Clinical Diagnosis Support with Convolutional Neural Network by Transfer Learning

Spencer Fogleman  
*SMU, sfogelman@mail.smu.edu*

Jeremy Otsap  
*SMU, jotsap@smu.edu*

Sangrae Cho  
*SMU, sangraec@mail.smu.edu*

---

Follow this and additional works at: [https://scholar.smu.edu/datasciencereview](https://scholar.smu.edu/datasciencereview)

Part of the Cancer Biology Commons, Data Science Commons, and the Diagnosis Commons

---

**Recommended Citation**

Available at: [https://scholar.smu.edu/datasciencereview/vol5/iss3/2](https://scholar.smu.edu/datasciencereview/vol5/iss3/2)

---

This Article is brought to you for free and open access by SMU Scholar. It has been accepted for inclusion in SMU Data Science Review by an authorized administrator of SMU Scholar. For more information, please visit [http://digitalrepository.smu.edu](http://digitalrepository.smu.edu).
Clinical Diagnosis Support with Convolutional Neural Network by Transfer Learning

Jeremy Otsap, Sangrae Cho, Spencer Fogelman, and Jeon Lee

Master of Science in Data Science, Southern Methodist University, Dallas TX 75275 USA
(jotsap, sangraec, sfogelman) @smu.edu

Lyda Hill Department of Bioinformatics, UT Southwestern Medical Center, Dallas TX 75390
Jeon.Lee@UTSouthwestern.edu

Abstract. Breast cancer is prevalent among women in the United States. Breast cancer screening is standard but requires a radiologist to review screening images to make a diagnosis. Diagnosis through the traditional screening method of mammography currently has an accuracy of about 78% for women of all ages and demographics. A more recent and precise technique called Digital Breast Tomosynthesis (DBT) has shown to be more promising but is less well studied. A machine learning model trained on DBT images has the potential to increase the success of identifying breast cancer and reduce the time it takes to diagnose a patient, leading to faster treatment. In this study, a Convolutional Neural Network (CNN) was trained on an open-source dataset from Duke of DBT images belonging to patients with no, benign, and malignant tumors. The model was designed to identify the presence of a tumor (both malignant or benign) or its absence. Robust open-source datasets of medical images are scarce due to the nature of medicine. Deidentifying medical images is very time-intensive, and labeling the dataset requires the expertise of a medical professional, in this case, a radiologist. The open-source dataset was small and imbalanced, so transfer learning, under-sampling the more prevalent healthy patient class, and image augmentation was used to improve prediction accuracy. Training a CNN is very computationally expensive, and a high compute VM environment with extensive RAM was created to facilitate learning the weights of a CNN.

1 Introduction

Breast cancer is the second most common cancer among women in the United States. Statistically, one out of eight women will have breast cancer. As a result, screening for breast cancer is one of the most prevalent medical imaging tasks, with over 39 million exams each year [4]. Screening techniques produce images that are reviewed and annotated by radiologists in
identifying tumors. Digital Breast Tomosynthesis (DBT) is a relatively new imaging process and has replaced mammography as the current standard of care for breast cancer screening. DBT is essentially a 3D mammogram. An X-ray will swing over the patient's breasts to generate a 3D image set in one-millimeter layers, leading to more precise breast imaging technology [3]. It also contains the same amount of X-ray exposure as a traditional mammogram and is well within the safety limits set by health authorities [7]. A 2D mammography looks at all the tissues in the breasts simultaneously, leading to the issue that tissue features can overlap in images and lead to misleading conclusions. Some previous issues from 2D mammography include overlapping healthy tissues looking cancerous and tiny cancers being hidden [2]. Doctors also must call back patients for a second screening if the 2D mammograms are inconclusive. With DBT, radiologists can look at many layered images before making a classification, helping to remedy the issues present with 2D mammograms. An application of such a model assists radiologists in diagnosing breast cancer or creating automatic triaging of cancer patients. Radiologists are not always accurate in determining cancer prognosis, and using a supplemental machine learning model may decrease their error rate.

Deep learning models have successfully classified images due to a rapid increase in access to computational resources and large-scale labeled data. However, organizing medical images is much more complex than other image types. There are many challenges in acquiring labeled radiology images for machine learning [4]. First, it is difficult to gain access to extensive collections of medical images, hindering utilizing machine learning, which favors being trained on as many images as possible. There is also a significant class imbalance between images containing tumors versus those without tumors. In this study, there were only images corresponding to 137 patients with cancer, compared to images for thousands of patients without cancer. Second, accessing and sharing medical images requires a very comprehensive compliance review and deidentification by the institution that owns the images to protect themselves from liability if patient information was leaked [4]. In this study, all patients' information has been edited to remove any information that could reveal a patient's identity. This makes collaboration on medical imaging projects difficult. Finally, creating labels for medical imaging datasets requires the expertise of radiologists [4].

2 Data

Duke curated the dataset used in this study to overcome the previously stated challenges. A machine learning model could potentially increase the accuracy of cancer identification over the radiologist alone and decrease the time it takes to identify a tumor, leading to faster treatment of the patient's cancer. We analyzed DBT volumes obtained from Duke Health System [4]. Specifically, Duke Health Systems' DEDUCE (Duke Enterprise Data Unified Content Explorer) tool was queried to get all radiology reports having the word
tomosynthesis,' and all pathology reports having the word 'breast' within the search dates of January 1, 2014, to January 30, 2018 [4].

DICOM images from a patient are a set of 2D slices from different views. There were four views included in the image dataset:

1) LCC (left craniocaudal),
2) LMLO (left mediolateral oblique)
3) RCC (right craniocaudal)
4) RMLO (right mediolateral oblique)

Each patient has multiple images for each view created from DBT. Patients were broken down into four groups [4]:

1) Normal group: DICOM images belonging to this group had no sign of cancer, and a biopsy was never performed. This group included 1,680 studies from 1,680 patients, totaling 120,028 images.
2) Actionable group: This group resulted in further imaging because cancer seemed possible, but a biopsy was not performed. This group was excluded from the analysis.
3) Benign group: DICOM images belonging to this group showed signs of cancer. A biopsy was performed, and the tumor was classified as benign by a radiologist. This group included 137 studies from 137 patients, totaling 8,722 images.
4) Cancer group: DICOM images belonging to this group showed signs of cancer. A biopsy was performed, and the tumor was classified as malignant by a radiologist. This group had 86 studies from 86 patients, totaling 5,531 images.

The biggest challenge with creating models using medical image data is that positive cases (Ex: cancer is present) are typically an asymmetrically small proportion of patients in a dataset. This is defined as imbalanced classification, a skewed distribution of class. Most machine learning models will suffer from lower performance because of this skewed class distribution. The majority class is a normal case in this domain, which means fewer positive cases to learn from the dataset. It can easily obtain a very high recall at the expense of precision and vice-versa. In this study, patients with benign and malignant tumors were combined into a single "tumor" class. This increased the number of images in the positive class and simplified the target from three classes to two classes: tumor and no tumor. Furthermore, we experimented with the model learning from two different versions of the data.

2.1 Data strategy 1: Using all the images taken for each patient

A single patient will have many images associated with them after a DBT scan. About 70 images are taken at four different angles, creating about 280 images for a single patient. Radiologists will only look at one or two images when trying to identify the presence of tumors. In the first dataset, we took all images for any patients with either malignant or benign tumors. We randomly
sampled 15% of the images of healthy patients out of the hopes the model would learn the features of the tumor. Because there are multiple images per patient, we also had to make sure no patient images were in both the training and testing sets. This would result in data leakage, and the model would learn features of the patient rather than of the tumors, and in turn, would not generalize well to new patients. Figure 1 summarizes the data preparation strategy.

Figure 1: Data prep using all image

2.2 Data strategy 2: Using a single image for each patient

Instead of using all the images associated with a patient with tumor presence, we can choose only the best quality image that the radiologist reviewed. To increase the positive class size, image augmentation was used. Image augmentation is the slight modification of an image to create a "new" image based on it. Some techniques for image augmentation are zooming in on an image, horizontally shifting an image, vertically shifting an image, changing the image brightness, or rotating an image—figure 2 outlines this data preparation strategy.
Preprocessing:

The DICOM format (.dcm files) is the current standard used for medical images and medical videos [2]. This includes images produced from X-rays (such as in DBT), CT scans, MRIs, ultrasounds, etc. DICOM allows different image types from different manufacturers and techniques to be viewed on a single computer by a radiologist [6]. Saving all healthcare images in a universal format allows easy transfer, storage, collaboration in hospital systems. DICOM also stores metadata about a patient [2].

However, DICOM images must be converted into a more traditional image type, such as JPEG, to be fed into a machine learning model. Compressed DICOM images were first converted into a series of 2D JPEG images. Only one JPEG image was chosen to represent a patient. In our final dataset, we had 1680 JPEG images belonging to the normal group, 86 JPEG images belonging to the cancer group, and 137 images belong to the benign group.

Cancer imaging is naturally imbalanced, as only 1% of screening exams result in a cancer diagnosis [4]. The study used data augmentation to increase
the number of images containing cancer artificially. Data augmentation involves taking existing images and creating new images through various methods, such as rotation, cropping, changing the brightness, or zooming in and out of the image.

3 Methods/Results

3.1 Method 1: Using all images to build the model

Our first iteration involved using all images associated with a patient. We used transfer learning with two models: VGG15 and Inception v3. VGG15 is a population convolution neural network (CNN) developed by K. Simonyan and A.Zisserman from Oxford. VGG15 achieves a 92 percent accuracy on the popular ImageNet dataset, consisting of 14 million images from 1000 classes. Inception v3 is another population CNN from Googlenet, and it achieves a 93 percent accuracy on the ImageNet dataset. Both models have two parts: a layer of convolution to extract features and a fully connected layer to combine these features into classifications. Using transfer learning saves time optimizing a neural network to learn the features of an image and saves the computational power required to understand the said features.

3.1.1 Baseline model:

Our first baseline experiment used transfer learning with VGG with the following hyperparameters: an optimizer of stochastic gradient descent (SGD), the learning rate of 0.001, the momentum of 0.9, batch size of 64 images, images of size 224x224 pixels, and ten epochs of training. Key features of the baseline VGG model include stacked convolutional layers with small 3x3 filters followed by max pooling. Each layer uses a relu activation function with He weight initialization. This baseline model overfits the training set. By looking at the loss curves, we see that the training set levels off after only two epochs and that the testing set accuracy did not improve over the epochs. The test accuracy was around 55%. Figure 3 shows the loss and accuracy curves for the baseline model.
3.1.2 Baseline model with augmentation:

Our next iteration involved using image augmentation to increase the number of images in our training set artificially. Training deep learning neural network models on more data can result in more skillful models [20]. The augmentation techniques we used were random horizontal and vertical flipping of the images and random rotation of the images by up to 90 degrees. We also added early stopping so that the model would stop updating its weights after learning.

Here we see that test set accuracy improved to around 62%. However, the test set stopped learning around 11 epochs while the training set continued to improve, showing overfitting was still happening. Figure 4 shows the accuracy and loss curves.
3.1.3 Adding more dense layers

We only had one dense layer of 100 nodes in the previous models after flattening before our final binary classification layer. There are now four dense layers with the following nodes: 8192, 2048, 512, 128. With this neural network structure, the test accuracy improved to 64%. The accuracy was achieved only after epoch two, while the training accuracy improved, indicating overfitting. Figure 5 shows the loss and accuracy curves.
3.1.4 Using Adam optimizer

The previous models used the SGD optimizer. We switched to the Adam optimizer to see if the model would improve. Overfitting is still an issue, as seen in Figure 6.
3.2 Method 2: Using balanced single slice image with augmentation

The current best model achieves an accuracy of 64%. All the previous models utilized black and white images of size 224x224 pixels. We chose a smaller image size to increase the speed of training but hypothesized that larger images could learn more complicated features. However, if the images were too large, the computer could run out of RAM. We decided to increase the training image dimensions to 500x500 pixels.

3.2.1 Baseline model:
We continued to use the VGG model with transfer learning with the following hyperparameters: optimizer of SGD, the learning rate of 0.001, batch size of 64 images, and early stopping with patience of 20. By increasing the image size, our test accuracy jumped to around 90%, almost a 36% increase in improvement from our previous iteration. Our training and testing accuracy continued to improve throughout the epochs, indicating that overfitting is no longer a problem. Figure 7 shows the loss and accuracy curves.

![Cross Entropy Loss](image)

![Classification Accuracy](image)

Figure 7: baseline model with augmentation (using single slice only)

We could have set this model as our final model; however, we added another transfer learning with Inception v3 compared to VGG 16.

3.2.2 Using Inception model
We were interested to see if switching to Inception v3 would improve over VGG16. Inception v3 is a more complicated model than VGG16. It involves the heavy use of 1x1 convolutions, error feedback at multiple points in the network, very deep models (22 layers), and using a global average pooling function after convolutions. We added a final dense layer of 128 nodes to try and tailor it towards tumor detection and continued to use image augmentation. The model's accuracy jumps to 94%, but the model stops learning after early epochs for both the training and testing sets. The loss and accuracy curves are shown in Figure 8.

![Cross Entropy Loss](image)

![Classification Accuracy](image)

Figure 8: Loss and Accuracy Learning Curves with Inception (single slice)

### 3.2.3 Partial inception v3 (Case I)

We next tried to use a shallower model by cutting layers in Inception v3. After the "mixed 3 layer" in the original Inception model, we cut off all layers and used the same hyperparameters. The testing accuracy is lower at 89.78%, but early saturation of learning and overfitting is not present. However, we can see very large fluctuations in the loss and accuracy between epochs. Figure 9 shows the loss and accuracy curves.
We next tried to include more layers by cutting off the Inception model at a later layer, the "mixed five layers." Our testing accuracy jumped to 94.9%, which is now the best performing model. Early saturation and overfitting are still not an issue, and now the fluctuations in both loss and accuracy are gone. Figure 10 shows the loss and accuracy curves.

3.2.3 Partial inception v3 (Case II)

3.2.4 Partial inception v3 (Case III)
For our final model, we increased the depth of our network to the "mixed seven-layer." The test accuracy was not as high as the previous model at 93.3%. However, the recall score was 95%. Recall quantifies the ability of the model to find patients with tumors. Since missing a patient with a tumor is extremely detrimental, we decided to use this model. Figure 11 shows the loss and accuracy curves.

Figure 11: Partial Inception (single slice) by mixed 7

4 Summary of Results and Best model

A summary of the results is provided below:

Using all the images
1. VGG16 baseline : 54.0% accuracy
2. 1 + Data Augmentation: 62.0% accuracy
3. 2 + adding 3 more layers : 64.0% accuracy
4. 3 with Adam optimizer : 64.0% accuracy

Using the single best quality image reviewed by a radiologist
5. VGG16 : 91.4% accuracy
6. Inception v3: 94.5% accuracy
7. Partial Inception v3(mixed3): 89.8% accuracy
8. Partial Inception v3(mixed5): 94.9% accuracy
9. Partial Inception v3(mixed7): 93.3% accuracy

Figure 12 summarizes both the accuracy and recall of our models. We can see that recall is maximized with the partial inception v3 (mixed 7) model. Although the accuracy is not as high as some other models, it is still relatively high and justifies its use as the final model.

5 Discussion & Conclusion

5.1 Lessons Learned

First, when taken, mammography images are in a dicom format, which makes transfer across a hospital system much easier, does not lend itself to being fed into a machine learning algorithm. The Dicom to JPEG conversion required rather complicated code that required extensive knowledge about the underlying Dicom format. A thorough analysis of the published paper that accompanied the data source was needed to discover the correct dependencies that allowed the Dicom file to be created and undone to make the JPEG images.

Second, the actual format of the images made extreme differences in the processing time of the machine learning algorithm. Converting the image into a two-dimensional array of black and white pixels would accelerate the training and scoring times, but possibly at the expense of losing more complicated
feature extraction by the convolutional layers of the CNN [41]. The image size was also significant. Small image width and heights would lead to faster training and scoring times. Increasing the image size would lead to better accuracy, but if too large could saturate RAM and stall the training process. Cloud computing can help this problem, but not for free.

Finally, with image studies with many images per patient, the training and testing split must be stratified. A simple random split would result in images of a patient in both the training and test set, a phenomenon known as data leakage. In this situation, the model would learn features of the patients rather than the tumors, giving a highly inflated accuracy. This model would not generalize well to new patients.

5.2 Limitations

Despite the advances made in machine learning, both in this specific scenario of diagnosing cancer and the general discipline, it is still a new technology whose adoption has grown exponentially in the past decade [35]. Largely due partly to cloud adoption, which facilitates highly scalable compute that can be billed on an as-needed basis rather than requiring large capital investments upfront. Big data farms can scale out to hundreds of processors for the duration of model training and then scaled back afterward.

There are, however, limitations and obstacles around adopting ML (Machine Learning), and not all of them are respective of the technology. First and foremost, one of the biggest challenges isn't a result of the technology itself but a lack of skilled resources. Not just for the technical resources who develop the ML models, but also a lack of strategic acumen in the understanding and application throughout organizational leadership. Even though 93% of executive leadership view AI as a critical business need [36], their lack of awareness and data literacy can cascade downward to affect multiple levels of company culture. A data scientist can create an exceptionally performing model; however, if the organization doesn't correctly leverage it or integrate it with relevant business processes, it cannot derive any tangible value.

A genuine limitation of any exercise specifically for the medical field is around the availability of patient images. Unlike a more generic deep learning exercise, such as identifying a specific product category from an image online, patient data is classified as sensitive information and falls under the privacy legislation of HIPPA compliance. Because of that, we only had access to very limited set of images. It was not possible to obtain more to tune the model further or validate how well the model generalizes on new images. In addition to this, the patients are fully anonymized, so it is not possible to validate or refute sample bias. Where these mammograms all from a sample of women that accurately reflects the general population? Or were they all between the ages 40 – 55 and from the same clinic in a high-income area like Orange County? Any type of machine learning will have better alignment to reality when trained on more data.
The quality of data additionally complicated this. Specifically, image consistency can have a dramatic impact on model performance [37]. Many images are blurry or of low quality in this study, which can harm the model performance. The model will be trained with non-distinguishable features due to their blurriness. The combinations within our second data strategy, however, do show that the model was learning well, and we have reasonable control over the rate of learning without overfitting.

Looking more broadly at the problem in the context of medical imaging, there are other challenges, such as a human error in the diagnosis that creates the initial training labels and variation between the imaging equipment or the radiologists using it. Specifically, with Convolutional Neural Networks (CNN), variation in factors such as image resolution, contrast, or rotation can make it challenging for the image to be classified correctly. [38]

Certainly, there are methods in which this limitation can be mitigated. To deal with these challenges, we artificially expanded our repository of images via augmentation. Unfortunately, augmentation techniques essentially duplicate images with minor adjustments such as the percent zoomed in or out, horizontal or vertical rotation, and image shift.

Lastly, the class imbalance is the norm for real-world medical research, as positive cases are rarer. Specifically, in this study, it is less than 10 percent.

5.2 Future Research

There are many reasons why future studies would be needed to validate the usefulness of this model. First, we did not have a holdout set because of the extremely small dataset we were working with. It would be advisable to try and see if the model would generalize well to a new cohort. It would also be interesting to see if the model would be generalized to patients outside of the Duke Hospital systems, from which all the images were taken.

Second, the images did not contain any patient demographic information because this is considered PHI, and releasing this information would make the hospital liable. Repeating this image with known metadata is paramount. Not knowing the demographic information of the patient could lead to high model bias. For example, if the model was only trained on white women, would it generalize well to women of other ethnicities, races, and nationalities? Other demographic considerations will be to see if the model performs better on specific age groups. Even in the current process of radiologist review, a radiologist is much more accurate in diagnosing certain age brackets [42].

Despite our limitations, we achieved a greater than 90 percent recall score. In the future, we might be able further to improve our results with additional regularization such as 1) more aggressive dropout in each layer, 2) further weight decay, and 3) hyperparameter tuning. However, this requires building a model from scratch, not using transfer learning; even with cloud computing resources, this can take days or weeks to compute, depending on cost constraints, which are not insignificant. Note that we limited our focus to a classification problem due to constraints around time, money, and domain...
knowledge, as well as the provided data set. Adding detection algorithms with transfer learning can be a great candidate for further study.

5.3 Ethical consideration

Unlike other use cases for image-based deep learning, such as retail or real estate, a Type 2 error risk is highly asymmetric. The result could potentially be loss of life. Given how new AI adoption is, clear legal guidance doesn't exist. [39] If the model fails, where does the blame lie? With the data scientist who built the model? With the medical facility that has adopted machine learning as part of their treatment? With the manufacturer of the X-ray machine which took the original images used in the training set? With the study organizer, who may or may not have properly selected patients for the training sample? Was the patient made aware that their diagnosis was the result of a machine learning algorithm? Does the doctor treating the patient adequately understand how the algorithm decided and then explain this to the patient?

5.4 Final Thoughts

In general, Deep Learning algorithms need vast amounts of data to adequately train the model, especially with computer vision-based cognitive services. In practice, it is quite a challenge to be provided with a sufficiently large data set, and high-quality well-curated medical images are even rarer. Medical institutions may not be able to wait days or weeks of training to build a high-performing Convolutional Neural Network model; conversely, individual researchers may not afford the cost of cloud computing resources to accelerate the time needed for model iteration. Transfer learning can be the great alternative solution when dealing with these constraints.

Despite these challenges, with Machine Learning, there is a compelling opportunity to leverage automation by operationalizing a model into a clinical process [40]. This can help minimize human bias in the diagnosis and improve consistency, whether specifically mammograms or other types of a cancer diagnosis. The benefit for the patient is reducing additional radiation exposure. The benefit for the medical organization is improved time efficiency, which can translate a variety of outcomes, such as helping more patients or focusing on developing new treatments. Clearly, AI-based technology is only going to get more pervasive. And its successful adoption is going to require that organizations adopt a strategy for change management, as well as our governing bodies and their respective legislation.
6 Technical Appendix

The purpose of the appendix is to briefly summarize an abbreviate view of the different technologies that were part of this effort. While all were researched and evaluated, not all were used to execute and deploy the final model, as they were deemed inappropriate or suboptimal given project constraints.

6.1 Traditional Hadoop Architecture:
Hadoop is a Java-based big data platform designed to run on inexpensive commodity hardware, which processes data in parallel via a Map-Reduce paradigm. At its simplest, Map-Reduce splits input data into separate units, organized and/or sorted as the "map" operation, and then aggregated as the "reduce" operation [25]. It is made up of 2 core services—the Storage Service and the Orchestration Service:

1. Storage Service: Hadoop Distributed File System (HDFS) as the name implies, is a file system meant to be run across inexpensive commodity hardware, where redundant blocks of data are replicated or distributed across the hardware landscape, and where data-locality is tracked through rack-aware topology mechanisms [23]. See Figure 14.
   - Distributed storage service designed to run on commodity hardware
   - Data redundancy natively built-in; 3 copies of each data block replicated across all participating nodes in the Hadoop cluster
   - Rack-awareness is maintained in metadata, thus optimizing data resiliency and performance
   - Based on Master / Slave architecture
     - NameNode (Master) – maintains in-memory storage metadata about HDFS structure, node names, and block allocation table
     - DataNodes (Slave) – worker nodes that communicate with the NameNode about changes to HDFS, or updates during local computations
2. Orchestration Service: The YARN Daemon is responsible for the split up map-reduce jobs and distributing low-cost commodity hardware. Tracks resource availability and managed job scheduling across the available compute landscape. [24] See Figure 15.
   - ResourceManager – the master service for the cluster that runs on one of the head nodes
     - Responsible for allocating cluster's resources and job scheduling on worker nodes
     - Runs on the master node
     - Global resource scheduler
     - Schedules node resources across applications and addresses resource contention
   - NodeManager – one instance per worker node
     - Runs on worker nodes
     - Runs periodic keep-alive checks with ResourceManager
     - Keeps track of available resources on each node and communicates back to ResourceManager
   - ApplicationMaster – a master service unique per application.
     - Coordinates the execution of an application in a cluster
     - Resource negotiation with the ResourceManager for the processing needs of the application
     - Runs as a Java JVM container
6.2 Spark Cluster Internals

Spark is the next technology iteration of big data processing, building upon the innovation created by Hadoop. It has a number of improvements over Hadoop, including improved performance of in-memory processing. Spark will optimize the dependencies and sequence of operations, increasing up to 100 times faster than Hadoop's purely disk-based processing [26].

Unlike Hadoop, which relies on an ecosystem or "zoo" of utilities to handle various data scenarios (i.e., Hive, Pig, Sqoop, Oozie, Kafka, Mahout, etc.), Spark has standardized on a set of API functionality that is natively built into the platform. See Figure 16.
Figure 16: Spark Cluster Ecosystem

- **Spark Core**: Spark Core is the base engine for distributed data processing. It is responsible for memory management and fault recovery, scheduling, distributing, and monitoring jobs on a cluster & interacting with storage systems. Natively supports APIs for Scala, Python, R, SQL, and Java
- **Spark Streaming**: Spark Streaming is the component of Spark that is used to process real-time streaming data like Kafka, Flume, etc
- **Spark SQL**: Spark SQL integrates relational processing with Spark's functional programming API. It supports querying data either via SQL, Hive, and JDBC/ODBC connections
- **GraphX**: GraphX is the Spark API for graphs and graph-parallel computation (i.e., vertices/edges).
- **MLlib**: MLlib stands for Machine Learning Library. Spark MLlib is used to perform machine learning in Apache Spark.
- **Cluster Manager**: Spark currently supports three different flavors: Spark Standalone, YARN, and Mesos

Specifics of its architecture are detailed below [28] see Figure 17.
Spark Driver runs the user's main program and breaks it down into individual tasks, then distributes operations across the group of Worker Nodes [i.e., the "cluster"]
- Executors process data and code specific to a respective application
- The Spark Driver communicates with each application's isolated Executor to divide the work as multi-threaded tasks

Worker nodes have individual tasks run on local compute resources. Local operations cache these partially-completed results in-memory as an intervening step into Resilient Data Sets (RDDs) for faster performance.

RDDs are the immutable core units of data, designed for parallel processing and fault-tolerant or "resilient" operations, distributed across the HDFS data store or persisting in cluster memory. DataFrame API and Dataset API are built on top of RDDs when working on semi-structured or tabular data. However, for media files such as images or video RDDs are the primary data construct [29].

The Cluster Manager monitors available compute resources across the cluster and then schedules individual tasks based on resource availability.
• Worker Node's local results are collected and aggregated by Spark Driver to produce the final output
• Databricks has added its proprietary storage service. Each Worker Nodes creates a mount point to various supported storage platforms, which are then abstracted as a single virtual storage construct called Delta Lake. Delta Lake is treated as a single data lake, allowing for highly parallelized data ingress/egress. Worker Nodes read/write to the Delta Lake, while Cluster Manager tracks the physical allocation and master index table [27]
  o Decouples storage from compute allowing each to be scaled independently
  o Enriched metadata that can elastically scale across distributed compute while still tracking and maintaining optimal data locality
  o Highly scalable to handle real-time and interactive in addition to batch
  o Can automatically optimize Spark partitions and caching to enhance query performance
  o Unlike traditional NoSQL data repositories, it allows for levels of consistency akin to traditional ACID systems
  o Can automatically detect variations in schema and enforce desired behavior
  o Native data versioning “Time Travel” that enables point-in-time rollbacks, as well as full audit logs

6.3 TensorFlow on Spark

The TensorFlow framework was used to develop the ML model and was powered by Spark for processing and deployment. Thus, the Spark Executor was responsible for loading the TensorFlow libraries. The Spark Executor reads in each RDD data unit and then passes that to the TensorFlow Core libraries. See Figure 18. (Alternatively, can use TensorFlow QueueRunners, allowing TensorFlow workers to run in the foreground, allowing Spark to retry any TensorFlow failures [30] automatically.)

Databricks greatly simplifies the process of running TensorFlow on Spark.

Can run Bash command in a notebook:

```
%pip install tensorflow-cpu==2.4.*
```

Validate:
import tensorflow as tf
print([tf.__version__])

from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential

# create nn
def create_model():
    model = keras_model_sequential()
    model.add(Dense(512, input_shape=(224,224),
        activation="relu"))
    model.add(Dense(10, activation="softmax"))
    return model

# compile model
model = create_model()

model.compile(
    optimizer = "rmsprop",
    loss="categorical_crossentropy",
    metrics=['accuracy']
)

6.4 Azure Data Science VM Technical Specifications:
There are over two dozen different types of Virtual Machines (VMs) offered by the Azure cloud platform, with a variety of different enhancements and features for specific scenarios. Given that cost was a very real constraint, while still allowing for adequate CPU, memory, and disk throughput, the DS13v2 series VM was selected. Additionally the OS paging file was separated out, as well as the data disk to optimize throughput. Features listed below:

DS13v2 series Virtual Machine
- 4 vCPU – 2.4GHz Xeon E5-2673v3
- 32GB of DDR4 RAM [28GB usable after 4GB overhead]
- All SSD Disks
  - C: %systemdrive% - 128GB SSD
    - AVG: 500 IOPS | 100MB/S throughput
    - MAX / BURST: 3500 max IOPS | 170MB/S max throughput
  - X: data drive - 512GB SSD
    - AVG: 2300 IOPS | 150MB/S throughput
    - MAX / BURST: 3500 max IOPS | 170MB/S max throughput
  - D: Paging / SWAP drive [for paging only] 64GB SSD [2 x RAM]
    - AVG: 500 IOPS | 100MB/S throughput

Tools Included [31]

<table>
<thead>
<tr>
<th>CUDA, cuDNN, NVIDIA Driver</th>
<th>LightGBM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horovod</td>
<td>H2O</td>
</tr>
<tr>
<td>NVidia SM</td>
<td></td>
</tr>
<tr>
<td>PyTorch</td>
<td>Intel MKL</td>
</tr>
<tr>
<td>TensorFlow</td>
<td>OpenCV</td>
</tr>
<tr>
<td>Azure ML SDK</td>
<td>CNTK</td>
</tr>
<tr>
<td>XGBoost</td>
<td>Jupyter Notebook Server / JupyterHub / JupyterLab</td>
</tr>
<tr>
<td>Vowpal Wabbit</td>
<td>Docker</td>
</tr>
<tr>
<td>Weka</td>
<td>Spark 3.1 (Standalone)</td>
</tr>
<tr>
<td>LightGBM</td>
<td>SQL Server (Dev Edition)</td>
</tr>
</tbody>
</table>

Intel Xeon E5 – Deep Learning Enhancements

A full detail of Intel CPU architecture is beyond the scope of this paper; simply making note of Intel's Math Kernel Library for Deep Neural Networks (MKL-DNN) allows improved performance for the following:
- Intel Optimized Caffe
- TensorFlow
- Apache MXNet

**Figure 19:** Training throughput of Intel Optimized Caffe and TensorFlow with ResNet-50, VGG-16 and Inception-v3 with various mini-batch sizes [32]

**Figure 20:** Inference throughput of Intel Optimized Caffe and TensorFlow with ResNet-50, VGG-16 and Inception-v3 with various mini-batch sizes [32]
6.5 Jupyter Architecture:
JupyterHub is a multi-user platform that allows for collaborative development between a team of data scientists using ipynb notebooks. Because the Kernel supports running on Spark, this made for a good starting development environment. [33] [34] See Figure 21.

- **Client**: user session to accept code and send to the kernel on server for processing as notebook documents
- **Kernel**: receives, executes, and processes code sent by Client. Returns results back to the Client. Handles Evaluate operations. Supports Spark for distributed compute context. ZeroMQ is asynchronous communication protocol between Clients and Kernel
- **Notebooks**: ipynb files stored in JSON format

![Diagram of Jupyter Architecture](<Figure 21 – Jupyter on Azure Data Science VM>)

6.6 Data Lake Overview
Data Lakes are a centralized repository storing all of an organization’s assets to house extremely large volumes of diverse data types, allow massive
parallel processing, defer imposing structure or schema, and allow business
time to quantify data's value to the business:

- Structured tables / tabular data
- Semi structured logs
- Semi structured IoT / sensor streams
- Semi structured web feeds
- Unstructured text files
- Media feeds
  - Image files
  - Audio files
  - Video files
- Geospatial / map data

Data Lake Zoning Structure
Functional layout to accommodate different use cases, types, formats,
performance tiers, and methods of access. See Figure 22.
• **Raw Data**: immutable copy of original, unaltered source data in native format; typically limited to data engineers and data scientists

• **Staged Data**: Data with varying schema or format is standardized; if possible normalized into a searchable / tabular format. Functionally consolidated based on related sources or business applications

• **Sandbox**: Workspace for exploratory analysis

• **Fully Curated**: Transformed, cleansed, organized, etc to allow for self-service. Additionally evaluated for correctness and value to business

• **Analytical Serving**: Serving layer for applications and dashboards; typically higher cost low-latency storage to facilitate high concurrency

• **Archive**: Aged data for historical reporting; lower cost cool storage

7 References


7. Hyo-Eun Kim, Hak Hee Kim, Boo-Kyung Han, Ki Hwan Kim, Kyunghwa Han, Hyeonseob Nam, Eun Hye Lee, and Eun-Kyung Kim. Changes in cancer detection and false-positive recall in


