Predicting the Presence of Cancer in Medical Images using Convolutional Neural Networks

---

An Engineering Paper

Presented to

Junior Science, Engineering, and Humanities Symposium

University of Missouri—Columbia

By

Connor Monahan

Senior

559 E. Highway N of
Wentzville, Missouri
63385

November 10, 2016 – January 11, 2017

Desire Kirchofer
Biology Teacher
Timberland High School

Acknowledgements

Desire Kirchofer

Sanjiv Bhatia

Michael Park

National Institute of Health

Wentzville School District

Abstract

Convolutional neural networks model human brain activity from sensory inputs and have been applied to various activities such as driving cars, recognizing speech, and playing complex games. These machine learning algorithms gained popularity in recent years as an effective method for pattern recognition. This paper proposes a novel convolutional neural network architecture to detect lung cancer in radiographic images. Given that patients routinely have x-rays taken for pre-emptive health screenings, and the growing amount of medical data from these imaging procedures, the need for automated detections of abnormalities increases. The solution used a dataset of labeled x-ray images of patients with confirmed instances of lung cancer combined with an equal set of cancer-free patients. The network relied on rectified linear unit activations and dropout to reduce overfitting. Then, this neural network model was trained with a standard backpropagation algorithm with an Adam optimizer using softmax regression. This model demonstrated a 91% accuracy rate on lung cancer chest x-rays.

Table of Contents

Introduction

The rate of lung cancers among the population has dramatically increased during the 20th century, posing a large threat to the public especially those with higher risks of the disease, such as former or current smokers and those with exposure to radiation or chemicals in the workplace. It is estimated that over 1,465,000 people die every year from cancers, 18.2% of which are a variant of lung cancer [1]. The tumors resulting from this disease at a certain stage are visible to experienced radiologists on such mediums as chest x-rays, CT scans, and PET scans, imagery often taken during diagnosis of these diseases or as a preventative measure in several cases. Currently, the low survival rate for lung cancer of around 10% is attributed to its frequent late detection.

Computer vision technology has recently been applied to problems of increasing complexity. Algorithms must find features in more complicated and often nuanced patterns such as those present in medical imagery with a usable degree of accuracy. A specifically tuned feature detector, such as SIFT [2], will not work on different varieties of inputs because it does not have the capacity to adapt as it will not update in response to new parameters. In the modern medical industry, imagery such as x-rays and CT scans are commonly used with computer vision solutions to warn an operator or radiologist about the likelihood of patients having certain conditions, a process known as Computer-Aided Detection [3]. However, these solutions based upon classical algorithms are often limited to certain conditions and data types. For example, a feature detector designed to identify breast cancer will not likely prove useful for tracking lung cancer unless modified. Convolutional neural networks are a promising solution to the problem of medical image analysis due to their ability to generalize based on their dataset. A recent evaluation of a non-medical dataset against expert human annotators proved that networks

trained on computers are useful as the human team performed with a top-5 loss of 5.1% [4]. Many network architectures have been developed for classifying images, such as AlexNet [5] and Google's Inception [6], the former of which won the ImageNet Large Scale Visual Recognition Challenge in 2012 and the latter won the same challenge in 2014 with a loss of 6.67%, narrowly approaching human error for the same database. It is not currently known how these architectures perform on binary classification. In this paper, a new neural network architecture for classifying medical imagery is presented and tested on a public, anonymized medical dataset.

Background

Classical computer vision, in contrast to machine learning technologies, covers the methods used to process imagery using specialized detection algorithms. A popular software programming library known as OpenCV contains the implementations of a multitude of highly optimized algorithms that are applied for research and commercial use to every platform: desktop, server, mobile, and embedded. These functions include image filters like blurring with Gaussian, edge detectors like Laplacian and Canny, the latter based on signal optimization, and segmentation methods such as finding contours and extracting their attributes such as area via Green's theorem and positioning through contour moments. Besides a handful of specific procedures, such as background subtraction, classical computer vision relies on the localization of certain handcrafted features in the image, rather than the automated response of constructing and tuning features based on positive stimulus.

One of the simplest and most popular methods of classifying data with learned features is known as logistic regression. Logistic regression [Equation 1] is used when the problem requires discrete output values, such as benign or malignant for a tumor, in opposition to linear

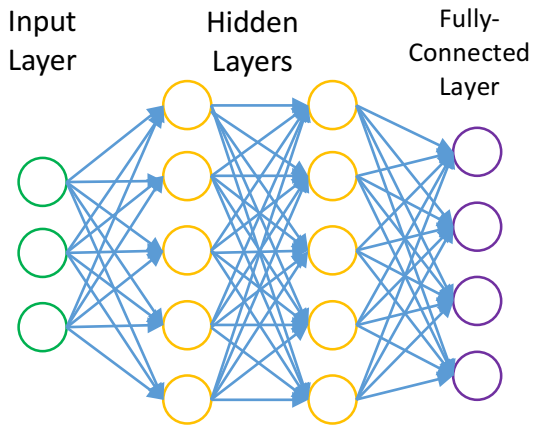
regression, a similar machine learning technique that can be used to predict values from a range. A sigmoid function is used to ensure that the output will map to one of two possibilities. This algorithm has a cost function which takes the current parameters as an input and outputs a cost value, or a number related to the function's distance from convergence. A cost function is also commonly known as a loss function.

Equation 1. Logistic Regression Cost Function

$$J(\theta) = -\left[\frac{1}{m}\sum_{i=1}^{m} y^i \log h_\theta\left(x^i\right) + \left(1 - y^i\right)\log(1 - h_\theta(x^i))\right]$$

Neural networks are another technique of machine learning which attempts to model the human brain to solve classification tasks. These techniques, originally developed in the twentieth century, have experienced a recent revival due to the improvement in computing power and now serve as the state of the art approach for solving a variety of problems from email spam classification to medical research. The simplest neural network consists of an input data layer, multiple hidden layers, and a fully-connected layer [Figure 2]. Each layer consists of several neurons which transform the data based on the specific weights and biases. The input reads the data and labels from a source and subtracts the mean image from the data to prevent close adhesion to the dataset. Hidden layers perform convolutions and pooling to extract features, and fully-connected layers combine the results of the last hidden layer into the correct interval for the output. The last fully-connected layer's dimensions match the intended number of output classes—in the case of ImageNet, 1000, or in the case of binary classification, 2.

Figure 2. Neural network layers



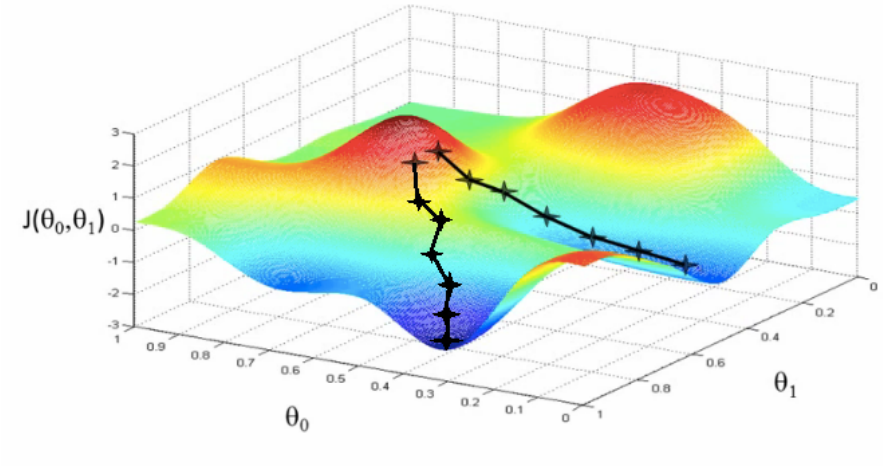
The backpropagation algorithm performs an update of the weights of the network by calculating the error of each layer of the network using the loss function.

Equation 3. Backpropagation algorithm

$$D_{ij}^{(l)} = \frac{1}{m}\Delta_{ij}^{(l)} + \lambda\Theta_{ij}^{(l)}$$

Neural networks are trained using a technique known as gradient descent. This algorithm, used in combination with backpropagation, attempts to find the global minimum of the cost function over the space of the network. In Figure 4, gradient descent is optimizing a cost function J on weights θ. This procedure is usually performed iteratively in which one image is provided to the input data layer which the network then processes, then using the cost function to determine the gradient between this prediction and the ground truth value, and then finally performing backpropagation to update the weights of the previous layers. Today, many alternatives exist to the traditional gradient descent process, such as the Adam optimizer [7].

Figure 4. Model of gradient descent process



Classification of data based on the current weights of the network uses an algorithm called forward propagation, which outputs a hypothesis. This algorithm calculates the activations for each layer in sequence by multiplying the previous value by the weight, subjecting this to the specified sigmoid activation function and summing this value with the bias. As the next layer's activation directly depends on the previous layer's activation value, connections are made between the neurons of each layer, creating the neural network. The backpropagation algorithm, shown above in Equation 3, uses forward propagation to compute a hypothesis value for each input data point which is used to find a delta value between the estimate and the actual value from the input label in supervised machine learning.

Equation 5. Forward propagation algorithm

$$a^{(n)} = g(\Theta^{(n-1)}a^{(n-1)})$$

In most cases, designing a suitable feature detector involves an in-depth knowledge into the subject matter being analyzed, while neural networks can be implemented without this requirement. The first research into convolutional neural networks occurred when researchers from AT&T explored their use for recognizing numbers on documents, a solution which is still

used today by the United States Postal Service for reading ZIP codes from envelopes [8]. These models differ from earlier neural models in that they have local receptive fields, shared weights, and subsampling. Receptive fields act as the namesake layer of a convolutional network in small kernels of defined pattern, usually 2x2 or 3x3, usually applied sequentially or concurrently across an image, but can be applied at an interval using a parameter known as stride, which allows the model to find features across the entire input space by reducing neurons [9]. Shared weights significantly reduce the amount of optimization necessary for training, as it only must update one weight instead of a number dependent on the size of the image. By finding the parameters through subsampling, the model gains the ability to respond to variations in the input. Convolutional neural networks have demonstrated an ability to generalize to any input and learn analysis and classification from a plethora of training data.

Related Work

In a recent project performed at the University of Bern, a group of researchers created a deep convolutional neural network architecture for the classification of lung diseases based on lung slices from computed tomography images that performed with an accuracy of 85.5% on its dataset [10]. However, this method trained upon smaller patches inside of the lung image scan which required manual segmentation by radiologists. The ability of deep convolutional neural networks to learn intricate details from an image obviates the need for such a process, which can be replaced by classification of an entire image based on final diagnosis of the patient. This method allows for an increase in automation of the procedure which allows for faster results from screening without a loss in accuracy.

In another project performed at the Federal University of Parana, a convolutional neural network was used for classifying images of cell slides of breast cancer patients [11]. That project

also used patch generation across the image to generate 1,000 patches per image using multiple techniques including random generation and sliding window. However, this approach is not applicable to x-ray scans of the chest because the lungs have a definitive geometric structure and orientation unlike that of cell culture screens, which can have abnormalities in any position. Datasets that are not reliant on geometric information often augment themselves through random rotation and cropping. To accommodate the variation in the PLCO dataset used in this project, rotations of ninety degrees were applied to create a uniform input.

Materials

All design and training of this algorithm was performed on a desktop computer running the Ubuntu Linux 16.04 LTS operating system. The open source nature of this platform has attracted skilled developers and created an ecosystem of quality software that was used in the preparation of this project. This operating system has also executed computer vision software faster than any competing platform due to the techniques used to accelerate disk access, a critical component of training a network due to the large dataset size. Software used in this project was originally designed to operate in this specific environment and some tools recently became available elsewhere but without the performance optimizations available by targeting a specific architecture.

This computer was powered by an Intel i7-6800K processor at 3.6 GHz, 32 GB of RAM, and a solid state hard drive which allowed ruling out bottlenecks in these components. High levels of system memory and a fast storage medium were required for this application which depended on loading a significant number of medical images for training and validation. To store the original dataset, two 2 TB hard drives were used as a source for the resizer. The machine learning algorithms were accelerated by a NVIDIA Pascal Titan X with 12 GB of GDDR5 RAM,

which stored the graph and all the learned parameters of the networks. The large size of the images when expanded into memory necessitated the use of this coprocessor.

To create the network and train it with high efficiency, an optimized neural network framework named TensorFlow was used [12]. This open source software solution was originally created by the Google Brain team for machine learning on textual data. The framework supports running the training operation of the network on traditional computer microprocessors or graphics processing units (GPUs) which have been proven to accelerate training speed, as well as custom integrated circuit logic used internally by Google for their products. This program builds a computational graph using its programming language interface which then executes it on the intended device. It also seamlessly supports several machine learning algorithms with the same optimizer. Relying on a framework to perform the convolutions, pooling and other internal computations used by convolutional neural networks permitted more focus on the design of the network and comparison to others instead of the recreation of existing technology. This program also contains functions for distributed computing, which would further accelerate the process of training in an industrial environment.

Due to its populated library of scientific computing resources and toolkits, the Python programming language version 3.5 was used for all aspects of this project, from loading image data to generating the final diagrams. The TensorFlow framework provided a backend for this language, so that the graphs describing the network architecture and optimizer could all be defined in software. A buffering data batcher was also created to provide images to the training algorithm in a random order to allow additional training after a pass over the entire dataset, or epoch, had already been completed. This solution also relied on the NumPy library for efficient

storage and manipulation of large vectors and the SciPy library for transformations such as uniform image scaling.

Procedure

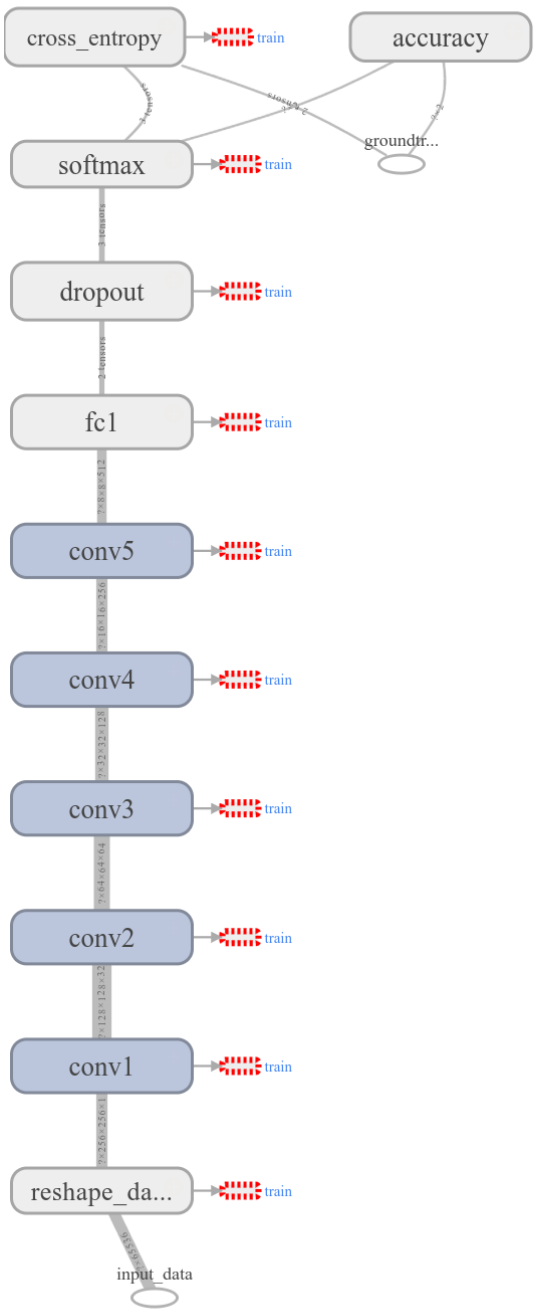
The accuracy of convolutional neural networks is primarily dependent on the size, complexity, and noise of the dataset utilized. In this case, the PLCO dataset available from the National Institute of Health [13] was chosen due to the consistent x-ray data type and confirmed ground truth labels by radiologists, a relatively unique feature among medical databases. The data was provided as a 2.2 TB collection of TIF image files at a resolution of 2000x3000 stored as 16-bit single-channel images which were identified by markers in a spreadsheet. An example image from the dataset is found in Figure 6. To fit the specific parameters of the neural network, and to increase training efficiency, all the images were resized and uniformly scaled to 256x256 floating point single-channel images using the OpenCV library.

Figure 6. A chest x-ray

Next, the loaded images were split into two classes, the first being x-rays linked to a positive lung cancer screening within 30 days and the second being x-rays not linked to lung cancer, based on ground truth values stored in a secondary input file parsed by a Python script. Data was removed from the more populous negative label to balance the data supporting each class, as imbalanced datasets have been known to exhibit suboptimal performance because of the presence of noise and outliers [14]. The data was further divided, with 75% of the images retained for training and 25% of the images set aside for testing the accuracy of the network through softmax regression after the training process. The program was designed to perform testing using 100 iterations of the validation dataset.

The custom network designed for this dataset (shown in Figure 7) was made much simpler than the AlexNet and Inception models as the binary classification required by this problem did not demand much depth, or many layers contributing to complexity. The network consisted of five convolutional layers with rectified linear unit activators, each followed by 2x2 max pooling layers.

Figure 7. Final neural network graph



A common problem in the pursuit of deep learning technologies is the tendency for models to over fit the data, or develop a dependence on the validation dataset. With small datasets, the relationships discovered by the neural network will become the result of sampling

noise, which appears as the accuracy of the network over the validation set will reach a maximum before regressing. An algorithm, known as dropout, has been proven to reduce the effect of overfitting and improve the performance of neural networks in the fields of vision, document classification, and others [9]. This benefits generalization by eliminating neurons and their connections within hidden and input layers based upon a constant, predefined probability each iteration. However, all neurons and connections of the network are active during the validation stage. This project employs a dropout strategy with the probability set to 50%, a value said to perform generally well across implementations of all fields. This step was performed during the second fully-connected layer following the hidden layers of the network. In total, this model contained 37,911,298 trainable parameters.

To train the network, 10,000 iterations of the optimizer were executed. Each iteration consisted of loading 16 images through a consistent, randomized batch loader to ensure adequate memory for parameter storage and to prevent the optimizer from focusing on runs in the data. Every 10 iterations, the accuracy was calculated against the training dataset and recorded. For the remainder of the iterations, the program ran a training step which performed backpropagation of the network over the data, updating the weights and biases of the network allowing the model to learn. The dropout layer effectively reduced the chance of overfitting and diverging. The model was trained through 28 epochs of the dataset, or complete repetitions of all the images.

Data

Testing of the network was performed on the model after successful training by backward propagation. A forward pass was performed on 100 data points selected from the dataset delineated for use in testing and the predicted output of the classification was recorded. These values were then compared to the ground truth values associated with each data point, and the

amount of predictions that were consistent with the actual values were divided by the total amount of data points tested to find the accuracy of the network. This accuracy was computed for every 10 iterations of the network during training to create a graph shown in Figure 8 of the change in accuracy versus steps.

Figure 8. Change in accuracy versus iterations of training data

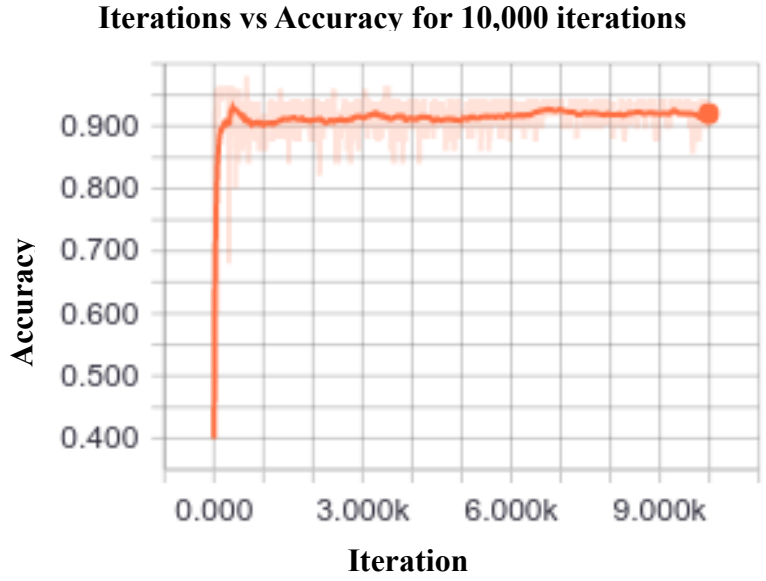


Figure 8 shows the progress of the neural network as it trains over 10,000 iterations of the dataset with a batch size of 16 images per iteration. As shown, the model rapidly accelerates in training performance within the first epoch to the baseline accuracy and increases in precision as the model continues to train. The amount of time it takes for a network to complete a full course of training traditionally takes on the order of hours or days. The relatively quick training of this model is attributed to its binary output, in contrast to a large network such as GoogLeNet which contains 1,000 outputs and necessarily takes more time per iteration. Figure 9 presents the total training time for this network and change in time per iteration.

Figure 9. Time difference between iterations

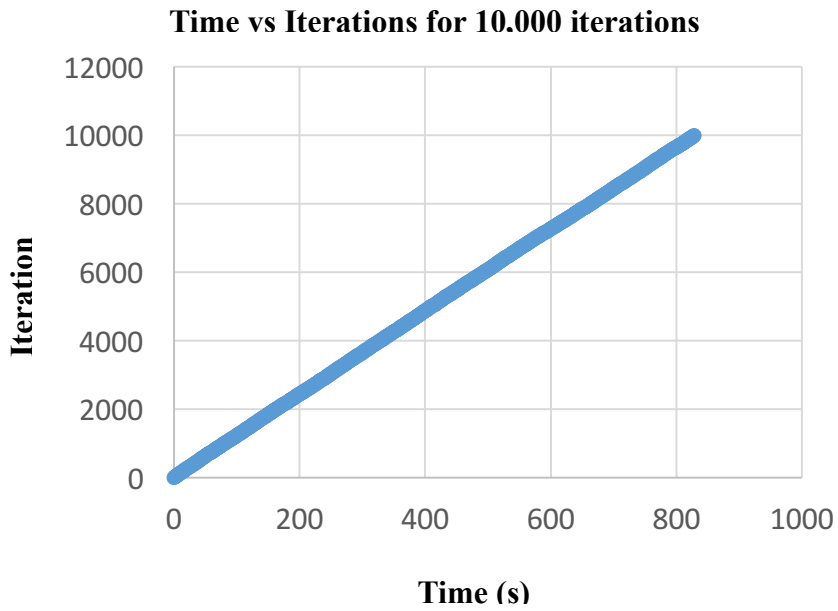


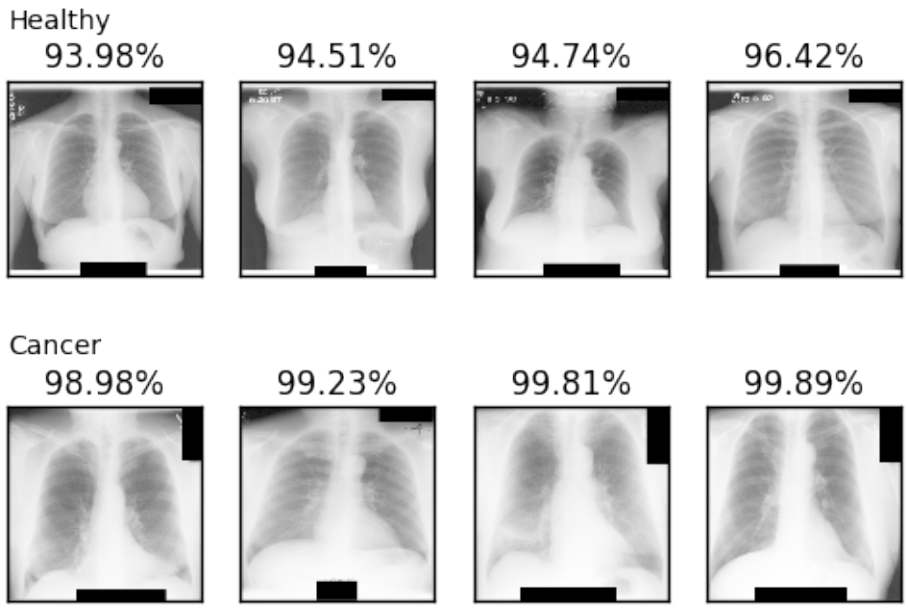
Figure 9 demonstrates the intuition that each iteration executes in constant time intervals. Taking the derivative of the above gives a constant in terms of iterations per second, which shows that the algorithm could train at a rate of around 12 iterations per second, or 192 training samples per second, with full data logging enabled.

Results

To generate the following visualization, four images from both classes were first selected. Then, the network was used to compute the confidence scores, a measurement taken directly from the neural network's softmax layer that expresses the amount of guesswork versus educated

prediction that the network made when determining this prediction. Each positive and negative image is a rescaled x-ray from the validation dataset.

Figure 10. Chart showing example validation set images from both classes and the output confidence scores



The confidence scores demonstrate that the network's knowledge of the parameters enabled it to make predictions that it perceived as accurate, rather than random choices.

For binary classifiers, such as this model, a matrix can be used to present several statistics about the performance of the algorithm. The core of this matrix is the contingency table, which is used to present the frequency of the real condition variable and the predicted condition variable. From this crosstabulation many ratios can be derived, most notably the descriptions of false positives and negatives as well as their respective rates among the population. To generate the following matrix, the network was computed against all 1,884 validation samples and the ground

truth label was recorded with the predicted output. The statistical ratios were calculated from these inputs.

Table 1. Binary classifier evaluation contingency table and confusion matrix followed by key

1884	1096	788	0.581740977	
1096	1013	83	92.43%	7.57%
788	83	705	10.53%	89.47%
	0.924270073	10.53%	877.50%	
91.19%	7.57%	0.894670051	8.46%	10366.74%

Total Population	Predicted Positive	Predicted Negative	Prevalence	
Real Positive	True Positive	False Negative	True positive rate	False negative rate
Real Negative	False Positive	True Negative	False positive rate	True negative rate
	Positive predictive rate	False omission rate	Positive likelihood ratio	
Accuracy	False discovery rate	Negative predictive value	Negative likelihood ratio	Diagnostic odds ratio

The results of this test demonstrate the model's accuracy as found earlier as well as its tendency to avoid false positives and negatives.

Conclusion

The final accuracy of the network over the validation set of 91% vastly surpassed the benchmark of 50% for random guessing on the two-class dataset which shows that the model has succeeded at learning patterns related to the presence of various types of confirmed lung cancer in these images. Note that a common machine learning pre-processing step, known as mean

value subtraction, was not used in this implementation. This would involve precomputation of the average value of the entire training dataset and a subtraction step before each training operation. For a particular dataset, this often improves training accuracy as the parameters are being optimized directly on the unique properties of each image, however this application achieved success without reliance on this process. A future inquiry could test whether mean value subtraction would impact the ability of a model to adapt to the variation of new input.

Discussion

Given that a human radiologist must spend a considerable amount of time on each image to make a correct prediction, screenings for many types of cancer do not often occur early, causing diagnoses to often arrive during the late stages of these diseases. As this model using TensorFlow could process images through a neural network at a speed of 3.41 milliseconds each, the potential to use such a model as a preliminary screening step may save many lives with early detection and from misdiagnoses. All cases of concern will necessarily have to be verified by an experienced radiologist, but the automation provided by this tool will decrease costs, increasing the accessibility of screenings, as well as increase the speed and accuracy of diagnoses.

Future areas of improvement to this solution include supporting predictions based on CT scans of potentially affected regions of the body. As shown by the National Lung Screening Trial [15], this type of imagery may have a higher potential for identifying cancerous tumors due to the larger dataset based on 3D imagery. Further research will be necessary to find the best method of representing this slice-based data, whether by expressing the data in vector form as done for the 2D imagery in this project or with knowledge about the location of each slice in the patient. As the technology behind these scans continues to improve, as shown through the

introduction of high resolution yet low-dose capture techniques, the importance of automated CT screenings will increase.

Bibliography

[1] V. Ambrosini, S. Nicolini, P. Caroli, C. Nanni, A. Massaro, M. C. Marzola, D. Rubello and S. Fanti, "PET/CT imaging in different types of lung cancer: An overview," *European Journal of Radiology,* vol. I, no. 81, pp. 988-1001, 2012.

[2] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision,* vol. 60, no. 2, pp. 91-110, 2004.

[3] M. Firmino, A. H. Morais and R. M. Mendoca, "Computer-aided detection system for lung cancer in computed tomography scans: Review and future prospects," *BioMedical Engineering OnLine,* vol. 13, no. 41, pp. 1-16, 2014.

[4] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision,* vol. I, no. 115, pp. 211-252, 2015.

[5] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, 2012.

[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan and V. Vanhoucke, "Going Deeper with Convolutions," in *Computer Vision and Pattern Recognition*, Boston, 2015.

[7] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization," in *International Conference on Learning Representations*, San Diego, 2015.

[8] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278-2324, 1998.

[9] N. Srivastava, G. Hinton and A. Krizhevsky, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research,* vol. I, no. 15, pp. 1929-1958, June 2014.

[10] M. Anthimopoulos, S. Christodoulidis, L. Ebner, A. Christe and S. Mougiakakou, "Lung Pattern Classification for Interstitial Lung Diseases Using a Deep Convolutional Neural Network," *IEEE Transcations on Medical Imaging,* vol. 35, no. 5, pp. 1207-1216, May 2016.

[11] L. Oliveira, F. Spanhol, C. Petitjean and L. Hautte, "Breast Cancer Histopathological Image Classification using Convolutional Neural Networks," in *International Joint Conference on Neural Networks*, Vancouver, 2016.

[12] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G., B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu and X. Zheng, "TensorFlow: A System for Large-Scale Machine Learning," in *USENIX Symposium on Operating Systems Design and Implementation*, Savannah, 2016.

[13] C. S. Zhu, P. F. Pinsky, B. S. Kramer, P. C. Prorok, M. P. Purdue, C. D. Berg and J. K. Gohagan, "The Prostate, Lung, Colorectal, and Ovarian Cancer Screening Trial and Its

Associated Research Resource," *JNCI Journal of the National Cancer Institute,* vol. 105, no. 22, p. 1684–1693, 2013.

[14] B. Lakshmanan, A. J. Priscilla, S. Ponni and V. Sankari, "Evaluation of Imbalanced Datasets using Fuzzy Support Vector Machine-Class Imbalance Learning," in *International Conference on Recent Trends in Information Technology*, Chennai, 2011.

[15] D. Aberle, A. Adams, C. Berg, W. Black, J. Clapp, R. Fagerstrom, I. Gareen, C. Gatsonis, P. Marcus and J. Sicks, "Reduced lung-cancer mortality with low-dose computed tomographic screening," *New England Journal of Medicine,* vol. 365, no. 5, pp. 395-409, 2011.

[16] H. Wang, A. Cruz-Roa, A. Basavanhally, H. Gilmore, N. Shih, M. Feldman, J. Tomaszewski, F. Gonzalez and A. Madabhushi, "Mitosis Detection in Breast Cancer Pathology Images by Combining Handcrafted and Convolutional Neural Network Features," *Journal of Medical Imaging,* vol. 1, no. 3, October 2014.