*Project Name*: Real-Time Analysis of Microscopic Video Data

*Report 4*

*Project Objectives*

Our goal is to stream in yeast and blood (white blood cell) microscopy and classify those cells based on features collected about each cell. While our main objective is solely classification, other long-term objectives include tagging the video, developing a lineage model of the cell (such as bacteria), and determining and tagging significant events, such as cells splitting or interacting or dying. This could also provide real time feedback about the window quality of the microscope on the sample, allowing the biologist to move the window to get a better view(such as making sure a minimum of multiple phenotypes are within the region depending on the experiment.)

*Significance*

Cell microscopy and observation is largely done by specialized human workers as humans are the gold standard of image detection and recognition as well as knowing what important biological features to assess. This is costly as the person’s time could be used on other facets of research and human labor itself is costly.

Cell experiments are also performed in stages. That is, one single gene or biological factor is looked for and analyzed through microscopy (generally analyzing the images after the experiment and after applying different filters to the data to make things stand out), after which another experiment is performed. If biological features that are sought after could be analyzed and detected in real time, the experiments could become more modular, allowing a researcher to apply a chain of experiments together, saving time and money.

This is very complex feature analysis that would have extremely accurate though. So the bare bones fundamental structure, such as performing lineage tracking and extracting features, would have to be able to function well.

*Features: Use Case/Scenario*

Our primary use case is to be able to detect the cells in each frame in a video. That is, given a frame and a possible number of cells to choose from, determine what type of cell is it. Counting the total number of cells is also biologically useful and an extension of this case.

Our current implementation is to use SIFT features to determine important key points for a yeast cell and identify those points within the video for each frame. Unfortunately, this does not work very well without full control over our microscopy videos, especially when it comes to determining the SIFT points of a yeast cell within the video. This could be due to how few SIFT points we found within the image, but changing the amount of points we generate on the image and video did not improved our methods of detection.

Training on more images of yeast and aggregating those features through image analysis aggregation schemes like Fischer Vector, VLAD, or Super Vector would yield much better results than just training and trying to detect one image. Using these schemes in MATLAB is an easy task but fairly uncertain one for Java. Such tasks would be for future, long-term implementation of this project, but not this one in its current state.

A second important use case, to pursue in the future, is tagging what state the cell is in. We wish to be able to tag a yeast cell that is budding, that is there is a smaller yeast cell growing conjoined to it, or E.Coli splitting, where E.Coli grows long and then tapers in the middle before it splits entirely into two different sister E.Coli.

Initial testing trials indicated that we had a better job of identifying yeast cells that are budding, at least for the image we trained on, however even though SIFT is rotation invariant, it wasn’t able to pick up budding cells that are in a different orientation. Again, training on more than one image and aggregating our features through an image analysis aggregation schema could improve our detection powers for classifying whether an image contains budding or splitting cells and annotate them correctly. This also proved to be difficult if we ever compared SIFT points of budding yeast cells to SIFT points of just regular yeast cells, as they are very similar.

These two use cases would allow us to map states of one image to another to determine lineage and map a lineage tree the cells, a third and currently not implemented use case.

*Approach*

* Data Sources
  + Published cellular research with microscopy movies (frames taken every 2-5 minutes). Youtube also offers video data of microscopy as well, generally with frames taken within a shorter window, but for lesser duration. The research data also tries to analyze more complicated mechanisms while the youtube data displays features such as budding and reproduction.
  + Research papers that we have looked at may be found in our literature review, and an example of youtube data would be the following:

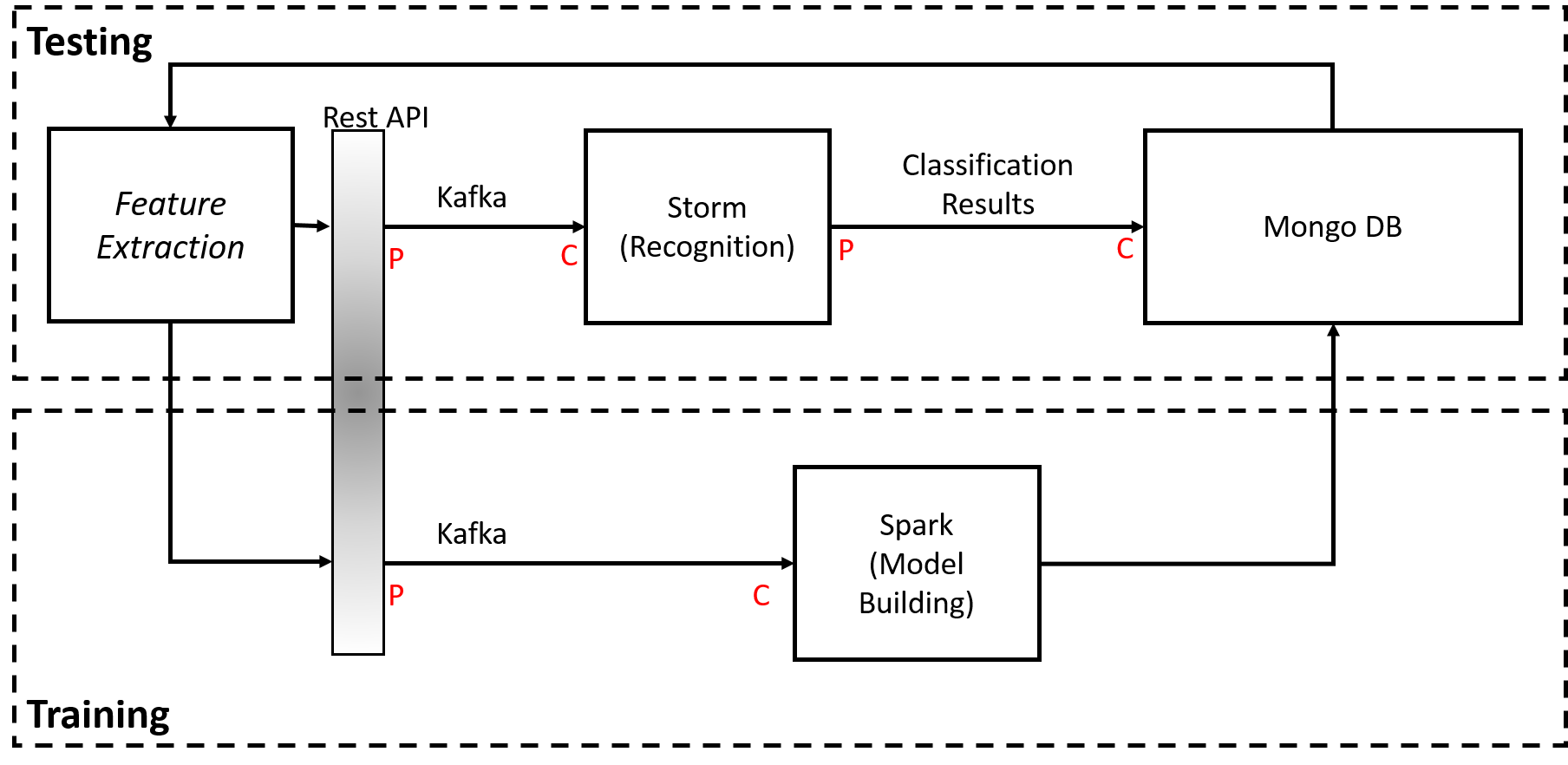
<https://www.youtube.com/watch?v=iOvrq6ssy2Y>

<https://www.youtube.com/watch?v=BTHYaf-EuYs>

* Analytic Tools
  + Spark- Apache Spark allows us to perform data parallelism through the mapreduce paradigm, allowing us to train and learn from our data in order to build a model to classify what parts of our images are cells and what state they exist in.
  + OpenImageJ- OpenImageJ is a library that allows the user to extract key points and frames through SIFT point detection and tagging.
  + Spark MLLib- the Spark MLLib is a rich resource of machine learning, allowing us easy access to decision trees, random forests, feature analysis, and neural network schemes.
  + Storm- Storm allows us to perform task parallelism so that when we stream in videos as frames, we are able to perform many tasks on those frames in order to determine where cells are and what state they are in.
  + Kafka- Kafka is a publisher subscriber system that allows a robust high volume queuing of data from which Spark and Storm can act as both consumers and producers of different topics in order to communicate between themselves and the client.
* Analytical Tasks
  + Detect salient features of a cell
  + Classify a cell based on SIFT features (yeast or white blood cell)
  + Train on multiple images and aggregate SIFT features from them
  + Extract many features from cells that are important for SIFT feature data used in classification.
* Expected Inputs/Outputs
  + The expected input would be a video of cell microscopy, where on the client side the main frames are extracted.
  + From these frames, the SIFT features are extracted and published to Kafka, which is consumed by Storm.
  + The output of storm is in MongoDB, which shows the frame number and whether the SIFT point in that frame was yeast or a white blood cell.
* Algorithms
  + SIFT aggregation schemes: Training on one image and using it as representative of an entire class doesn’t work and really doesn’t work with our data. As such, we need to aggregate many sift points from different cells of the same type and group them to classify what is a yeast cell or what is a white blood cell. These methods include Bag of Words at its most primitive, evolving into VLAD, Fischer Vector, and Super Vector as more complicated schemes. These were not implemented in this project, but will be used for any future investigation into this problem.
  + Random Forest/Decision Tree algorithms
  + Object detection, aggregation, and segregation using image analysis

*Related Work*

* Open Source Projects
  + MicrobeJ for ImageJ (I don’t think MicrobeJ is opens source though, but ImageJ is)
* Literature Reviews (some are from proposal)
  + <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4820279/> (ecoli videos)
  + <http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0032621> (yeast)
  + <http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1003396#s5> (yeast)
  + <http://www.ncbi.nlm.nih.gov/pubmed/27572972>
  + Chen, Zhou &amp; Wong. “Automated segmentation, classification, and tracking of cancer cell nuclei in time-lapse microscopy.” IEEE Transactions on Biomedical Engineering, vol. 53, no. 4, April 2006. <http://ieeexplore.ieee.org/document/1608529/>
  + Dewan, Ahmad &amp; Swamy. “A Method for Automatic Segmentation of Nuclei in Phase-Contrast Images Based on Intensity, Convexity and Texture.” IEEE Transactions on Biomedical Circuits and Systems, vol. 8, no. 5, October 2014. <http://ieeexplore.ieee.org/document/6762958/>
  + Goutam &amp; Sailaja. “Classification of acute myelogenous leukemia in blood microscopic images using supervised classifier.” 2015 IEEE International Conference on Engineering and Technology (ICETECH), March 2015. <http://ieeexplore.ieee.org/document/7275021/>
  + Tran, Pham &amp; Zhou. “Cell phase identification using fuzzy Gaussian mixture models.” Proceedings of 2005 International Symposium on Intelligent Signal Processing and Communication Systems, 2005. <http://ieeexplore.ieee.org/document/1595447/>
* Application Specification
  + OpenImaj/SIFT
* System Specification/Software Architecture
  + Spark - ML to assign probabilities of specific features in a frame
  + Storm - efficiently handle feature extraction from cell video data
  + Kafka - transferring video data between systems
  + Potentially utilizing MongoDB to store data that can be used to train and build our model for cell detection
  + AWS cloud - cloud computing platform we will utilize to implement the above technologies in real time
* Design of Big Data Analytics Server
  + Parallelism/Distribution: Task/Data
    - Data parallelism of splitting image into frame then applying ML algorithms to train for object classification.
    - Task parallelism within a frame to determine edges/sift points/biological tagging through color analysis or shape analysis.
  + General Workflow (as taken from lecture):



* + - Training: Data is extracted and sent to Spark (via Rest API). Spark builds a model and sends the corresponding If-Else decision tree to MongoDB
    - Testing: Data is extracted and sent to Storm (via Rest API). Storm classifies cells within video, draws a box around classified cells and labels based on the chosen classification. The annotated video is stored in MongoDB, and can be retrieved by the client.
  + Storm Topology:

storm.png

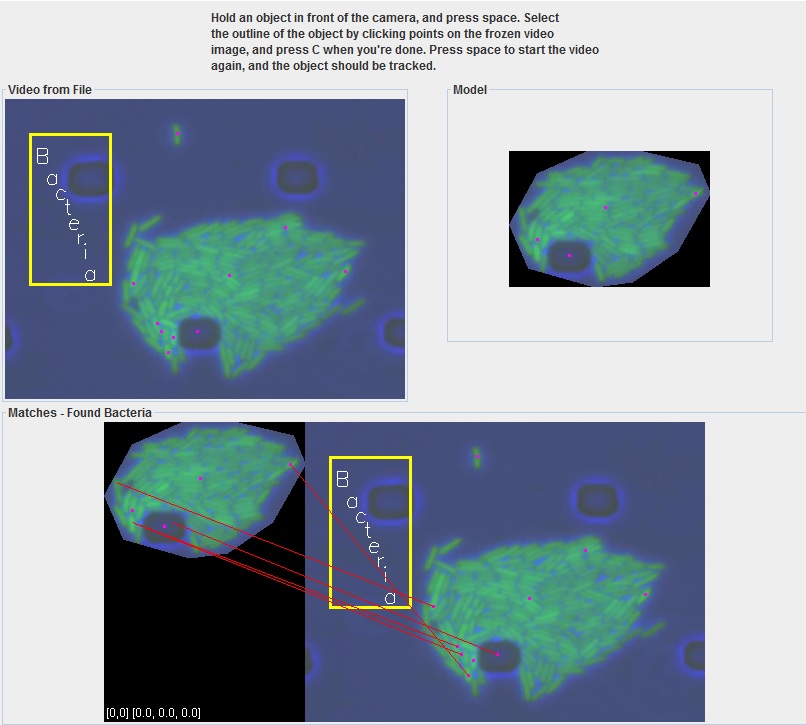
* + - KafkaSpout: Consumes streaming data from client
    - WBCBolt: Bolt that determines (based on generated If-Else decision tree) if incoming feature data is a White Blood cell. If so, it sends out the frame number and True (False if not) to MongoDB
    - YeastCellBolt: Bolt that determines (based on generated If-Else decision tree) if incoming feature data is a Yeast cell. If so, it sends out the frame number of True (False if not) to MongoDB
* Design of Mobile Client (smartphone/web): N/A
* FFmpeg (https://github.com/adaptlearning/adapt\_authoring/wiki/Installing-FFmpeg) to convert videos to grayscale
* Existing Applications/Services Used: Time-lapse microscopy videos (see references for other videos) (<http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4820279/>)

*Documentation of Milestones*

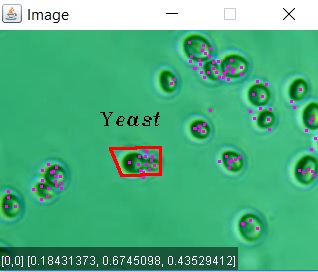
* Source Code GitHub Link:https://github.com/DRinKC/RT-BigData-Project\_Team1
* Video Demo GitHub Link: https://github.com/DRinKC/RT-BigData-Project\_Team1/tree/master/Demo
* Project Milestones:
  + Metadata output from Main Frame Detection
    - https://github.com/DRinKC/RT-BigData-Project\_Team1/wiki/Lab-3

consoleOutput.jpg

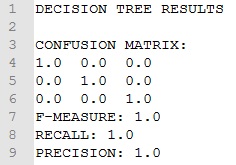
* + Object detection and tracking with annotation
    - https://github.com/DRinKC/RT-BigData-Project\_Team1/wiki/Lab-4



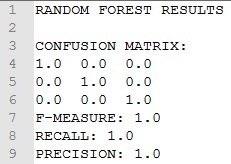
* + Object detection and annotation
    - https://github.com/DRinKC/RT-BigData-Project\_Team1/wiki/Lab-5



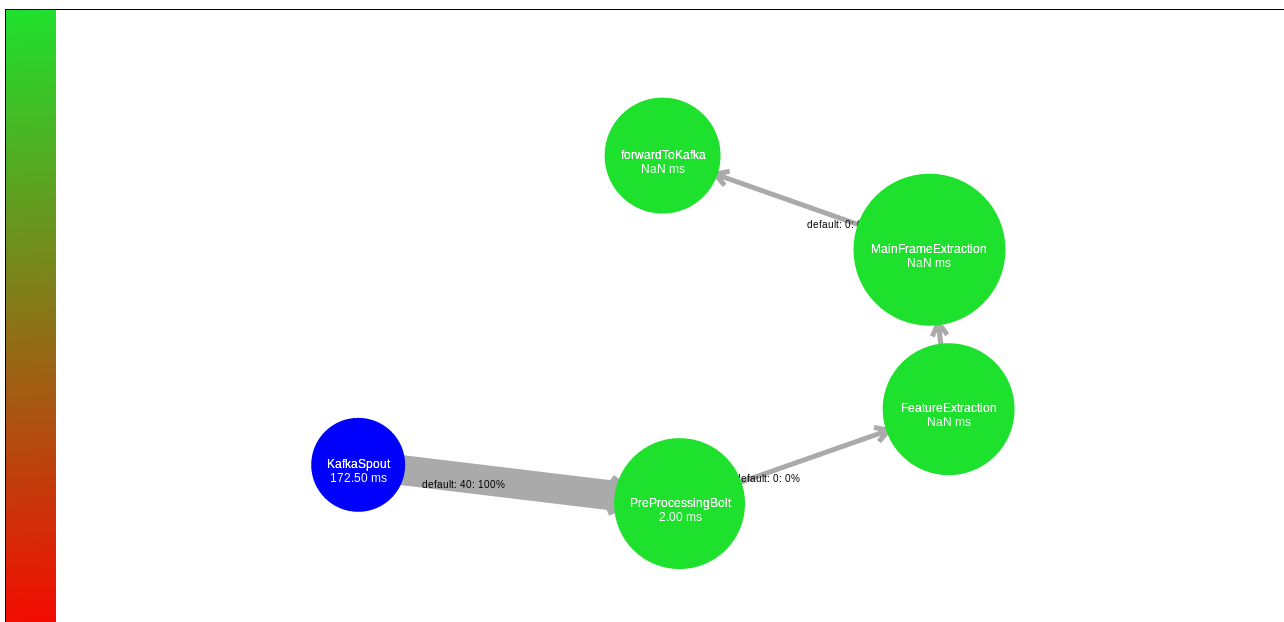
* + Decision Tree output:
    - https://github.com/DRinKC/RT-BigData-Project\_Team1/wiki/Lab-5



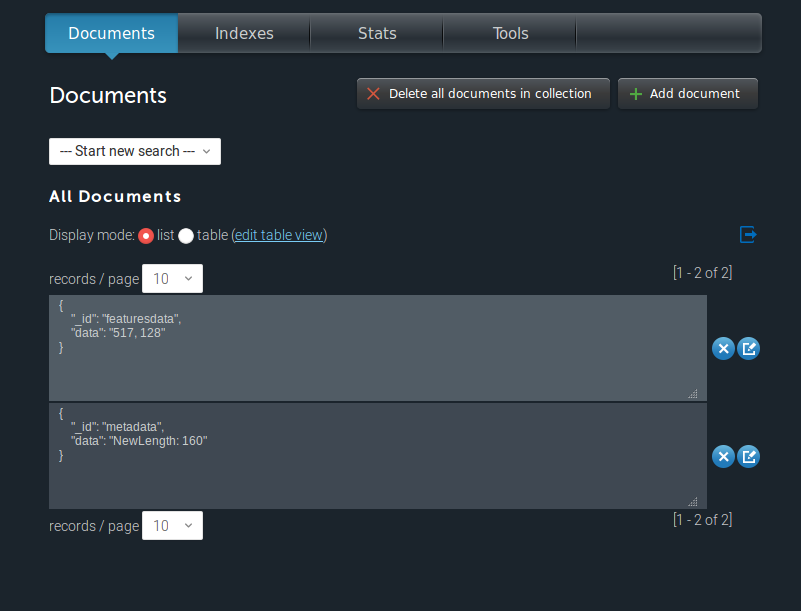
* + Random Forest output:
    - https://github.com/DRinKC/RT-BigData-Project\_Team1/wiki/Lab-5



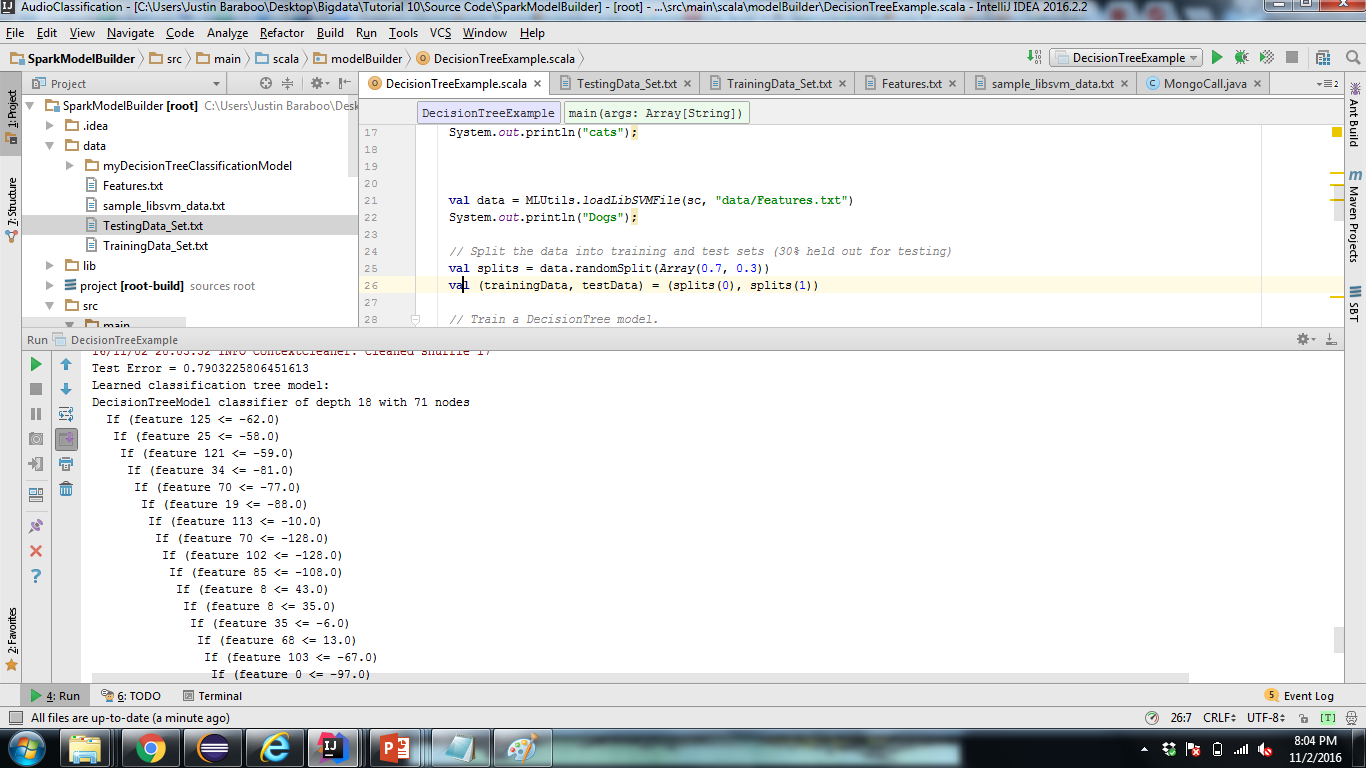
* + Extracting key frames, metadata and features, and sending data to Storm via Kafka Producer (no screenshot available)
    - https://github.com/DRinKC/RT-BigData-Project\_Team1/wiki/Lab-6
  + Creating Storm topology (previous version) to analyze extracted information
    - https://github.com/DRinKC/RT-BigData-Project\_Team1/wiki/Lab-7,8



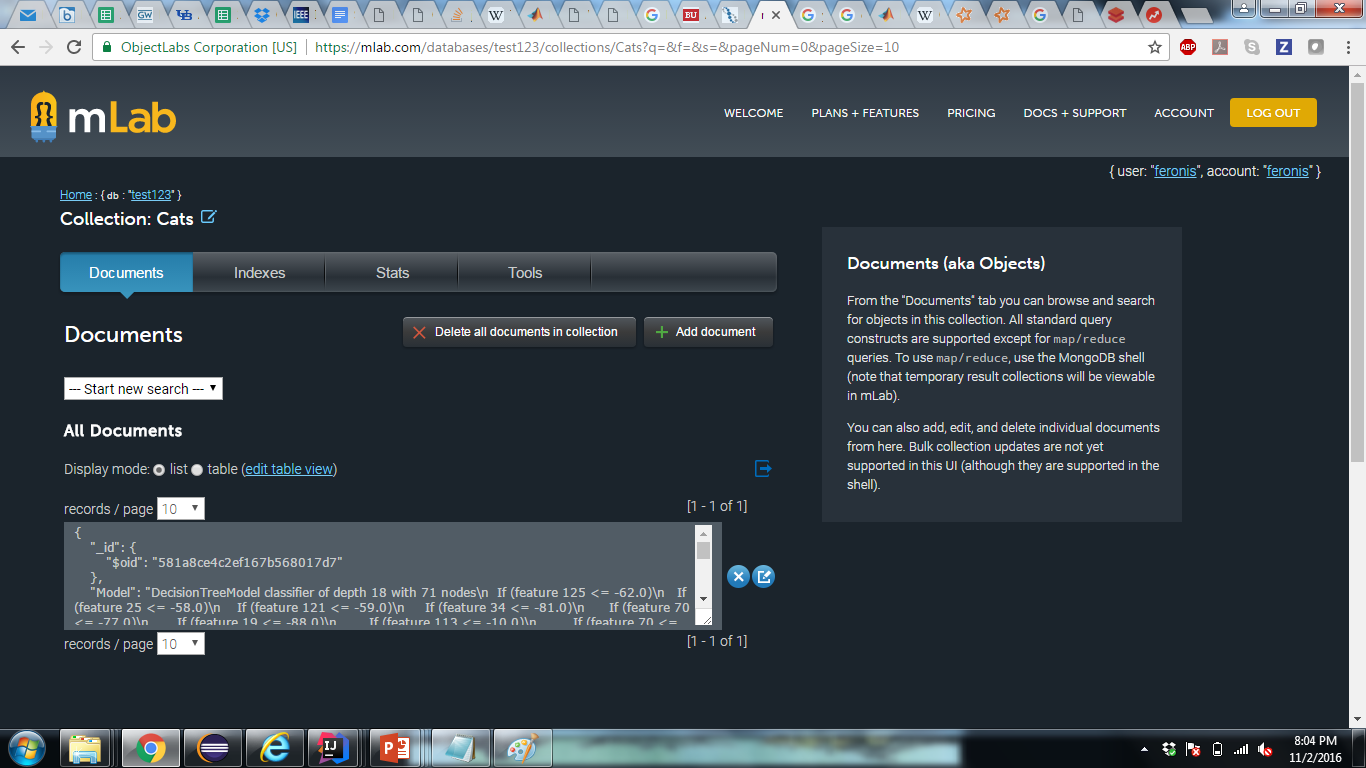
* + Sending results to MongoDB
    - https://github.com/DRinKC/RT-BigData-Project\_Team1/wiki/Lab-7,8



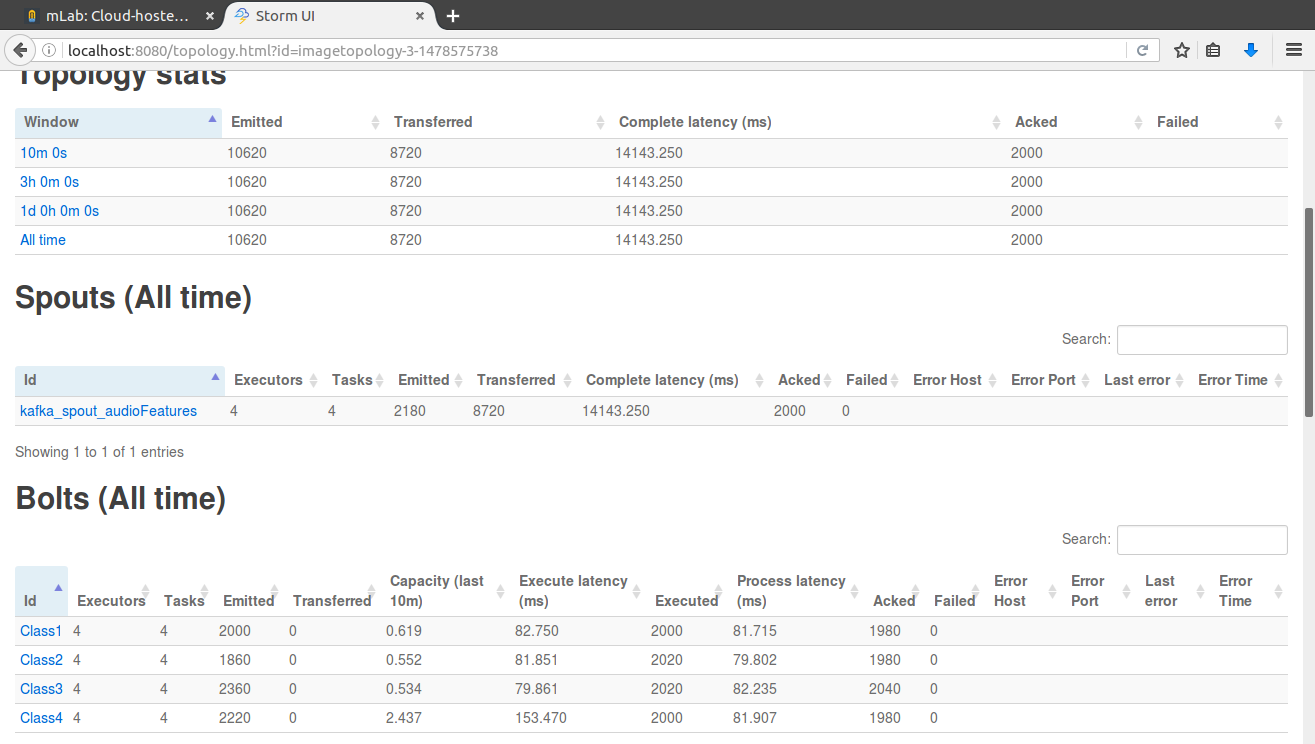
* + Creating If-Else decision tree with SparkModelBuilder
    - https://github.com/DRinKC/RT-BigData-Project\_Team1/wiki/Lab-9



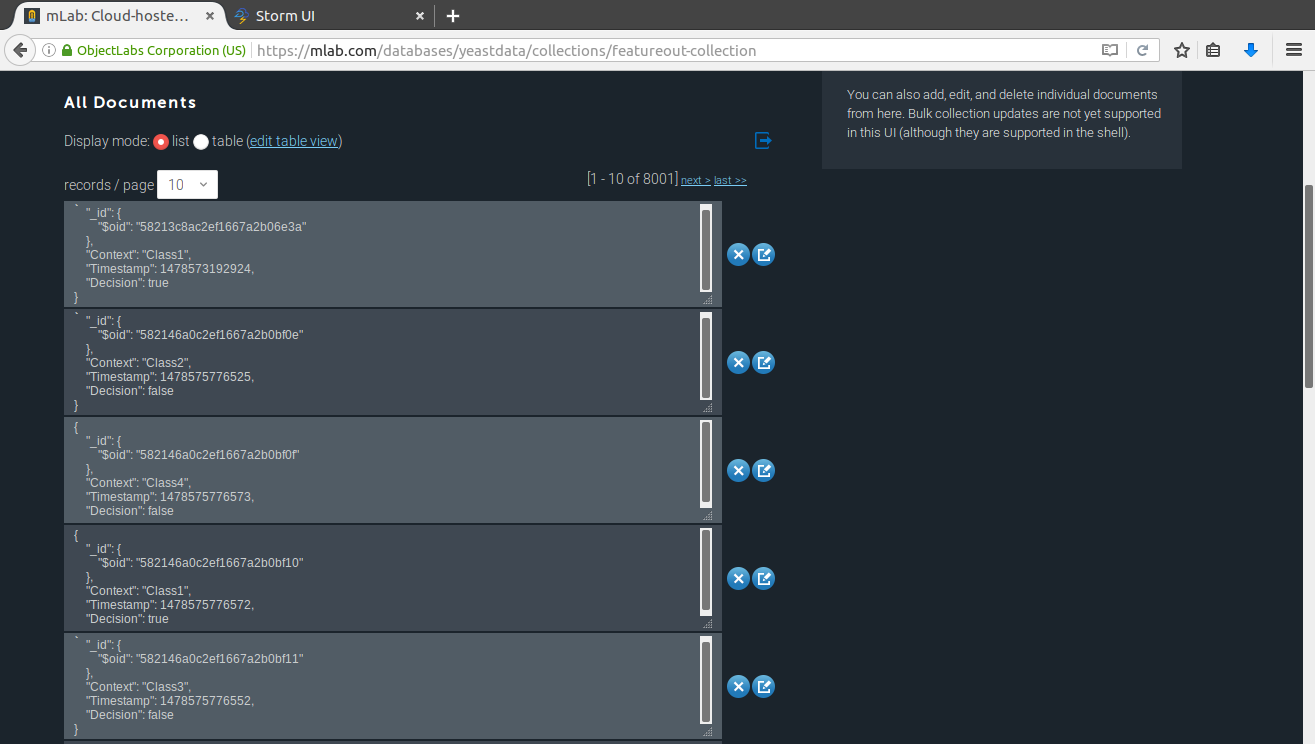
* + Sending If-Else decision tree to MongoDB
    - https://github.com/DRinKC/RT-BigData-Project\_Team1/wiki/Lab-9



* + Generating Storm topology based on decision tree in MongoDB
    - https://github.com/DRinKC/RT-BigData-Project\_Team1/wiki/Lab-10



* + Sending data to be classified through Storm whose output is sent to MongoDB
    - https://github.com/DRinKC/RT-BigData-Project\_Team1/wiki/Lab-10



*Project Management: Implementation status report*

* Work completed:
  + Description:
    - Generated final classification model (Yeast vs White Blood Cell) based on training data
    - Generated final topology
    - Obtained more videos for training and testing
    - Ran videos through entire workflow
    - Created video demo
    - Processed and analyzed results (training and testing)
  + Responsibility:
    - Bill: Obtained videos for training and testing
    - David: Obtained videos for training and testing; processed and analyzed testing data
    - Brendan: Generated final topology; ran videos through entire workflow; created video demo
    - Justin: Generated final classification model based on training data; processed and analyzed training data
  + Contributions (members/percentage): roughly equal (25%)
* Issues/Concerns:
  + Some videos have too many details/background data attached to them that it makes main frame detection nearly impossible for some videos
  + Videos are all irregular - some have a lot of background noise which can increase probability of irrelevant SIFT point detection

**Project Video**

GitHub Link: https://github.com/DRinKC/RT-BigData-Project\_Team1/tree/master/Demo

**Related Work**

One of the very first milestones of Big Data Analytics was the creation of Hadoop and the MapReduce framework. With the MapReduce framework, the data is split up into partitions and undergoes two phases, Map and Reduce. These two phases are the building blocks to many tasks and help to achieve a fairly efficient and fast performance that is unparalleled near the time of its creation. But it has some serious pitfalls. The growing complexity of Big Data problems of today is too great for the MapReduce with severe overhead and CPU issues. More importantly, MapReduce frameworks cannot sufficiently match the needs of data that needs be to processed and analyzed in real-time. The lambda architecture is a real-time architecture that can be applied to a multitude of frameworks, with its ability to handle both streaming, real-time data and also batch data (similar to MapReduce) to create a singular, adaptable model. In this review, we take a look at a variety of frameworks to illustrate the scalability, adaptability and efficiency of the lambda architecture.

Apache Spark is a framework that is similar to MapReduce while still supporting a lambda architecture. By itself, it is a micro-batch processing framework that can outperform MapReduce’s efficiency. It has the Resilient Distributed Dataset (RDD), that is immutable and helps to allow fast access to data. It is also fault-tolerant and helps to reduce overhead while running MapReduce operations. It works well in a lambda architecture with Storm or Heron. Storm and Heron are both primarily stream processing engines that take data continuously (in the form of tuples) and distribute them to bolts to carry out tasks. Heron is a much more complex, but also more efficient and fault-tolerant than Storm. These streaming engines have the capability of incorporating their input and output with Spark to create simultaneously a stream and batch framework, not unlike the lambda architecture. It provides the capability to create a model that can both learn from stored datasets as well as new data coming in, especially with Kafka. Kafka is a publisher-subscriber system to which data can be streamed and from which data can be consumed. This allows for a user to publish data, and have their Storm-Heron-Spark lambda architecture consume that data in real-time. It can also be the broker between Spark and Storm, and serve as a reliable, scalable, efficient piece of the lambda architecture, without which the key elements of streaming and batch processing would not be able to efficiently combine to tackle the scale of Big Data.

The trend to support both large scale batch and stream processing evolves with other frameworks, altering the architecture to handle a variety of data and environments. For instance, Pipeline61 framework is designed to be compatible with a multitude of execution environments, while S4 uses Processing Elements to handle data in any order or type while achieving high accuracy and speed. Both of these frameworks open potential to create complex lambda architectures to not just combine streaming and batch processing, but also a variety of data, offering perhaps a realistic approach to Big Data solutions. Summingbird is another framework that could work well with others to create a scalable lambda architecture. Its most powerful feature is to bring together online and batch MapReduce operations, which is well tailored to a Big Data lambda architecture.

Many may think that the downfall of having any streaming portion of an architecture is that it relies on already trained data or a model that has been built. Even if the model adapts to new data, it implies that the new data may not be as powerful as the data collected from a batch process. While this is not necessarily true, there are two frameworks that combine trained and untrained data to create a mostly unsupervised learning approach to address this very issue. Long Short Term Memory and Recurrent Neural Networks and Deep Learning of Traffic Flows are studies that have one major concept in common: autoencoders. Autoencoders play an important role in both reconstruction and prediction of data, and are especially important in neural network flexibility. As a result, these networks can learn an adaptable model with primarily streaming data as its basis for its model. While also having a lambda architecture, it can prioritize at once what it has learned and what is actual useful based on previous and incoming data; it’s usefulness has noted with predicting video sequences and traffic flows.

Other applications take on much larger scale issues that affect us on a daily scale. Cloud computing incorporates a lambda-like architecture in creating a synergistic robot to compute both pre-mapped models and generate new movement patterns based online data. WiseReplica works well with video-on-demand solutions that require adaptability based on user input and patterns to reduce replication and increase video bitrate. Such efficiency is owed in large part to its lambda architecture, being able to adapt to new information while retaining an overall model or goal. Scientists were also able to combine a variety of sources of data as well as a complex mix of Machine Learning algorithms to be able to predict human emergency mobility following natural disasters (such as an earthquake). While this model does not entirely implement a streaming engine of any sort, the fact that it can process both visual, audio, numerical, etc., data in large quantities shows that it has the adaptability and scalability of a lambda-like architecture, for these different data types cannot generate a similar model with a MapReduce-like framework. Indeed, the power of flexibility, scalability, and efficiency of these Big Data frameworks can only be achieved with a lambda architecture.

**Proposed Solution**

We wish to achieve object classification, such as cell differentiation, and state recognition, such as when a cell is splitting or moving. We propose a lambda architecture to solve this, composing of Spark for offline learning, Storm for online classification, and Kafka for communication from each component, the client, and a mongodatabase. Cell differentiation can be solved through aggregation and training of SIFT points as a keypoint feature. State recognition can be learned from the shape data of cells in their different points of splitting and reproduction. Both of these can be learned from different cell images and become training data for a machine learning algorithm. For a naive model, we are only looking at raw sift points data and creating a decision tree from that. These decision paths are parsed for each cell and becoming a classifier bolt in our storm topologies allowing classification when the client sends in frame data and features. This data is then output to MongoDB for the user to determine how each SIFT point was classified in each frame.

**Implementation**

We implemented a lambda architecture to solve this problem. We use Kafka for the user publish their feature data extracted from key main frames of the video, and have Storm consume the stream of data. Spark consumes the user’s data in batches to create a one-time produced model, stored in MongoDB. This model is the basis of the classifier bolts in Storm. While the model does not adapt to new data, it does combine both online and batch processing elements of a lambda architecture. The Storm topology consists of a Kafka spout to consume streaming data from Kafka and classifier bolts. Each bolt uses the Spark Model Builder decision tree to classify if streaming SIFT feature data belong to either a Yeast or White Blood Cell. It will emit “true” or “false” as well as the frame number.

The imagined workflow of this solution is to have a user stream extracted feature data from the main frames of a microscopy video. This data is published to Kafka for both Storm and Spark to consume. Spark consumes this data to build a classification decision tree model and stores this decision tree output in MongoDB. This decision tree is used to generate Storm bolts and create a topology used to classify incoming feature data. Upon consumption of this data by the Kafka Spout in Storm, the data is then classified as either Yeast or White Blood Cell in the form of a “true” or “false” message for each SIFT point in each frame. This data is stored in MongoDB, where the user can access the results.

To train the model, SIFT features from images were used. For testing the model, videos were in mp4 format, shortened to around 30 seconds or less and converted to grayscale to reduce variability amongst the test data.

**Results and Evaluation**

*Size and type of video data sent in*

Sec points frames runtime

white (3 vids) 87 12300 2514 3:45

yeast (2 vid) 19 16810 1268 3:40

In our evaluation, we had 3 white blood cell videos and 2 yeast cell videos. Above is the summary of such data.

*Confusion Matrix*

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Predicted | |
|  |  | White | NotWhite |
| Actual | White | 655 | 15638 |
|  | NotWhite | 269 | 12657 |

Accuracy: 13312/29219 = 45.6%

Error: 1 - Accuracy = 54.4%

Recall: 655/16293 = 4.02%

Precision: 655/924 =70.9%

This classifier performed about as same chance, with an almost equal accuracy and error rate. It seems overall this classifier misclassifies a lot of actual white blood cells as not white blood cells (recall = 4.02%). When it does predict white blood cells, however, it usually is fairly accurate (precision = 70.9%). Given by the low recall, it seems that the classifier needs to increase its sensitivity to improve performance.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Predicted | |
|  |  | Yeast | NotYeast |
| Actual | Yeast | 110 | 16536 |
|  | NotYeast | 97 | 12376 |

Accuracy: 12486/29119 = 42.9%

Error: 1 - Accuracy = 57.1%

Recall = 110/16646 = 0.66%

Precision = 110/207 = 53.1%

The classifier performed also just a little bit worse than chance (accuracy = 42.9%). The yeast classifier tends to misclassify actual yeast cells are not yeast cells (low recall < 1%), but when it does classify a SIFT point as a yeast cell, it performs a little bit better than chance (precision = 53.1%). Overall, this classifier had a low sensitivity, which would need to improve to improve the overall performance.

Taken together, both classifiers performed similar and tend to be less sensitive than desired. Given this, it would seem that there is more of a systemic error that is involved with the SIFT feature extraction for the model building and/or for testing.

**References**

Cloud Automation

Bekris K, Shome R, Krontiris A, Dobson A. "Cloud Automation: Precomputing Roadmaps for Flexible Manipulation", IEEE Robotics & Automation Magazine Vol 22: Issue 2, 2015.

Datanami

Alex Woodle. “From Elephants to Spiders: The History of Hadoop”.

https://www.datanami.com/2015/04/15/from-spiders-to-elephants-the-history-of-hadoop/

Deep-Intelligence Framework

Zhang W, Xu L, Li Z, Lu Q. "A Deep-Intelligence Framework for Online Video Processing". IEEE Computer Society, pp 44-51, Mar 2016.

Deep Learning Traffic Flow Prediction

Lv, Yisheng, et al. "Traffic flow prediction with big data: a deep learning approach." Intelligent Transportation Systems, IEEE Transactions on 16.2 (2015): 865-873.

Google

Dean J, Ghemawat S. "MapReduce: simplified data processing on large clusters". Communications of the ACM Magazine - 50th anniversary issue: 1958-2008, Vol 51: Issue 1, Jan 2008.

Heron

Kulkarni, Sanjeev, et al. "Twitter heron: Stream processing at scale." Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. ACM, 2015.

IBM

"What is big data?" IBM - Bringing Big Data to the Enterprise. URL: https://www-01.ibm.com/software/data/bigdata/what-is-big-data.html

Kafka

Kreps J, Narkhede N, Rao J. "Kafka: a Distributed Messaging System for Log Processing". NetDB Workshop, 2011.

Lambda

Bijnens N. "A real-time architecture using Hadoop and Storm". DataCrunchers, Dec 2013. URL: http://lambda-architecture.net/architecture/2013-12-11-a-real-time-architecture-using-hadoop-and-storm-devoxx

Pipelines

Wu, Dongyao, et al. "Building Pipelines for Heterogeneous Execution Environments for Big Data Processing." IEEE Software 33.2 (2016): 60-67.

Unsupervised Learning/LSTM

Srivastava N, Mansimov E, Salakhutdinov R. "Unsupervised Learning of Video Representations using LSTMs", Proceedings of the 32nd International Conference on Machine Learning (ICML-15), pp 843-852, 2015.

S4

Neumeyer, Leonardo, et al. "S4: Distributed stream computing platform." 2010 IEEE International Conference on Data Mining Workshops. IEEE, 2010.

Simulator of Human Emergency Mobility

Song, Xuan, et al. "A Simulator of Human Emergency Mobility following Disasters: Knowledge Transfer from Big Disaster Data." Twenty-Ninth AAAI Conference on Artificial Intelligence. 2015.

Storm

Toshniwa A, Taneja S, Shukla A, Ramasamy K, Patel JM, Kulkarni S, Jackson J, Gade K, Fu M, Donham J, Bhagat N, Mittal S, Ryaboy D. "Storm @Twitter". Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, pp 147-156, 2014.

Summingbird

Boykin, Oscar, et al. "Summingbird: A framework for integrating batch and online mapreduce computations." Proceedings of the VLDB Endowment 7.13 (2014): 1441-1451.

Twitch

“Welcome Home the 2015 Retrospective” URL:

URL: https://www.twitch.tv/year/2015

WiseReplica

Silvestre S, Buffoni D, Pires K, Monnet S, Sens P. "Boosting Streaming Video Delivery with WiseReplica". Transactions on Large-Scale Data-and Knowledge-Centered Systems, 34-58.

YouTube

“Statistics - YouTube”. YouTube URL: https://www.youtube.com/yt/press/en-GB/statistics.html

**Video/Image References:**

*Training for yeast/budding yeast:* (https://www.dropbox.com/sh/gtslv0u1l6hgvlq/AAAa96AsSGDkiPHCzvuil7pBa?dl=0)

*Testing videos yeast*

https://www.youtube.com/watch?v=PTGYLYru2Sw

https://www.youtube.com/watch?v=5vWDqZu-iHk

https://www.youtube.com/watch?v=YzRpAfYArQg

https://www.youtube.com/watch?v=CbuPYxtM0Y8

https://www.youtube.com/watch?v=cfTomGVcZYc

https://www.youtube.com/watch?v=sFZkDM2Dv7U

https://www.youtube.com/watch?v=9Mg-WqE-TwY

https://www.youtube.com/watch?v=YMKbRPIAFdw

https://www.youtube.com/watch?v=0wtA3p6kqtA

https://www.youtube.com/watch?v=N8bp1FC6myc

*Testing videos budding yeast:*

https://www.youtube.com/watch?v=5ruXdbUl\_TE

https://www.youtube.com/watch?v=BTHYaf-EuYs

https://www.youtube.com/watch?v=YsIxw5LWXvI

https://www.youtube.com/watch?v=iyWtp\_L0Kzc

https://www.youtube.com/watch?v=GFEgB\_ytDZY

https://www.youtube.com/watch?v=Bg\_6hSVycxE

https://www.youtube.com/watch?v=89x05ivxbpM

https://www.youtube.com/watch?v=Eo3v0XTQBTs

https://www.youtube.com/watch?v=zc2MyYn7D7s

https://www.youtube.com/watch?v=cgss48c76fk

*Training images from youtube videos for white blood cell (test portions do not include training images):*

https://www.youtube.com/watch?v=Va1jaBGwoT8

https://www.youtube.com/watch?v=zgJNhhtIAvg

https://www.youtube.com/watch?v=JnlULOjUhSQ

https://www.youtube.com/watch?v=sYCUWqc\_x3U

https://www.youtube.com/watch?v=PsrnMeu6sIY

https://www.youtube.com/watch?v=qvGVoxdy-yM

https://www.youtube.com/watch?v=2TKTSHrw5QI&t=2s

https://www.youtube.com/watch?v=KAnFGn-ecNU

https://www.youtube.com/watch?v=rj-MS4Oxdu8

https://www.youtube.com/watch?v=f0UGnI5N8lg