihaasa Nallamothu].

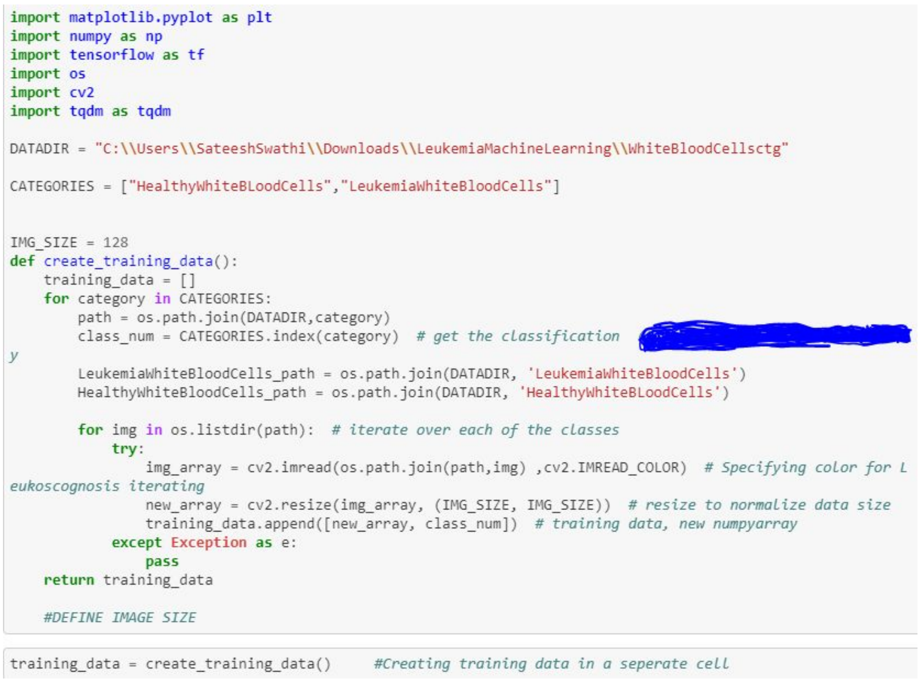
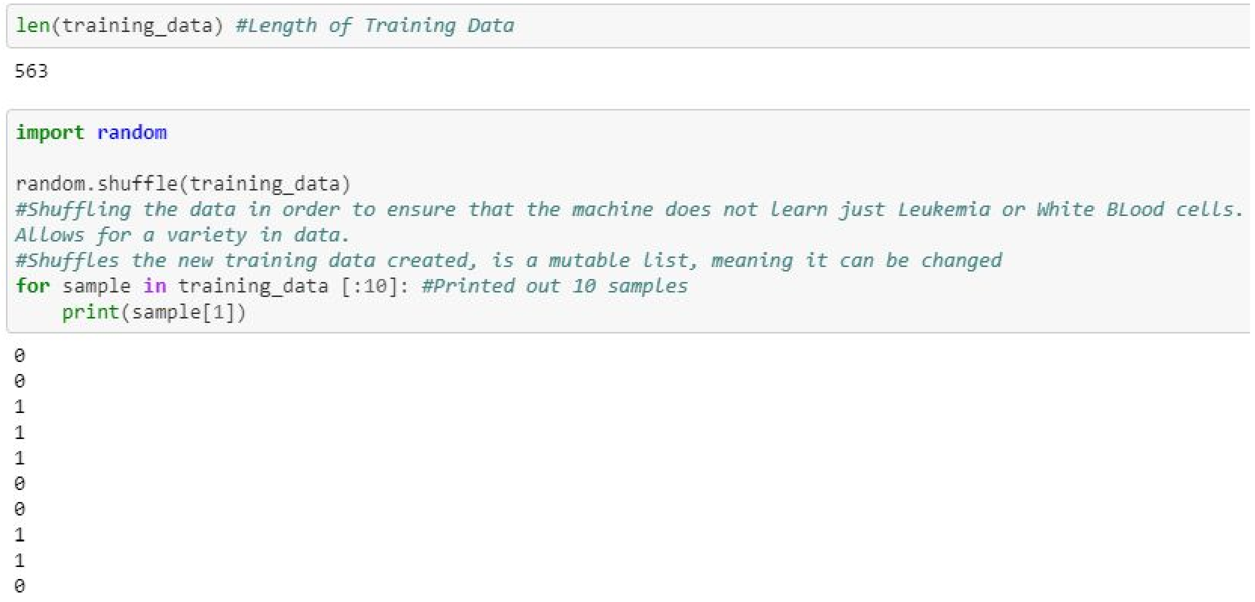
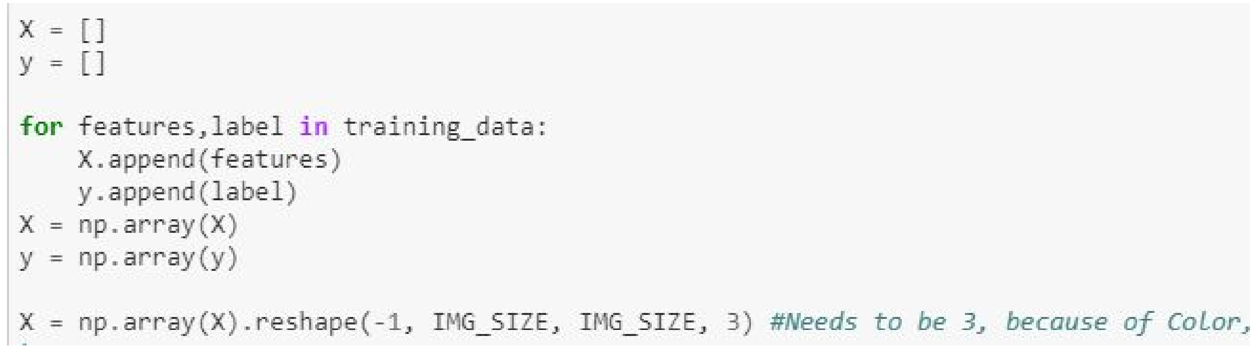

```
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
import os
import cv2
import tqdm as tqdm

DATADIR = "C:\\Users\\SateeshSwathi\\Downloads\\LeukemiaMachineLearning\\WhiteBloodCellsctg"

CATEGORIES = ["HealthyWhiteBLoodCells","LeukemiaWhiteBloodCells"]


IMG_SIZE = 128
def create_training_data():
    training_data = []
    for category in CATEGORIES:
        path = os.path.join(DATADIR,category)
        class_num = CATEGORIES.index(category)  # get the classification [redacted]
y
        LeukemiaWhiteBloodCells_path = os.path.join(DATADIR, 'LeukemiaWhiteBloodCells')
        HealthyWhiteBloodCells_path = os.path.join(DATADIR, 'HealthyWhiteBLoodCells')

        for img in os.listdir(path):  # iterate over each of the classes
            try:
                img_array = cv2.imread(os.path.join(path,img) ,cv2.IMREAD_COLOR)  # Specifying color for L
eukoscognosis iterating
                new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))  # resize to normalize data size
                training_data.append([new_array, class_num])  # training data, new numpyarray
            except Exception as e:
                pass
    return training_data

    #DEFINE IMAGE SIZE
```
```
training_data = create_training_data()     #Creating training data in a seperate cell
```

Figure 8-Depicts the code for the creation of the training data.
Source: 2020. *LeukosCognosis:Code.* [image] The code in this image was generated and created by the author [Sirihaasa Nallamothu].



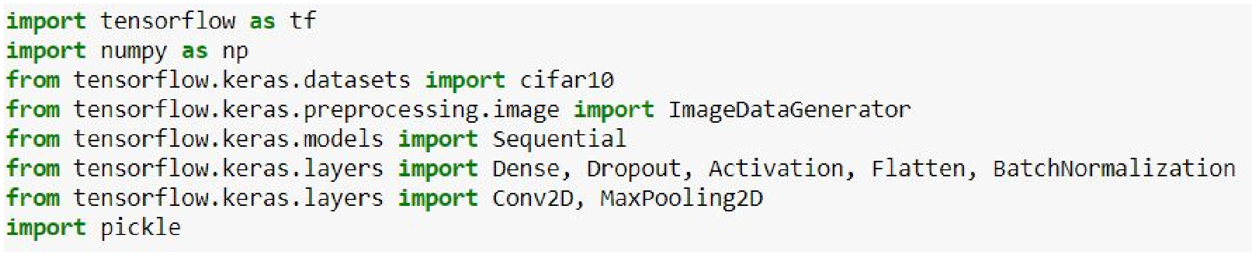
```
len(training_data) #Length of Training Data

563

import random

random.shuffle(training_data)
#Shuffling the data in order to ensure that the machine does not learn just Leukemia or White BLood cells.
Allows for a variety in data.
#Shuffles the new training data created, is a mutable list, meaning it can be changed
for sample in training_data [:10]: #Printed out 10 samples
    print(sample[1])

0
0
1
1
1
0
0
1
1
0
```

Figure 9-Depicts the code behind shuffling the training data to prevent bias.
Source: 2020. *LeukosCognosis:Code.* [image] The code in this image was generated and created by the author [Sirihaasa Nallamothu].



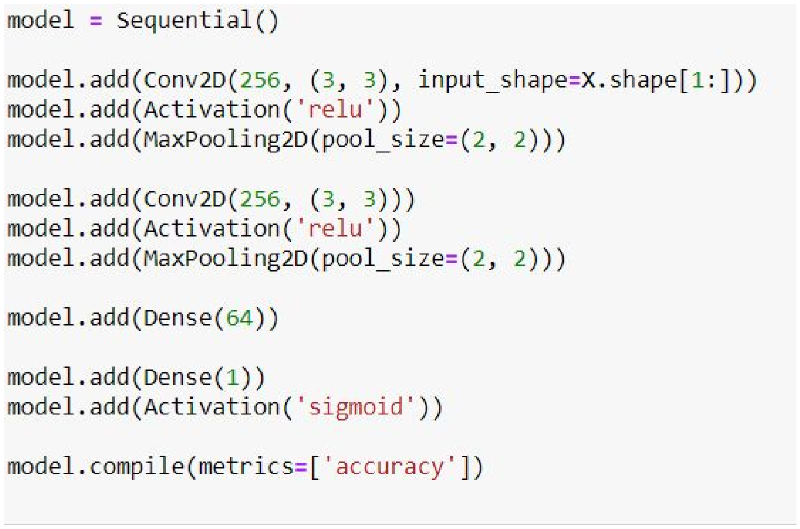
```
X = []
y = []

for features,label in training_data:
    X.append(features)
    y.append(label)
X = np.array(X)
y = np.array(y)

X = np.array(X).reshape(-1, IMG_SIZE, IMG_SIZE, 3) #Needs to be 3, because of Color,
```

Figure 10-Depicts the code behind creating the NumPy arrays for features and labels.
Source: 2020. *LeukosCognosis:Code.* [image] The code in this image was generated and created by the author [Sirihaasa Nallamothu].



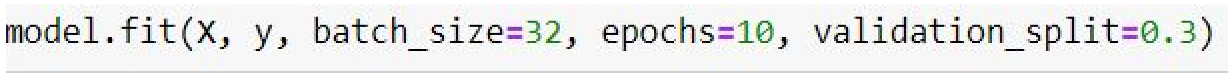
```python
import tensorflow as tf
import numpy as np
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, BatchNormalization
from tensorflow.keras.layers import Conv2D, MaxPooling2D
import pickle
```

Figure 11-Depicts the Tensorflow modules that were imported in the Jupyter Notebook to create the convolutional neural network.
Source: 2020. *LeukosCognosis:Code.* [image] The code in this image was generated and created by the author [Sirihaasa Nallamothu].



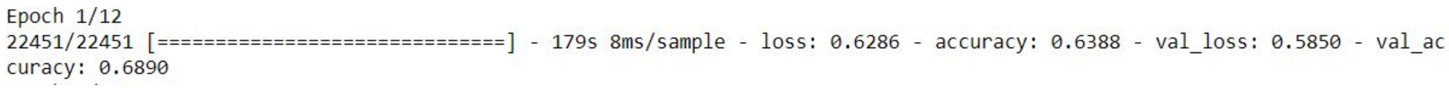
```python
model = Sequential()

model.add(Conv2D(256, (3, 3), input_shape=X.shape[1:]))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(256, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Dense(64))

model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(metrics=['accuracy'])
```

Figure 12-Depicts the First draft and layers of the convolutional neural network. The layers and capabilities featured in this first draft are extremely simple- the baseline of a convolutional neural network.
Source: 2020. *LeukosCognosis:Code.* [image] The code in this image was generated and created by the author [Sirihaasa Nallamothu].



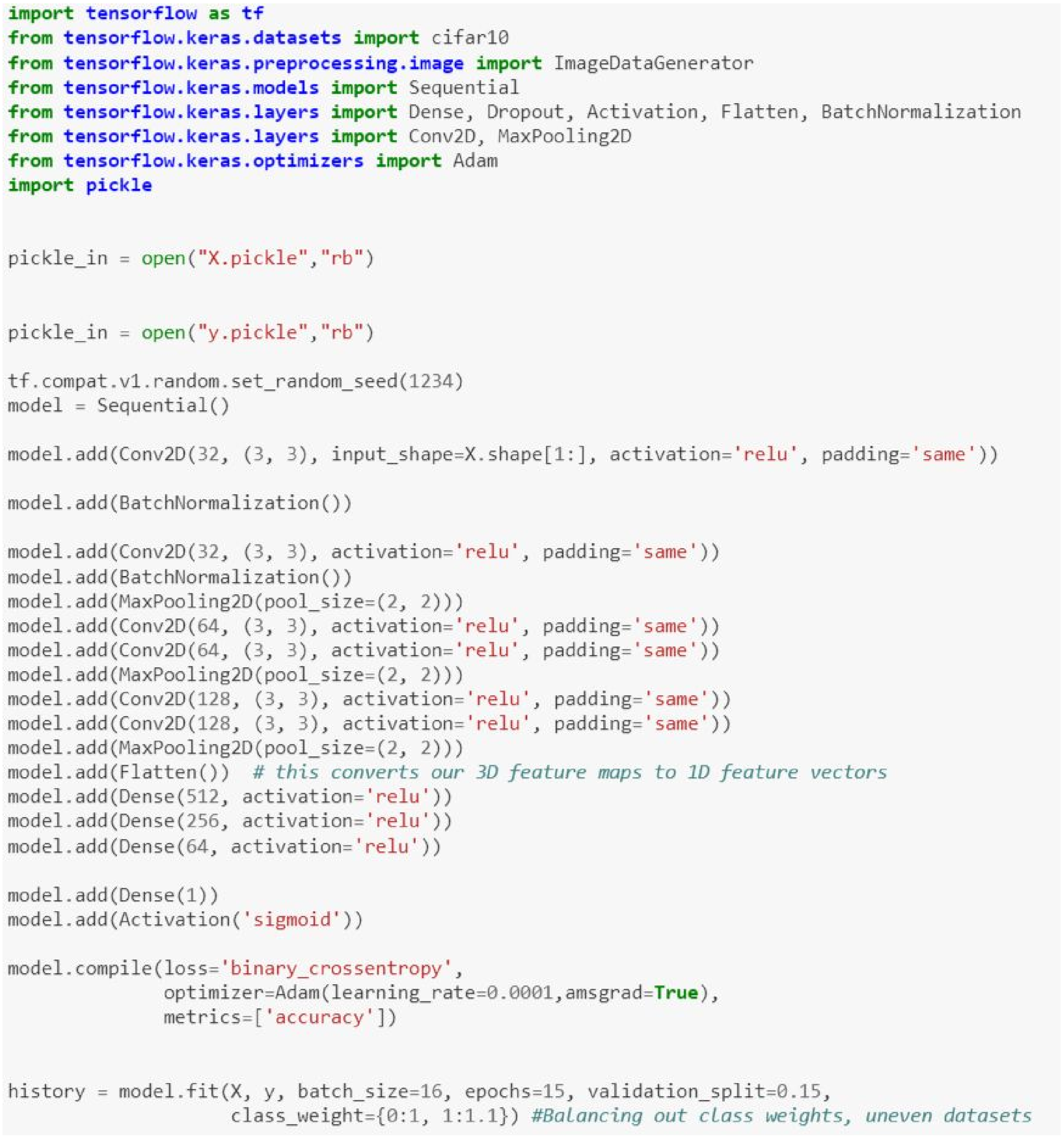
```python
model.fit(X, y, batch_size=32, epochs=10, validation_split=0.3)
```

Figure 13-Depicts the last line needed in a basic convolutional neural network- necessary to specify the parameters when training the model.
Source: 2020. *LeukosCognosis:Code.* [image] The code in this image was generated and created by the author [Sirihaasa Nallamothu].



```
Epoch 1/12
22451/22451 [==============================] - 179s 8ms/sample - loss: 0.6286 - accuracy: 0.6388 - val_loss: 0.5850 - val_ac
curacy: 0.6890
```

Figure 14-Depicts and example of an epoch, along with the displayed metrics. Validation loss and accuracy along with model loss and accuracy were also included.
Source: 2020. *LeukosCognosis:Code.* [image] The code in this image was generated and created by the author [Sirihaasa Nallamothu].



```
import tensorflow as tf
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, BatchNormalization
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.optimizers import Adam
import pickle


pickle_in = open("X.pickle","rb")


pickle_in = open("y.pickle","rb")

tf.compat.v1.random.set_random_seed(1234)
model = Sequential()

model.add(Conv2D(32, (3, 3), input_shape=X.shape[1:], activation='relu', padding='same'))

model.add(BatchNormalization())

model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())  # this converts our 3D feature maps to 1D feature vectors
model.add(Dense(512, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dense(64, activation='relu'))

model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer=Adam(learning_rate=0.0001,amsgrad=True),
              metrics=['accuracy'])


history = model.fit(X, y, batch_size=16, epochs=15, validation_split=0.15,
                    class_weight={0:1, 1:1.1}) #Balancing out class weights, uneven datasets
```

Figure 15-Depicts the final draft of the LeukosCognosis Convolutional Neural Network code after a procedure was implemented and the model code was perfected. This code yields 98% accurate results.
Source: 2020. *LeukosCognosis:Code.* [image] The code in this image was generated and created by the author [Sirihaasa Nallamothu].

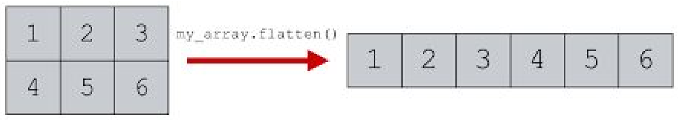
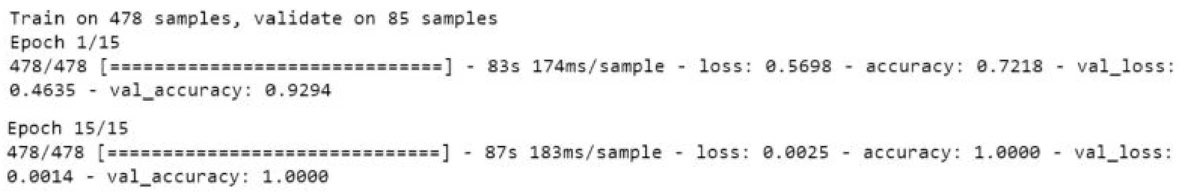
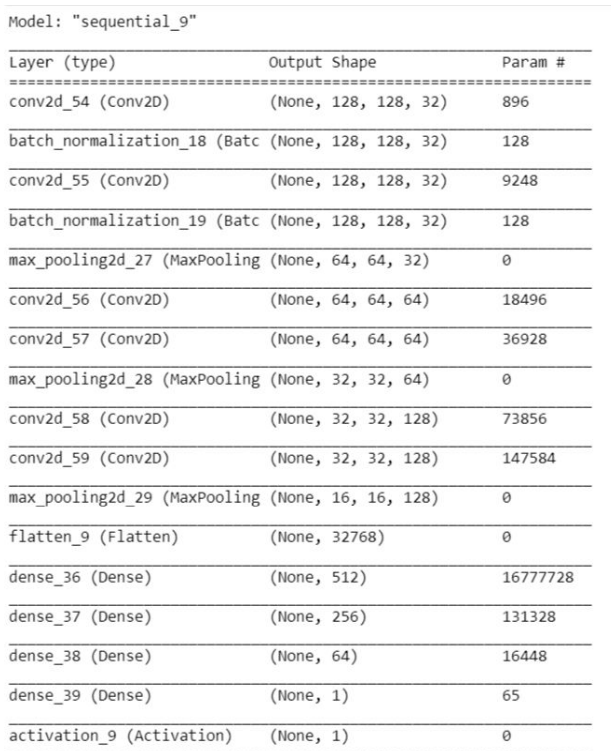


Figure 16-Depicts the effect of Flatten() to a NumPy array, changing a multidimensional array into a 1D array.
Source: 2019. NumPy Arrays flatten. [image]. Available at: <https://www.sharpsightlabs.com/blog/numpy-flatten>[Acessed 9 June 2020]



```
Train on 478 samples, validate on 85 samples
Epoch 1/15
478/478 [==============================] - 83s 174ms/sample - loss: 0.5698 - accuracy: 0.7218 - val_loss:
0.4635 - val_accuracy: 0.9294

Epoch 15/15
478/478 [==============================] - 87s 183ms/sample - loss: 0.0025 - accuracy: 1.0000 - val_loss:
0.0014 - val_accuracy: 1.0000
```

Figure 17-Depicts the epoch iterations of the final LeukosCognosis code. The code was 'iterated over' 15 times. The validation loss decreased throughout the repetitions and the accuracy increased. This image is a side by side comparison of the first epoch and the last epoch.
Source: 2020. *LeukosCognosis:Code.* [image] The code in this image was generated and created by the author [Sirihaasa Nallamothu].



```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_54 (Conv2D)           (None, 128, 128, 32)      896
_________________________________________________________________
batch_normalization_18 (Batc (None, 128, 128, 32)      128
_________________________________________________________________
conv2d_55 (Conv2D)           (None, 128, 128, 32)      9248
_________________________________________________________________
batch_normalization_19 (Batc (None, 128, 128, 32)      128
_________________________________________________________________
max_pooling2d_27 (MaxPooling (None, 64, 64, 32)        0
_________________________________________________________________
conv2d_56 (Conv2D)           (None, 64, 64, 64)        18496
_________________________________________________________________
conv2d_57 (Conv2D)           (None, 64, 64, 64)        36928
_________________________________________________________________
max_pooling2d_28 (MaxPooling (None, 32, 32, 64)        0
_________________________________________________________________
conv2d_58 (Conv2D)           (None, 32, 32, 128)       73856
_________________________________________________________________
conv2d_59 (Conv2D)           (None, 32, 32, 128)       147584
_________________________________________________________________
max_pooling2d_29 (MaxPooling (None, 16, 16, 128)       0
_________________________________________________________________
flatten_9 (Flatten)          (None, 32768)             0
_________________________________________________________________
dense_36 (Dense)             (None, 512)               16777728
_________________________________________________________________
dense_37 (Dense)             (None, 256)               131328
_________________________________________________________________
dense_38 (Dense)             (None, 64)                16448
_________________________________________________________________
dense_39 (Dense)             (None, 1)                 65
_________________________________________________________________
activation_9 (Activation)    (None, 1)                 0
```

Figure 18-Depicts the summary of the LeukosCognosis Convolutional Neural Network, in terms of the layers, functions, and neurons implemented in the code. The 'Params' short for parameters are depicted, which specify the amount of data being transferred from neuron to neuron.
Source: 2020. *LeukosCognosis:Code.* [image] The code in this image was generated and created by the author [Sirihaasa Nallamothu].

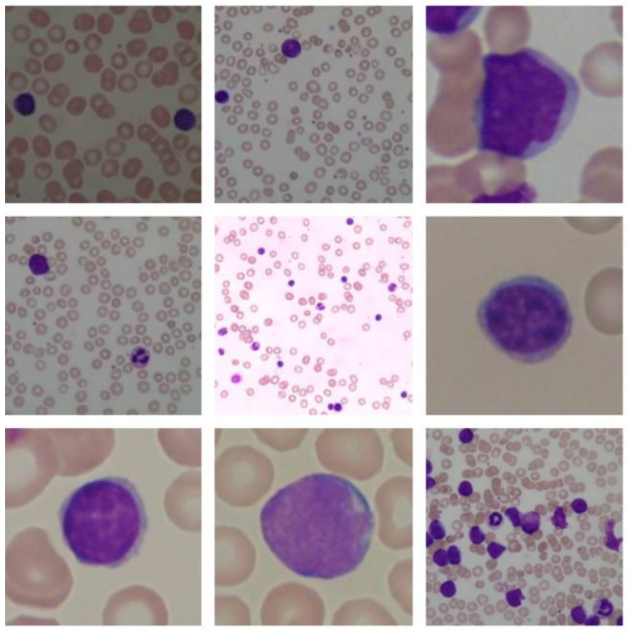
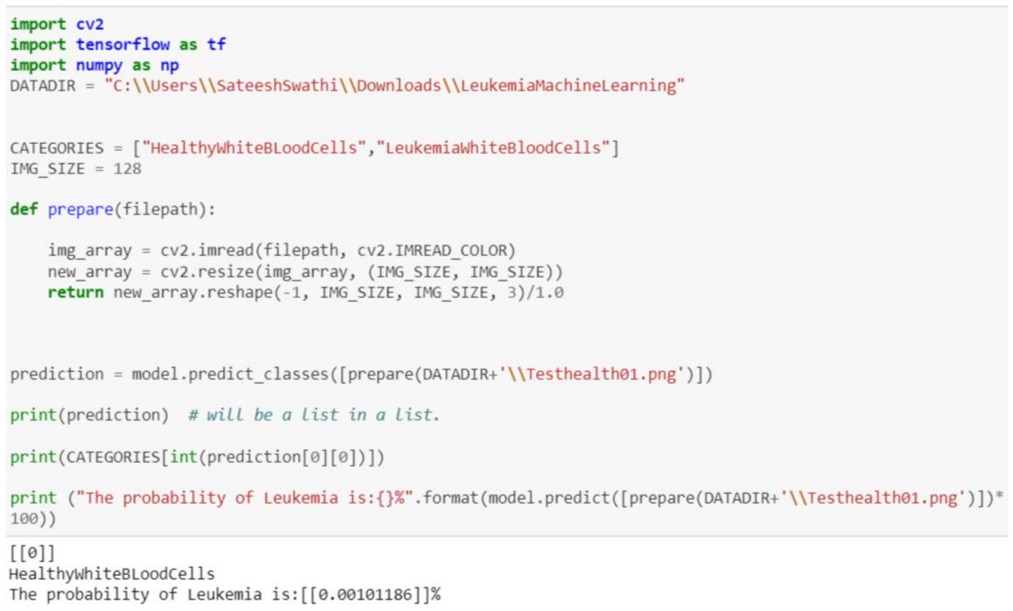
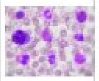
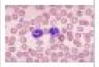
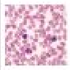
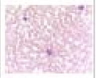
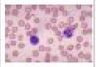


Figure 19-Depicts a variety of testing images, not part of the training dataset, that the author utilized in order to test the overall categorization accuracy of the model.
Source: 2010.*Dataset Sample.*[image] Available at:
<https://homes.di.unimi.it/scotti/all/>[Acessed 9 June 2020] Note: The author arranged these images in an appealing way.



```
import cv2
import tensorflow as tf
import numpy as np
DATADIR = "C:\\Users\\SateeshSwathi\\Downloads\\LeukemiaMachineLearning"


CATEGORIES = ["HealthyWhiteBLoodCells","LeukemiaWhiteBloodCells"]
IMG_SIZE = 128

def prepare(filepath):

    img_array = cv2.imread(filepath, cv2.IMREAD_COLOR)
    new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
    return new_array.reshape(-1, IMG_SIZE, IMG_SIZE, 3)/1.0


prediction = model.predict_classes([prepare(DATADIR+'\\Testhealth01.png')])

print(prediction)  # will be a list in a list.

print(CATEGORIES[int(prediction[0][0])])

print ("The probability of Leukemia is:{}%".format(model.predict([prepare(DATADIR+'\\Testhealth01.png')])*
100))
[[0]]
HealthyWhiteBLoodCells
The probability of Leukemia is:[[0.00101186]]%
```

Figure 20-Depicts the code utilized to test the accuracy of LeukosCognosis. In this image, a healthy blood cell is tested, with the probability of Leukemia at a very low percentage.
Source: 2020. *LeukosCognosis:Code.* [image] The code in this image was generated and created by the author [Sirihaasa Nallamothu].

LeukosCognosis: Prototype Accuracy Testing: Healthy Blood Cells					
Trial #	Keyword[Changes/Additions to Image URL Dataset]	Model Classification	Probability of Leukemia	Probability of Healthy Cells	Test Image
Draft 1	Microscopic, Smear	Leukemia	76%	24%	
Draft 2	BLood Smear Image	Leukemia	63%	37%	
Draft 3	Blood film smear	Healthy	30%	70%	
Draft 4	High Resolution, Smear	Leukemia	55%	45%	
Draft 5	White Blood Cells, Healthy,Microscopic Smear	Healthy	20%	80%	
Draft 6	White Blood Cells, Healthy,Microscopic Smear	Healthy	9%	91%	

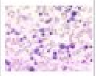
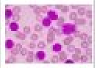
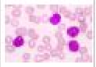
LeukosCognosis: Prototype Accuracy Testing:Leukemia White Blood Cells					
Trial #	Keyword[Changes to Dataset]	Model Classification	Probability of Leukemia	Probability of Healthy Cells	
Draft 1	Leukemia, Blood Smear	Healthy	30%	70%	
Draft 2	Abnormal Lymphocytes, Microscopic	Healthy	45%	55%	
Draft 3	Leukemia Blast cell, Image	Leukemia	70%	30%	
Draft 4	Microscopic Image Leukemia	Leukemia	65%	35%	
Draft 5	Leukemia Blood film smear	Healthy	40%	60%	
Draft 6	Leukemia Microscopic Blood Smear Image	Leukemia	81%	19%	

Figure 21-Depicts the table utilized to record the progression of accuracy and change in keywords of the LeukosCognosis Prototype Model.
Source: 2020. *LeukosCognosis: Prototype Table.* [image] The code in this image was generated and created by the author [Sirihaasa Nallamothu].

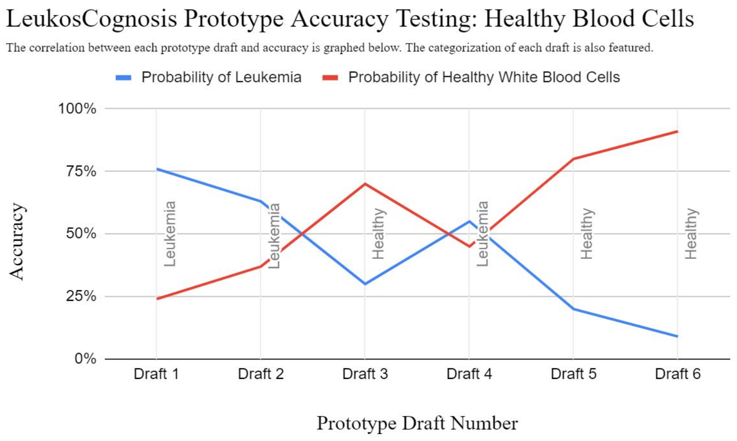
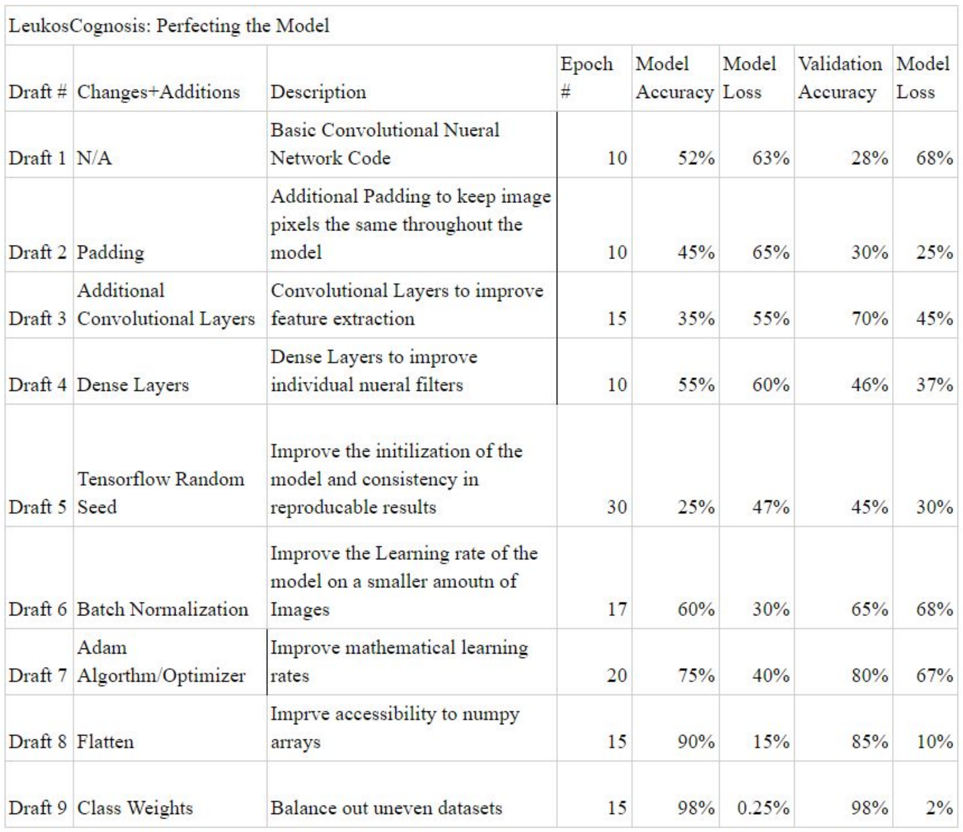


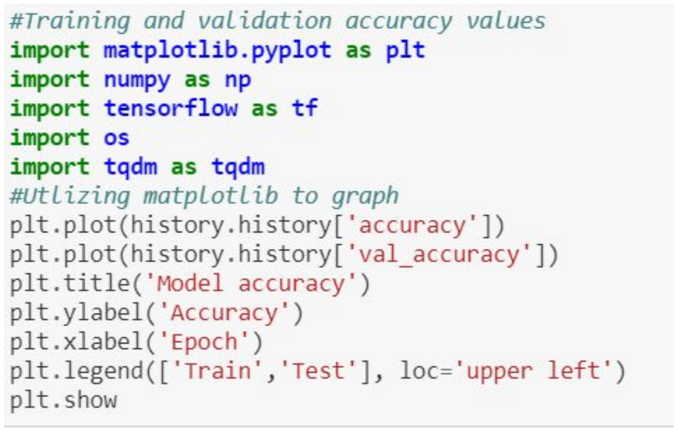
Figure 22-Depicts the LeukosCognosis Prototype Graph in correlation to the accuracy of the model.
Source: 2020. *LeukosCognosis: Prototype Graph.* [image] The code in this image was generated and created by the author [Sirihaasa Nallamothu].



LeukosCognosis: Perfecting the Model

Draft #	Changes+Additions	Description	Epoch #	Model Accuracy	Model Loss	Validation Accuracy	Model Loss
Draft 1	N/A	Basic Convolutional Nueral Network Code	10	52%	63%	28%	68%
Draft 2	Padding	Additional Padding to keep image pixels the same throughout the model	10	45%	65%	30%	25%
Draft 3	Additional Convolutional Layers	Convolutional Layers to improve feature extraction	15	35%	55%	70%	45%
Draft 4	Dense Layers	Dense Layers to improve individual nueral filters	10	55%	60%	46%	37%
Draft 5	Tensorflow Random Seed	Improve the initilization of the model and consistency in reproducable results	30	25%	47%	45%	30%
Draft 6	Batch Normalization	Improve the Learning rate of the model on a smaller amoutn of Images	17	60%	30%	65%	68%
Draft 7	Adam Algorthm/Optimizer	Improve mathematical learning rates	20	75%	40%	80%	67%
Draft 8	Flatten	Imprve accessibility to numpy arrays	15	90%	15%	85%	10%
Draft 9	Class Weights	Balance out uneven datasets	15	98%	0.25%	98%	2%

Figure 23-Depicts the LeukosCognosis Perfecting the model table
Source: 2020. *LeukosCognosis: Perfecting the Model Table.* [image] The code in this image was generated and created by the author [Sirihaasa Nallamothu].



```
#Training and validation accuracy values
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
import os
import tqdm as tqdm
#Utlizing matplotlib to graph
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train','Test'], loc='upper left')
plt.show
```

Figure 24-Depicts the code utilized to generate the Matplotlib graph[Figure 25].
Source: 2020. *LeukosCognosis: Code.* [image] The code in this image was generated and created by the author [Sirihaasa Nallamothu].

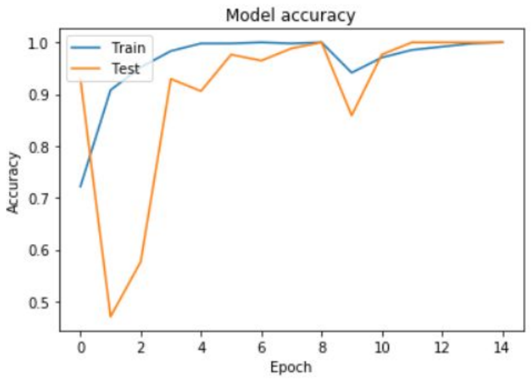


Figure 25-Depicts the Model Accuracy and Validation of the Final LeukosCognosis Machine Learning Algorithm throughout the epochs.
Source: 2020. *LeukosCognosis: Graph.* [image] The graph in this image was generated and created by the author [Sirihaasa Nallamothu].

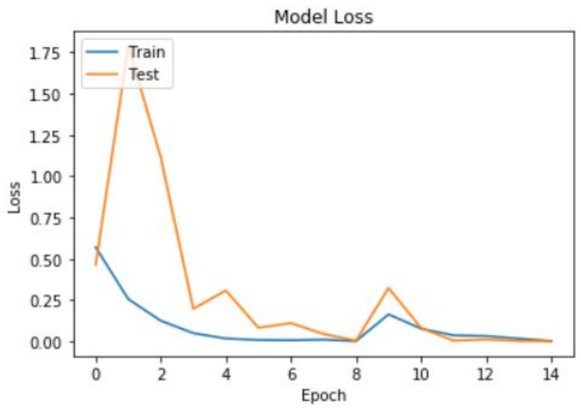
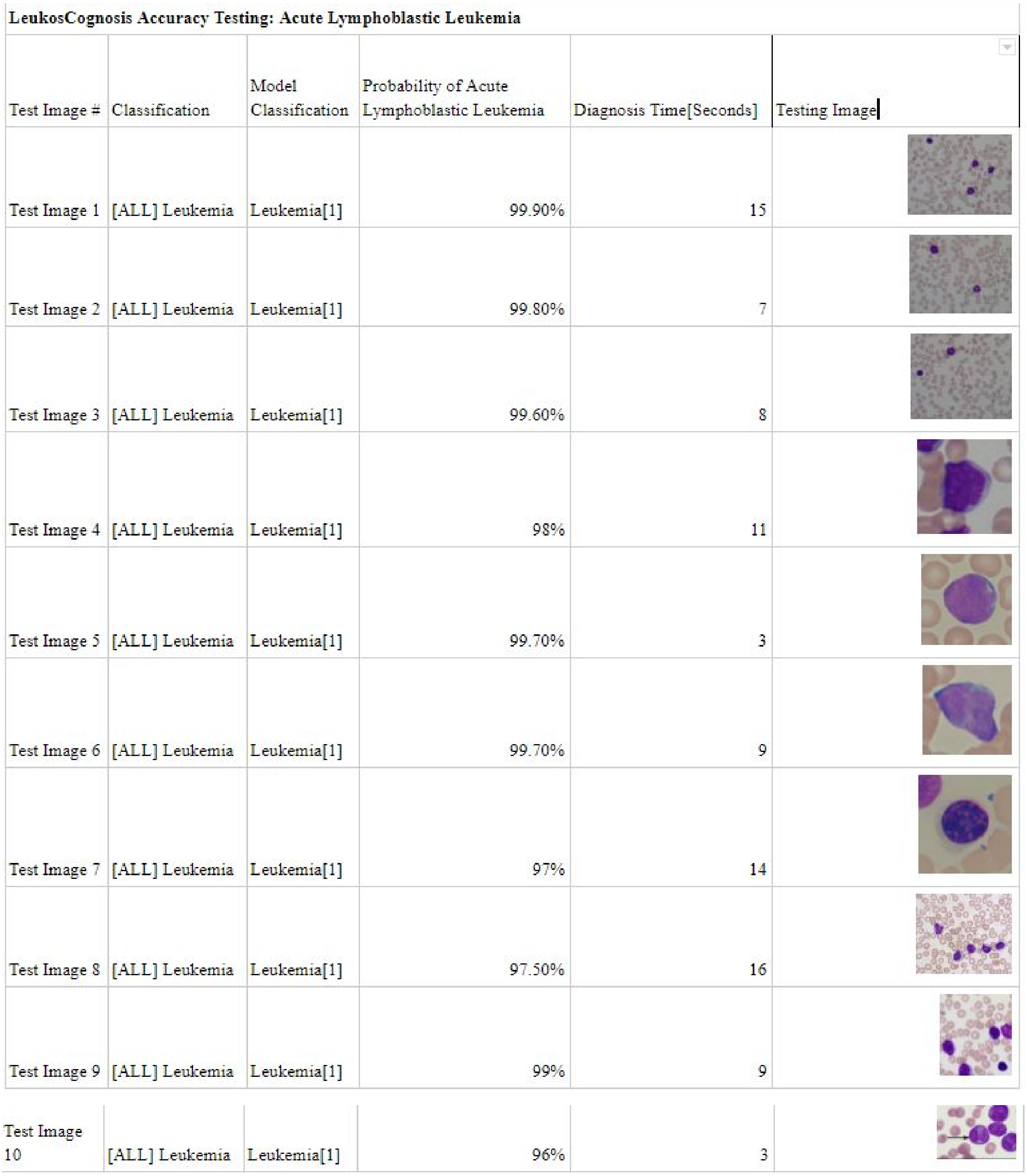


Figure 26-Depicts the Model Loss and Validation of the Final LeukosCognosis Machine Learning Algorithm throughout the epochs.

Source: 2020. *LeukosCognosis: Graph.* [image] The graph in this image was generated and created by the author [Sirihaasa Nallamothu].



LeukosCognosis Accuracy Testing: Acute Lymphoblastic Leukemia

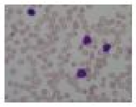
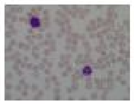
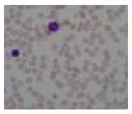
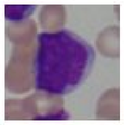
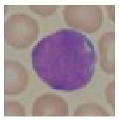
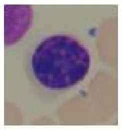
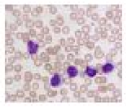
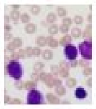
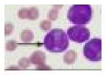
Test Image #	Classification	Model Classification	Probability of Acute Lymphoblastic Leukemia	Diagnosis Time[Seconds]	Testing Image
Test Image 1	[ALL] Leukemia	Leukemia[1]	99.90%	15	
Test Image 2	[ALL] Leukemia	Leukemia[1]	99.80%	7	
Test Image 3	[ALL] Leukemia	Leukemia[1]	99.60%	8	
Test Image 4	[ALL] Leukemia	Leukemia[1]	98%	11	
Test Image 5	[ALL] Leukemia	Leukemia[1]	99.70%	3	
Test Image 6	[ALL] Leukemia	Leukemia[1]	99.70%	9	
Test Image 7	[ALL] Leukemia	Leukemia[1]	97%	14	
Test Image 8	[ALL] Leukemia	Leukemia[1]	97.50%	16	
Test Image 9	[ALL] Leukemia	Leukemia[1]	99%	9	
Test Image 10	[ALL] Leukemia	Leukemia[1]	96%	3	

Figure 27-Depicts the Accuracy testing of the Acute Lymphoblastic Leukemia images.
Source: 2020. *LeukosCognosis:Table.* [image] The table in this image was generated and created by the author [Sirihaasa Nallamothu].

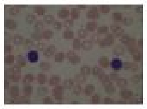
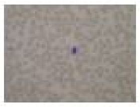
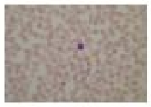
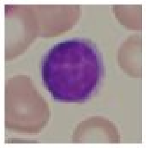
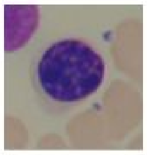
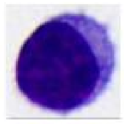
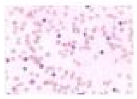
LeukosCognosis Accuracy Testing: Healthy					
Test Image #	Classification	Model Classification	Probability of Acute Lymphoblastic Leukemia	Diagnosis Time[Seconds]	Testing Images
Test Image 1	Healthy	Healthy[0]	1%	10	
Test Image 2	Healthy	Healthy[0]	0.60%	5	
Test Image 3	Healthy	Healthy[0]	0.10%	6	
Test Image 4	Healthy	Healthy[0]	0.14%	11	
Test Image 5	Healthy	Healthy[0]	0.03%	3	
Test Image 6	Healthy	Healthy[0]	0.20%	6	
Test Image 7	Healthy	Healthy[0]	0.01%	3	
Test Image 8	Healthy	Healthy[0]	0.10%	12	
Test Image 9	Healthy	Healthy[0]	2%	4	

Figure 28-Depicts the Accuracy testing of Healthy Blood cell images, through the Luekoscognosis model, in correlation to the amount of time, in seconds, it takes to diagnose the blood cell smear.

Source: 2020. *LeukosCognosis:Table.* [image] The table in this image was generated and created by the author [Sirihaasa Nallamothu].

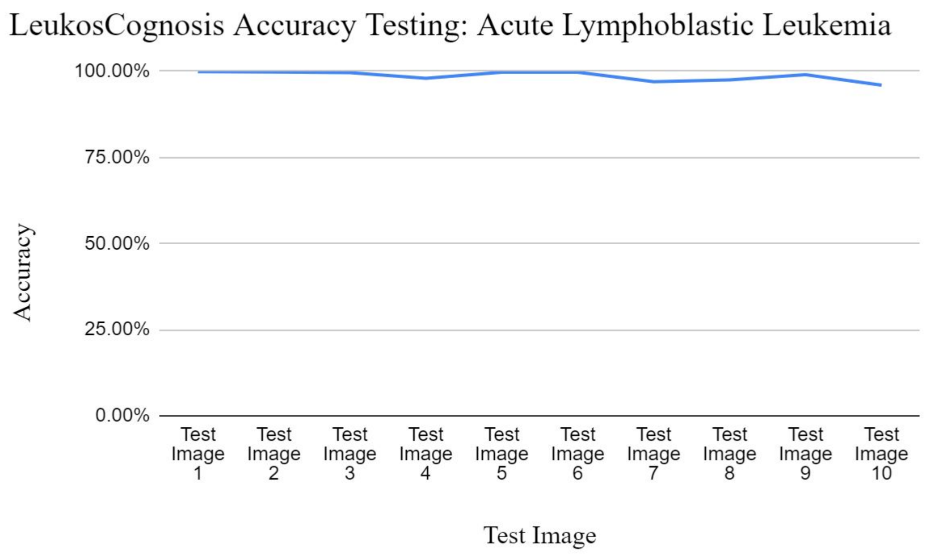


Figure 29-Depicts the correlation between the Acute Lymphoblastic Leukemia test images and model predicted accuracy.
Source: 2020. *LeukosCognosis: Graph.* [image] The graph in this image was generated and created by the author [Sirihaasa Nallamothu].

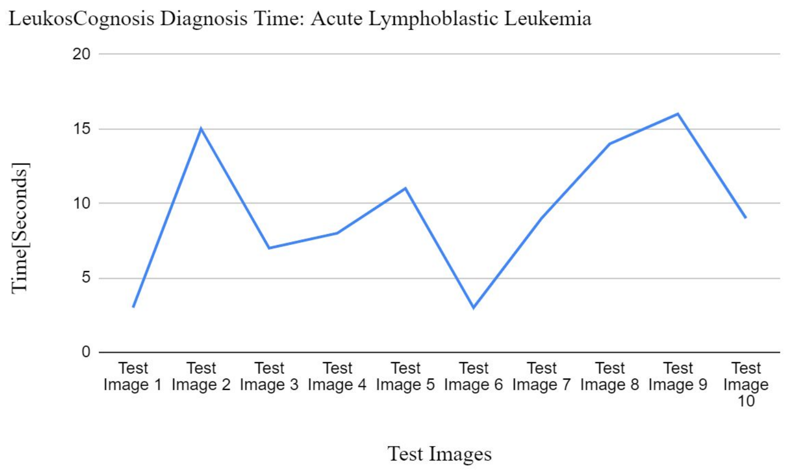


Figure 30-Depicts the correlation between the Acute Lymphoblastic Leukemia test images and diagnosis time.
Source: 2020. *LeukosCognosis: Graph.* [image] The graph in this image was generated and created by the author [Sirihaasa Nallamothu].

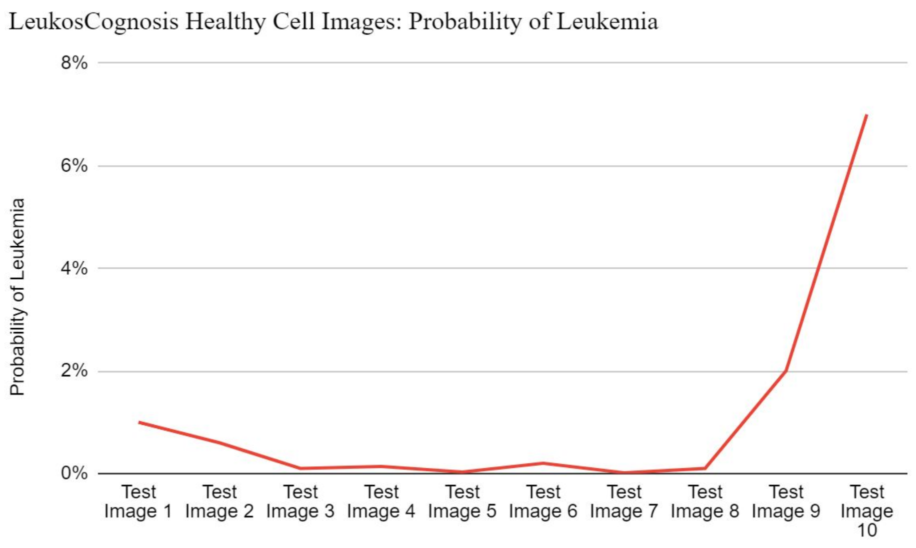


Figure 31-Depicts the correlation between Healthy test images and accuracy.
Source: 2020. *LeukosCognosis: Graph.* [image] The graph in this image was generated and created by the author [Sirihaasa Nallamothu].

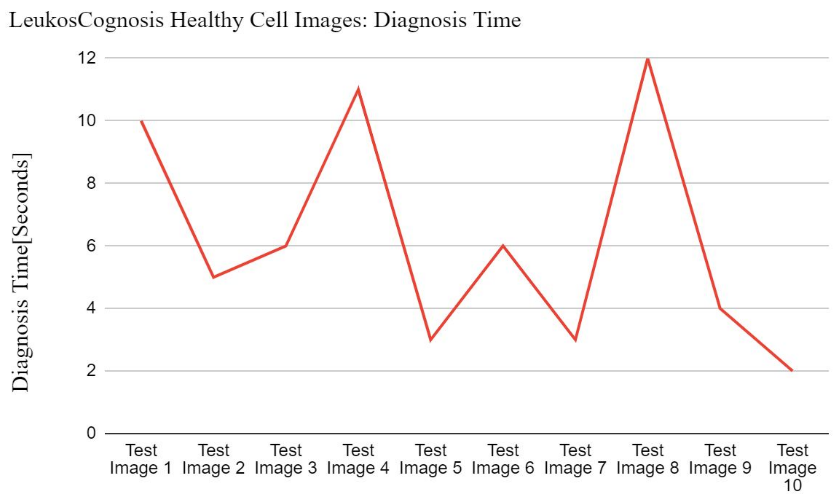


Figure 32-Depicts the correlation between Healthy test images and diagnosis time.
Source: 2020. *LeukosCognosis: Graph.* [image] The graph in this image was generated and created by the author [Sirihaasa Nallamothu].

Acknowledgments

The LeukosCognosis Algorithm would not have been possible without the help of my Father, Sateesh Nallamothu. This was my first time venturing into the world of machine learning and data science, and my dad gave me the advice and help I needed throughout building teh convolutional neural network. He helped me work out errors and gave me necessary encouragement. I would also like to thank my family, Swathi Kakarala, Sateesh Nallamothu and Shreya Nallamothu, for giving me the time and resources I needed to code my model and write this research paper. Lastly, I would like to thank the team at Youth STEM 2030 for providing me with this amazing opportunity. I truly appreciate the efforts of this organization.

References

[1] Lls.org. 2020. Facts And Statistics | Leukemia And Lymphoma Society. [online] Available at: <https://www.lls.org/facts-and-statistics/facts-and-statistics-overview/facts-and-statistics> [Accessed 6 May 2020].
[2]Cdc.gov. 2019. Questions And Answers About Leukemia. [online] Available at: <https://www.cdc.gov/nceh/radiation/phase2/mleukemi.pdf> [Accessed 11 June 2020].
[3]Publishing, H., 2014. Leukemia - Harvard Health. [online] Harvard Health. Available at: <https://www.health.harvard.edu/cancer/leukemia> [Accessed 18 May 2020].
[4]Hematology.org. 2020. Leukemia. [online] Available at: <https://www.hematology.org/education/patients/blood-cancers/leukemia> [Accessed 13 May 2020].
[5] ALL-IDB [Internet]. ALL-IDB Acute Lymphoblastic Leukemia Image Database for Image Processing. Fabio Scotti; 2006 [cited 2020Apr17]. Available from: https://homes.di.unimi.it/scotti/all/#
[6]Saha, S., 2018. A Comprehensive Guide To Convolutional Neural Networks -- — The ELI5 Way. [online] Medium. Available at: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53> [Accessed 19 May 2020].
[7]Malik, M., 2018. Machine Learning Of The Human Brain. [online] Medium. Available at: <https://towardsdatascience.com/machine-learning-of-human-brain-739ab0419612> [Accessed 10 May 2020].
[8]Lls.org. n.d. Leukemia & Lymphoma Society | Donate Today!. [online] Available at: <https://www.lls.org/> [Accessed 12 May 2020].
[9]GeeksforGeeks. 2019. How To Write A Pseudo Code? - Geeksforgeeks. [online] Available at: <https://www.geeksforgeeks.org/how-to-write-a-pseudo-code/> [Accessed 9 June 2020].
[10]Wu, J., 2019. AI, Machine Learning, Deep Learning Explained Simply. [online] Medium. Available at: <https://towardsdatascience.com/ai-machine-learning-deep-learning-explained-simply-7b553da5b960> [Accessed 10 May 2020].
[11]Nicholson, C., 2019. A Beginner's Guide To Neural Networks And Deep Learning. [online] Pathmind. Available at: <https://pathmind.com/wiki/neural-network> [Accessed 17 May 2020].

Images

[1]2019. *Prototyping Design Process-Software Development.*[image] Available at: <https://www.plutora.com/blog/software-development-life-cycle-making-sense-of-the-different-methodologies>[Accessed 8 May 2020] Note: Author made edits to the image for clarity
[2]Generated by Google Image Search Engine. Available at: <Leukemia White Blood Cells Microscopic Images> [Accessed 6 May 2020]

[3]2019. *Convolutional Neural Network-Layman's Terms*. [image] Available at: <https://towardsdatascience.com/convolutional-neural-network-a-step-by-step-guide-a8b4c88d6943 > [Accessed 4 May 2020].

[4] 2014.*Structure of Convolutional Neural Network-Neurons*. [image] Available at: <https://www.researchgate.net/figure/The-structure-of-a-convolutional-neural-network-adopted-in-this-paper-The-circles_fig2_286240187>[Acessed 10 May 2020]

[5]2019. NumPy Arrays flatten. [image]. Available at: <https://www.sharpsightlabs.com/blog/numpy-flatten>[Acessed 9 June 2020]