Classification of Histopathological Breast Cancer Images using CNN on Apache Spark

Krishnamohan Pathicherikollamparambil
Ananth Sankarasubramanian
Kausal Mahadas

Electrical and Computer Engineering
Rutgers University – New Brunswick

***Abstract*—Breast cancer is one of the main causes of death in women worldwide. The diagnosis of biopsy tissue with hematoxylin and eosin stained images as either benign or malignant is a very complex procedure and specialists often go wrong in the final diagnosis. Computer aided approaches can greatly simply this problem and aid the specialists in their diagnosis. Conventional machine learning approaches are not suited for this scenario as their feature learning abilities are limited. Here we propose to use Convolutional Neural Networks (CNN) on top of Apache Spark to classify the biopsy tissue images as benign and malignant. The project explores two different approaches for implementing CNN; one using the SparkDL package provided by Databricks and the other using a custom made CNN model built using the Keras application. The SparkDL also provides API for transfer learning using which the output of the deep learning module is passed on to a logistic regression model and a SVM model. The architecture of the custom made CNN is specifically trained on the cancer tissue biopsy images and extracts nuclei level features from the images. By running the whole implementation on top of Spark we can distribute the computations and greatly reduce the time required to predict the diagnosis for several hundreds of images.**

I. INTRODUCTION

Cancer is a major global health concern. It affects people of all ethnicities and age groups. Based on data from GLOBOCAN 2012 which was produced by the International Agency for Research on Cancer (IARC), in 2012 alone there were more than 14.1 million new instances of cancer and 8.2 million deaths due to cancer worldwide [1]. These numbers are only projected to rise in the coming years; by the end of 2030 the number of occurrences and the number of deaths due to cancer are expected to increase by 50% and 60% respectively [4]. In the U.S. in particular, the American Cancer Society estimates 1,688,780 new instances of cancer and 600,920 deaths due to cancer for 2017 . Lung cancer is the leading cause of cancer related deaths in males regardless of the developmental state of their country and has surpassed breast cancer as the cause of death in females from developed countries. However, breast cancer is the leading cause of cancer death in females from less developed countries.Hence, the detection and diagnosis of cancer has been an integral part of medical research. This project focuses specifically on breast cancer and the use of computer aided mechanisms to classify images as benign or malignant.

In the initial stages of diagnosis, mammography, ultrasound imaging, magnetic resonance imaging (MRI) and other similar diagnostic tools are used to detect breast cancer. These tests are always followed by a breast tissue biopsy, which remains the most prominent form of diagnosis of breast cancer (National Breast Cancer Foundation, 2005). In a biopsy, a sample of the patient's tissue or fluid is taken from the affected area and then an H&E (hematoxylin and eosin) stain is applied to better visualize the tissue cells. Once this is completed, a pathologist will histologically analyze the samples under a microscope. These tests are fairly straightforward and allow the pathologist to visually differentiate between benign tissue and cancerous tissue.

When it comes to procuring samples, there are three types of biopsies doctors usually use: fine-needle aspiration, core-needle biopsy, and surgical biopsy. A fine-needle aspiration is used in cases where the doctor suspects the region is question is filled with fluid. A needle is inserted and the fluid is drained, if the lump re-emerges an additional biopsy will likely be performed to obtain sample cells. In the case of a core-needle biopsy, a larger needle is used to remove a small sample size of suspicious tissue. As this is more intensive than a fine-needle aspiration, the breast will be anesthetized. Likewise, with a surgical biopsy, the patient will be under local anesthesia, but the surgeon will remove a much larger part or all of the abnormal tissue.

The issue with this method of analysis is that there is a lot of dependency on manual verification from the pathologist. This not only makes the process heavily reliant on the experiences of the pathologist but decreases productivity as more effort is placed on analyzing these samples for signs of abnormality. For these reasons researchers have turned to computer-aided diagnostic (CAD) systems for potential solutions to address these problems. CAD systems automate the image analysis process by filtering based on classifications of benign or malignant. This not only improves the efficiency of diagnosing cases, but also increases the accuracy.

II. PREVIOUS WORK

Current approaches to image classification rely on small datasets. As there is only a handful of researchers working on this issue, data and knowledge is fairly limited. A study of breast cancer based on fine needle biopsy microscopic images tested various clustering algorithms on the segmentation of nuclei. Although, they reported an accuracy in the range of 90%, their dataset only consisted of 500 image. Other studies also focus on similar concepts. For instance, one study focused on analyzing cytological images for nuclei segmentation. This study was one of the few to touch base on machine learning models such as support vector machines and convolutional neural networks; however, the same drawback of a limited dataset was observed in this study as well. Due to the highly specialized nature of this subject and the lack of documentation, many research studies cite the same sources for reference material. As the use and reliance of computer-aided tools are increasing, the application of convolutional neural networks (CNNs) and support vector machines (SVMs) has also been on the rise.

Convolutional neural networks are used in machine learning for image recognition and classification. They are based on how information is processed in biological neural networks. CNNs use a multilayer perceptron (MLP) which has layers of nodes that reduce pre-processing. As such, CNNs are able to efficiently extract features and how found much support in the field of medical image analysis. However, similar issues were found in studies that utilized CNNs; most of the studies used datasets that contained images of small pixel ratios and/or the datasets they used were not publically available.

Ultimately, we were able to find two research papers that explored concepts similar to our own and we were able to draw inspiration from their models when constructing the model used in this project. One of the papers focused on histopathological image classification of breast cancer. A publically available dataset, called BreaKHis, of 7907 breast cancer images with image dimensions of 700x460 pixels was used in the study. A classification system based on four machine learning models was developed to differentiate between benign and malignant instances. The classifiers are as follows: a 1-nearest neighbor (1-NN), quadratic linear analysis (QDA), SVMs, and random forests (RF) of decision trees.

The second paper focused on classification of breast cancer histology images using convolutional neural networks. Unlike other studies, the researchers classified the dataset into four categories: normal tissue, benign lesion, in situ carcinoma, and invasive carcinoma (Araujo et al., 2017). This was done in an effort to integrate histological data from multiple layers. Additionally, the studied utilized 2 datasets of 249 and 20 images all composed of 2040x1536 high resolution images. To visually analyze the images and better observe the tissue structures, the researchers created 12 overlapping patches of 512x512 pixels.

By studying the publically accessible models provided by the two papers discussed above, we were able to draw inspiration for the two approaches used in our project. Finally, the Databricks platform was also used to integrate the multiple libraries utilized in our model with Amazon Web Services.

III. OUR CONTRIBUTION

While previous attempts on the BreaKHis dataset involved adopting Machine Learning Algorithms such as SVM and KNN to classify the images as Benign or Malignant. Our contributions involve adopting the more recent Deep Learning techniques such as Convolutional Neural Networks – using both the inbuilt Deep Learning Pipeline as well as building our very own model using Keras. Furthermore, the task of image classification was optimized by conducting it on a distributed frame work such as Spark.

IV. EXPERIMENTAL SETUP AND PLATFORM

A. Setup:

	Description
Spark Platform	Databricks
Spot Instances	Amazon EC2 Configuration: Driver node : p2.xlarge (beta) • 61Gb, 1 GPU, 2 DBU • Min – Max Workers: 2-8
Cloud Storage	Amazon S3
Deep learning Library	Keras with TensorFlow Backend
Pre-trained model used	Inception V3
Language	Pyspark

Table 1

a. Data bricks:

Data Bricks is a unified Analytics Platform that is built on top of Apache Spark native to cloud and is about 10 - 40X faster than other Spark distributions.

The Project involved analyzing more than 2000 images, integration with libraries such as Keras and Tensor Flow and had to be computationally powerful - doing all this in a distributed manner. Data Bricks was the ideal platform as it met all the demands explained above and eliminated the mundane tasks and helped us focus on the main aspects of the project.

b. EC2 Spot Instances:

The computational requirements of the program (i.e.) processing 512 X 512 pixel images meant that we had to rent out GPU capable systems. The Amazon p2.xlarge EC2 systems were used with 1 Driver and 8 Workers - each having 61 Gb memory and an onboard GPU each.

c. Cloud Storage:

Amazon S3 is an object storage for storing and retrieving large amount of data while being highly durable, secure and available. In addition S3 can support sophisticated big data analytics on data as required by our program. The powerful integration between Databricks and Amazon Web Service made S3 a natural choice for storing our Training, Testing and Validation Images.

d. Keras:

Keras is a high level deep learning library that calls into lower level deep learning libraries such as TensorFlow and Theano, making it easy and fast to implement Convolutional and Recurrent Neural Networks using Python. In addition, programs using Keras can be run on both CPUs and GPUs.

While Approach 1 using the Deep Learning Featurizer : calls the Keras Library internally, the second approach involving the construction of the various layers of the convolutional neural network for image classification explicitly uses the Keras Library for declaring and initializing the various layers and hyper - parameters for the model.

B. Data Bricks Deep Learning Pipeline:

Leveraging the power and efficiency of deep learning is a challenge not only because of the fact that it is still and active area of research but also the complexity involved in distributing the work across nodes. Deep Learning Pipelines allow for implementing deep learning algorithms at scale through high – level Python based APIs. It helps achieve the scale using Spark's powerful distributed engine. Keras and Tensorflow support techniques such as Transfer Learning and Hyper Parameter Tuning in addition to providing tools to turn models into SQL functions accessible by a wider audience. Some of the features are as follows:

a. Working with images:

One of the most important requirements of distributed deep learning is the ability to handle Millions of Images. The Deep Learning Pipelines provided by spark allow for manipulation at scale by supporting image loading and decoding, paving way for manipulation at scale.
The output Data Frame would contain the file path to each image and an image struct column containing the decoded image.

b. Transfer Learning:

Training a Convolutional Neural Network is a pretty arduous task because it is tough to find a dataset of sufficient size. To eliminate this problem, it is common to train a CNN on a very large dataset and use the features extracted to make predictions on the new image.

CNN as a fixed feature extractor: The last fully connected layer obtained by running the CNN on a large dataset is removed and used as a feature extractor on the new data set. The output is generally a vector of codes. A linear classifier such as logistic regression or an SVM is trained on the new dataset.

Fine Tuning the CNN: Layers of pre-trained CNN become progressively specific to a dataset. It is therefore wise to fine-tune a few layers of the CNN (higher layers).

With respect to Spark, the Deep Learning Pipelines provides functions and Pre-Trained Models to perform transfer learning. In this paper, the InceptionV3 pre-trained model (trained on the ImageNet dataset and capable of predicting on 1000 classes) is used to make predictions in parallel. In general, Keras models and Tensor Flow graphs too can be added to the spark prediction pipeline.

c. Distributed Hyperparameter Tuning:

Better results can be obtained from Deep Learning by tuning the various parameters such as the batch size, learning rate and epochs. This is called Hyper Parameter tuning and the required infrastructure is already built into spark.

d. Deploying models in Data Frames and SQL

Deep Learning Models built can be accessed using a function in SQL and this makes it easy to be accessed right about by anyone from a software engineer to a Data Scientist and make predictions accordingly.

We adopted two approaches to classifying Breast Cancer Images as to whether they are Benign or Malignant. The first involves using the SparkDL Libraries and the second one involves using Keras to build our own convolutional neural network. A more detailed description of the two approach are as follows.

V. METHODOLOGY

A. Approach 1:Using SparkDL and MLlib

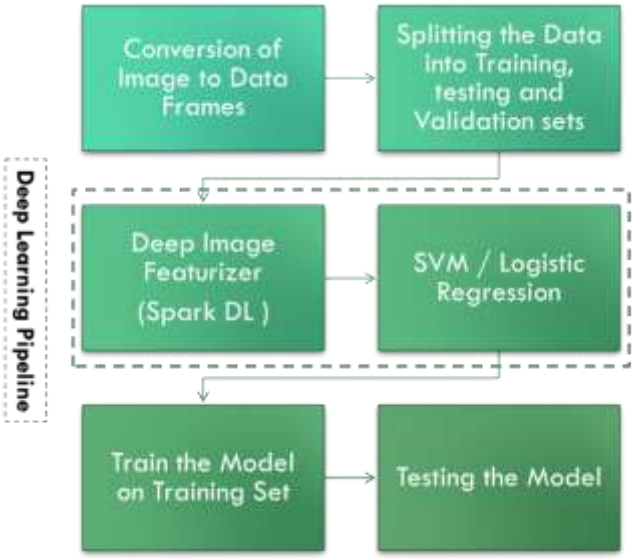

Figure 1

The function readImages in the sparkdl package is used to load the images stored on Amazon S3 into a Spark data frame. The images in our bucket were classified into benign and malignant. The resultant Data-frame contains a column file path, a column indicating the number of channels, pixels and the binary encoding and a column indicating the label. This is then split into training, testing and validation sets and fed to the Deep Learning Pipeline.

The Deep Learning Pipeline consists of two stages:

1. The Deep Image Featurizer: Transforms images to numeric features. Peels the last layer of the pre-trained neural network and uses the output of the previous layers as features for the classifier.

2. The SVM or Logistic Regression Classifier: The machine Learning algorithm that analyzes the independent features to classify the image according to one of the two classes.At this stage the parameters such as the number of iterations, the regularization parameter are set.

The classifier is then run on the training set to obtain a model. As the number of Iterations increase, the error decreases and accuracies improves. The resultant model is then applied on the testing set. The model accuracy is calculated using the Multi Classification Evaluator.

The Second segment of the pipeline is the Algorithm used for classification. The two algorithms used are Logistic Regression and Support Vector Classifier.

B. Logistic Regression:

This is the first algorithm used to classify the Images based on the features received from the Deep Image Featurizer. It is basically a classification algorithm used to predict binary outcomes such as Pass or Fail, Present or Absent etc.

While the input is continuous the output is required to be binary. This criterion is handled very well by the Sigmoid function that is defined as follows:

$$g(z) = \frac{1}{1+e^{-z}}$$

For any input, the output is defined between 0 and 1. In other words, it follows the properties of the probability distribution. If threshold is set as 0.5, when an outcome produces a probability greater than 0.5 it is set to 1 and if it produces a probability less than 0.5 it is set to 0.

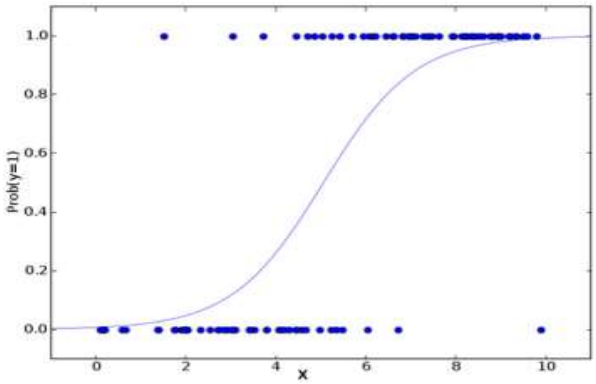

Figure 2

As shown in the figure above, for a certain feature X plotted on the x-axis the probability of the image being benign or malignant is plotted on the y-axis. A model is developed based on the training data points and considering the response variable to be categorical. Given a new image, depending on the probability obtained, if the probability is greater than 0.5 it is labelled to Malignant and if it is less than 0.5 it is labelled to be Benign.

The distributed version of Logistic regression can be found in the MLlib package where parameters such as the threshold, the number of iterations etc. can be controlled.

C. Support Vector Classifier:

The Support Vector Classifier is a method that uses a nonlinear mapping to transform the original training data into a higher dimension. Once it does so, it finds the linear optimal separating hyperplane.

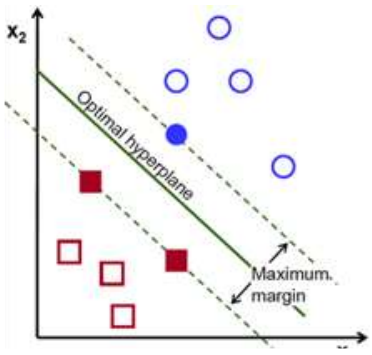

Figure 3

As can be seen in the plot above, the SVM algorithm is an iterative algorithm that looks at finding the hyperplane that

maximizes the margin between the training samples (i.e.) twice the largest minimum distance to the training samples.

This property of the SVM minimizes the risk of misclassifying examples and makes it a very good suit for classifying images. In our case the SVM works in a n-dimensional space where n depends on the number of features and finds the n-dimensional hyper plane that distinguishes between benign and malignant images.

D. *Hierarchy of DL transformers for Images*

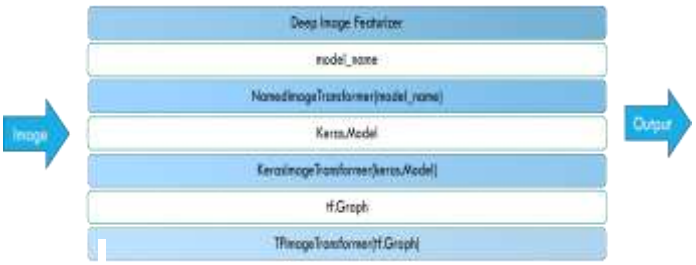

Figure 4

a. *DeepImageFeaturizer*

Programmers could use this classs to pass the popular name of a model to the next level of the DL hierarchy. The input image column should be in a 3-channel SpImage format and will be converted into an input dataframe. The input to the function is a dataframe of the following format:

filePath:	string
image:	struct
mode:	string
height:	integer
width:	integer
nChannels:	integer
data:	binary
label:	integer

The final output returned by the function is an MLlib vector so that DeepImageFeaturizer can be used in the MLlib Pipeline.

b. *NamedImageTransformer:*

It is one of the internal classes of the DL package called within the DeepImageFeaturizer class. The NamedImageTranformer class has functions to fetch the corresponding model ID using the popular name from the DeepImageFeaturizer class as the input. It also contains additional functions to resize the input images into the appropriate tensor format.

The NamedImageTransformer has an additional attribute to choose the type of output which can either be predictions or the features from the images. Also based on the model ID, the NamedImageTransformer calls the _buildTFGraphForName function to load the tensorflow graph corresponding to the model being called. Currently the API only supports pre-trained modules from the keras application module.

The resized image tensor along with the unique model ID and other parameters will be passed on to the TFImageTransformer class.

c. *KerasImageTransformer:*

The KerasImageTransformer class is functionally very similar to the NamedImageTransformer class but the main difference is that it allows any pre-trained Keras model to be passed to the TFImageTransformer class and not just the ones in the Keras application. User can implement and train his own model using Keras and save the weights file which can be later passed on to the KerasImageTransformer class. The input and the output to the class will be of the same format as mentioned in the NamedImageTransformer class.

d. *TFImageTransformer:*

This class forms the core of the DL transformer stack. It applies the tensorflow graph passed by the NamedImageTransformer to the resized image column in the dataframe. The output of the TFImageTransformer will be a MLlib vector.

It converts the ImageSchema data into a vector of type string and uses tensorframes to effectively distribute the graph to the workers. The loaded tensorflow graphs will then act on the portioned data to give the output predictions or features.

Restrictions with the current API (as mentioned in tf_image.py file within the SparkDL package) are as follows:

1. Does not use minibatches, and so directly affects the performance.
2. Only one output node can be specified
3. All images in the data frames are expected to be of the same datatype.

E. *Approach 2: Using Custom Built CNN Model*

The InceptionV3 model used in approach 1 and several other pre-trained models such as VGG16 available in the keras application are trained on the imagenet dataset which consists of regular images of objects and people. Considering the unique nature of the BreaKHis dataset we need to have a neural network specifically trained on such histopathological images.

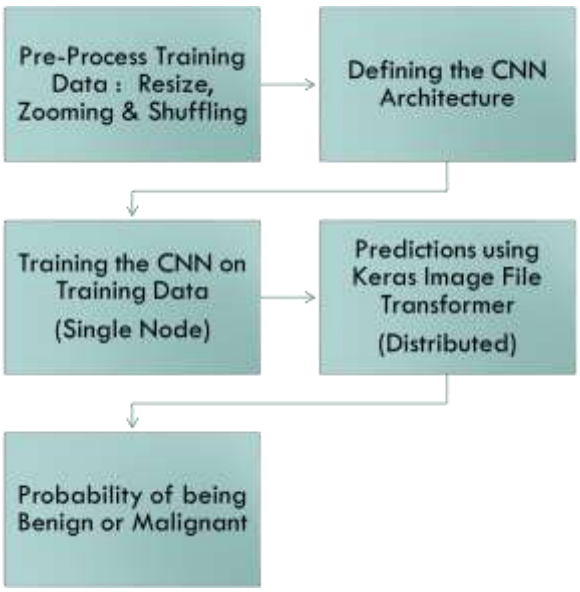

Figure 5

The convolutional neural networks are the best when it comes to learning features from images and Keras provides a very convenient API to construct the CNN model. The model which was used in the second approach was based on the architecture provided in one of the previous works

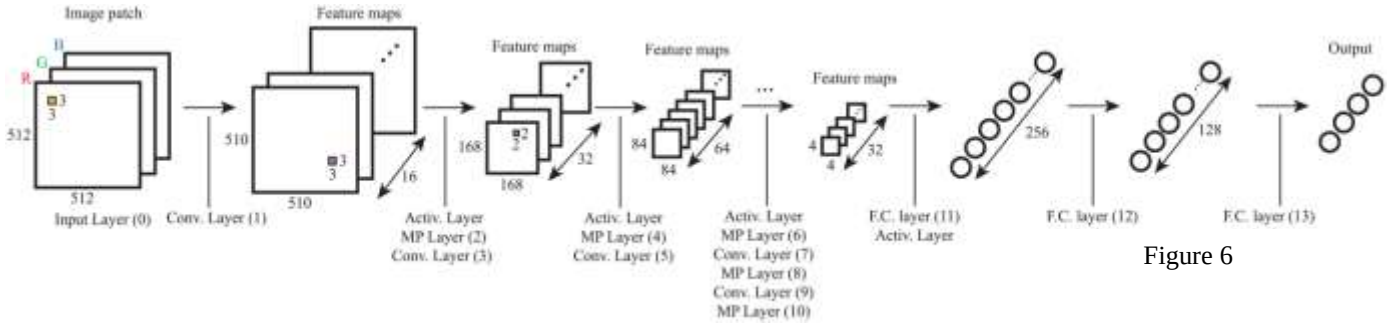

Figure 6

which used CNN to classify similar biopsy tissue images[2].

The structure of the CNN model used for the project is given in figure 6
The neural network architecture shown in figure has got several convolutional-pooling layers followed by three fully connected layers.

Input Layer: The input layer is designed for 3 channels (RGB) and 512 X 512 pixels (images will be resized to these dimensions)

Depth of the network: As evident from the dataset, it is essential that the network should be able to analyze several features at scale. It is important to analysis nuclei scale features to make accurate predictions and therefore the CNN has two convolutional layers with enough maps to study all such features.

Max-pooling: The max pooling layer helps in keeping the number of parameters in the network at a reasonable value by only choosing the most important elements within the pooling window.

Non-saturating nonlinearity: The relu layers within the convolutional layer and the fully connected layers help in introducing non-linearity in the model and hence reduce the chance of over fitting.

Output Layer: The output layer consists of two neurons corresponding to the two classes (benign and malignant) that are normalized with a softmax activation function.

Furthermore we use the Adam optimizer to improve the convergence of the model. The Adam optimizer uses some important tricks such as increasing momentum and adaptive learning rate. Momentum is achieved by adding some fraction of the previous update to the current update so that repeated update in a particular direction builds up momentum. In this case we build up momentum in the direction of the minimum and this helps in speeding up the convergence of the model. Also, Adam adaptively selects a separate learning rate for different parameters such that appropriate learning rates vary across parameters. Since the Adam algorithm has to compute different learning rates for different parameters, the processing time during training is slightly more when compared to using optimizers like SGD.
The training set consisted of 2872 images from the BreaKHis dataset, all of them from the 100x,200x magnification category . And the validation set consisted of 400 random images belonging to the same magnification

			Layer type	Maps & Neurons	Filter size	Effective receptive field	Effective receptive field (μm)
		0	Input	3M × 512 × 512N		1 × 1	0.4 × 0.4
	A	1	Convolutional	16M × 510 × 510N	3 × 3	3 × 3	1 × 1
		2	Max-pooling	16M × 170 × 170N	3 × 3	5 × 5	2 × 2
C	B	3	Convolutional	32M × 168 × 168N	3 × 3	11 × 11	4.6 × 4.6
		4	Max-pooling	32M × 84 × 84N	2 × 2	14 × 14	5.9 × 5.9
		5	Convolutional	64M × 84 × 84N	3 × 3	26 × 26	11 × 11
		6	Max-pooling	64M × 42 × 42N	2 × 2	32 × 32	13 × 13
	D	7	Convolutional	64M × 42 × 42N	3 × 3	56 × 56	24 × 24
		8	Max-pooling	64M × 14 × 14N	3 × 3	80 × 80	34 × 34
		9	Convolutional	32M × 12 × 12N	3 × 3	152 × 152	63.8 × 63.8
		10	Max-pooling	32M × 12 × 12N	3 × 3	224 × 224	94.1 × 94.1
		11	Fully-connected	256N		512 × 512	215 × 215
		12	Fully-connected	128N		512 × 512	215 × 215
		13	Fully-connected	4N		512 × 512	

Table 2

category. The model was run for 25 epochs on the training data and took approximately 7.76 hours to complete.

VI. Internal Workload Distribution

Tensorflow forms the backbone of the computation in both the approaches. In the first approach the DeepImageFeaturizer is used to load the tensorflow graph and apply the InceptionV3 model weights on the graph. A copy of the graph will be sent to each worker node along with slices of the partitioned training data as dataframes. The same graph will train on separate chunks of the data and learns the features. This phase requires much computational power and the Amazon EC2 instances powered by GPUs provide the same. After each worker node finishes training, the output feature vectors from each worker node are collected and send to the driver node. The driver processes the feature vectors from each worker and returns the final features as a MLlib vector.

In the second approach using the custom CNN model, the training is done on the driver node only as fully functional distributed training of Keras on Spark is not yet available in the scientific community. After the training phase the model weights will be saved. The weights file will then be passed to the KerasImageFileTransformer class which loads a tensorflow graph and applies the model weights on the graph. Identical copies of the graph will then be sent to all worker nodes. The validation dataset is then partitioned and sent to the worker nodes for prediction. Further processing will be similar to the first approach. Unlike the first approach the final output will be a dataframe containing the predicted labels and the actual labels.

VII. Results

Spark DL + Logistic Regression		
Iterations	Run Time (mins)	Accuracy (%)
5	14.93 + 10.16	79.4
10	10.17 + 10.75	82.7

Table 3

Spark DL + Support Vector Classifier		
Iteration	Run Time (hrs)	Accuracy
5	1.44	80.46
10	2.54	81.96

Table 4

Convolutional Neural Networks					
Training Samples	Validation Samples	Epochs	Run Time (hrs)	Training Accuracy	Validation Accuracy
2872	397	25	7.76	85.51%	91.93%

Table 5

The first approach is divided into two parts. In the first part we used the features extracted from the DeepImageFeaurizer as inputs to the logistic regression class from the MLlib library. In the second part the feature extracted from the DeepImageFeaturizer as inputs to SVM class from the MLlib library. The model was run separately for 5 and 10 iterations and the results are shown in Table().

Even though the SVM produced better accuracy compared to logistic regression the run-time for the former was much higher compared to the latter.
In the second approach, using a custom made CNN model in Keras, we achieved a training accuracy of 85.1% and a validation accuracy of 91.93% using 2872 images and running for 25 epochs. The previous work used as reference had achieved an accuracy of 81% using CNN and 83% using CNN+SVM [2]. They had used a smaller dataset with 249 images which were heavily preprocessed, and subjected to data augmentation to breakdown the 249 images into smaller varied chunks that added up to almost 7000 images. The model was trained on this augmented dataset for 50 epochs.

The second approach presented in the report required 2872 images and gave better accuracy for half the number of epochs.

VIII. Challenges

As we were completing the project, we encountered a few obstacles. The main issue was the lack of publically available datasets and the relevant code for running those models. The ones that were available dealt with either CIFAR-10 or MNIST which are used for fairly simple objects. The CIFAR 10 dataset is composed of images with a pixel size of 32x32 and MNIST is composed of images with a size of 28x28. But the dataset we used for image classification was much larger with a size of 700x600 pixels. Compressing our dataset to fit the range of the CIFAR-10 or MNIST dataset didn't work as it would diminish the quality of the images thus making it more difficult to extract the features needed for the image classification.

Furthermore, distributed convolutional neural networks in image processing and parallelizing Tensorflow are a very niche and active area of research. Especially in regards to pre-training the convolutional neural network to classify images. As such, there is a lack of documentation which is crucial in advancing the research. Convolutional neural networks depend on a variety of parameters such as the

number of layers, the number of filter, the learning rate, etc. The subject matter is very vast and covers a wide range of topics. Due to the time frame of the project, we were not able to analyze all the topics in depth and instead focused on the two papers mentioned previously that served as the basis for our model.

Finally, the last problem we encountered was finding a platform that can meet the computational needs of the project and provide the required libraries. There were a lot of dependencies on Tensorflow, SparkDL, Keras and integrating these libraries with Amazon Web Services. Ultimately, we decided Databricks was the ideal platform to use for our purposes.

IX. Future

Deep learning is a heavily researched field, but its application and integration with convolutional neural networks and image processing has only recently been gaining traction. Our model, although it utilizes CNNs, can only classify images as benign or malignant. Additionally, this model only analyzes images from a pre-selected region rather than the breast as a whole. Future work can focus on expanding the classifiers such that the model can identify not just the presence of cancer, but also the type of cancer (i.e.) Adenosis, Fibroadenoma, Phyllodes or Tubular. Having these expansions in place would also increase the practicality of the project.

Work can also be done to use deep convolutional nets on large sets of streaming images or video sequences where the structure of the region in question provides the information that is missing or less obvious in static images. This would expand the model so that it's able to identify the proliferation of cancer cells. We also only analyzed a dataset from one source. In order to confirm the results and verify the accuracy it would be ideal to expand the dataset to include images from different sources.

One obvious expansion is the scope for real-time prediction. As the current model only classifies images that have been obtained through biopsy, integrating the model with systems that are capable of real-time prediction would be of tremendous help to pathologists and doctors as it would make the rate of diagnosis more efficient and more accurate.

Finally, enhancing the research in implementing distributed CNNs on frameworks such as Tensorflow, Keras or Theano would create more documentation on this topic which in turn would help progress and develop this field.

X. Conclusion

Two different deep learning based approaches were introduced for the classification of stained histological breast cancer images is proposed. Images are classified as either benign or malignant forms of cancer. The Spark DL library, MLlib library and the concept of transfer learning were integrated to perform the classification on 2872 biopsy tissue images. An accuracy of 82.7% was achieved using Spark DL + logistic regression and the Spark DL + SVM pipeline yielded an accuracy of 81.96%, after 10 iterations. A second method was implemented where a CNN model was built from scratch using Keras application and was trained on the aforementioned number of images for 25 epochs. The model achieved a training accuracy of 85.51% and a validation accuracy of 91.93% on a validation dataset of 400 images. A comprehensive study on how the workload is distributed on Spark and also the differences between the two methodologies were discussed. Finally, since the custom made CNN model gives promising results, the proposed model can be extended to distinguish between the subclasses within the benign and malignant tissues, and also for studying the proliferation of cancer cells over time

XI. Referances

1. Ferlay J, Soerjomataram I, Ervik M, et al; International Agency for Research on Cancer. GLOBOCAN 2012 v1.0, Cancer Incidence and Mortality Worldwide: IARC CancerBase No. 11.

2. Araújo T, Aresta G, Castro E, Rouco J, Aguiar P, Eloy C, et al. (2017) Classification of breast cancer histology images using Convolutional Neural Networks. PLoS ONE 12(6): e0177544. https://doi.org/10.1371/journal.pone.0177544

3. Spanhol, Fabio & Soares de Oliveira, Luiz & Petitjean, Caroline & Heutte, Laurent. (2015). A Dataset for Breast Cancer Histopathological Image Classification. IEEE transactions on bio-medical engineering. 63. . 10.1109/TBME.2015.2496264.

4. Howlader N, Noone AM, Krapcho M, Miller D, Bishop K, Kosary CL, Yu M, Ruhl J, Tatalovich Z, Mariotto A, Lewis DR, Chen HS, Feuer EJ, Cronin KA (eds). SEER Cancer Statistics Review, 1975-2014, National Cancer Institute. Bethesda, MD, https://seer.cancer.gov/csr/1975_2014/, based on November 2016 SEER data submission, posted to the SEER web site, April 2017.

5. Siegel, R. L., Miller, K. D. and Jemal, A. (2017), Cancer statistics, 2017. CA: A Cancer Journal for Clinicians, 67: 7–30.

6. Torre, L. A., Bray, F., Siegel, R. L., Ferlay, J., Lortet-Tieulent, J. and Jemal, A. (2015), Global cancer

statistics, 2012. CA: A Cancer Journal for Clinicians, 65: 87–108.

7. J. E. Joy *et al.*, Eds., *Saving Women's Lives: Strategies for Improving Breast Cancer Detection and Diagnosis*. Washington, DC, USA: Natl.Acad. Press, 2005.

8. NationalBreastCancerFoundation. Breast Cancer Diagnosis; 2015. Available from: http://www.nationalbreastcancer.org/breast-cancer-diagnosis.

9. López C, Lejeune M, Bosch R, Korzynska A, García-Rojo M, Salvadó MT, Alvaro T, Callau C, Roso A, Jaén J. Digital image analysis in breast cancer: an example of an automated methodology and the effects of image compression. Stud Health Technol Inform. 2012;179:155-71.

10. Kowal M, Filipczuk P, Obuchowicz A, Korbicz J, Monczak R. Computer-aided diagnosis of breast cancer based on fine needle biopsy microscopic images. Computers in Biology and Medicine. 2013; 43 (10):1563-1572.

11. George YM, Zayed HH, Roushdy MI, Elbagoury BM. Remote computer-aided breast cancer detection and diagnosis system based on cytological images. IEEE Systems Journal. 2014; 8(3):949-964.

12. Wang, H. et al. Mitosis detection in breast cancer pathology images by combining handcrafted and convolutional neural network features. Journal of Medical Imaging 1, 034003–034003 (2014).

13. Spanhol FA, Oliveira LS, Petitjean C, Heutte L. A Dataset Breast Cancer Histopathological Image Classification using Convolutional Neural Networks. IEEE Transactions on Biomedical Engineering. 63 (7). 1455-1462. 2016